

# Instrukcja obsługi

## Lord of User Interface

Dla wersji programu 3.0.9

Krzysztof Mroczek

1 lutego 2015

# Spis treści

<b>1</b>	<b>Instrukcja obsługi w pigułce</b>	<b>2</b>
<b>2</b>	<b>Informacje ogólne</b>	<b>3</b>
<b>3</b>	<b>Licencja programu</b>	<b>4</b>
3.1	Sklep internetowy LordUI . . . . .	5
3.2	Podpisywanie procedur . . . . .	5
<b>4</b>	<b>Instrukcja obsługi</b>	<b>7</b>
4.1	Zapisywanie projektu . . . . .	7
4.1.1	Pliki LUI . . . . .	7
4.1.2	Pliki FUI . . . . .	7
4.1.3	Pliki XML . . . . .	7
4.2	Widok programu Lordui . . . . .	8
4.2.1	Panel Listy odtwarzania . . . . .	9
4.2.2	Zmienne globalne . . . . .	9
4.3	Narzędzia Lordui . . . . .	10
4.3.1	Procedury . . . . .	10
4.3.2	Odtwarzacze procedur . . . . .	12
4.3.3	Nasłuchiwanie . . . . .	12
4.4	Lordui jako biblioteka Java . . . . .	13
4.4.1	Interfejs biblioteki Lordui . . . . .	13
4.4.2	Podłączanie biblioteki do różnych rodzajów aplikacji . .	14
4.5	Wyrażenia . . . . .	17
4.5.1	Typy wyrażeń . . . . .	18
4.5.2	Edycja wyrażeń . . . . .	18
4.5.3	Zmienne obiektów . . . . .	19
4.5.4	Funkcje na wyrażeniach . . . . .	22
4.5.5	Przykłady wyrażeń . . . . .	29
4.6	Komendy procedur . . . . .	29
4.6.1	Strzałki . . . . .	30
4.6.2	Ikonki paska start . . . . .	30
4.6.3	Kliknięcie użytkownika . . . . .	32

4.6.4	GUI . . . . .	33
4.6.5	Rysowanie . . . . .	34
4.6.6	Obsługa obrazu . . . . .	35
4.6.7	Urządzenia wejścia (operacje myszy i klawiatury) . . .	39
4.6.8	Baza danych . . . . .	43
4.6.9	Wsparcie obiektów Java . . . . .	44
4.6.10	Komenda SO . . . . .	44
4.6.11	Socket . . . . .	46
4.6.12	Integracja . . . . .	47
4.6.13	Dźwięk . . . . .	48
4.6.14	Sound and Video . . . . .	49
4.6.15	Tworzenie obiektów lordui . . . . .	50
4.6.16	Lordui meta-elementy . . . . .	52
4.6.17	Polecenia standardowe . . . . .	53
4.6.18	Rozszerzenia . . . . .	58
4.7	Tworzenie klasycznych makr . . . . .	58
4.8	Przygotowywanie zmiennych obrazkowych . . . . .	59
4.9	Natywna część Lordui . . . . .	59
4.9.1	Rozmycie obrazu . . . . .	59
4.9.2	Operacje natywne . . . . .	60
4.9.3	Przezroczyste okna . . . . .	60
4.10	Moduły Lordui . . . . .	60
4.10.1	Moduł Zdalnej Java'y . . . . .	60

## Rozdział 1

# Instrukcja obsługi w pigułce

Lordui jest narzędziem, o którym należy myśleć obiektowo. Głównym składnikiem jest **procedura**. Procedury można uruchamiać tworząc **procesy**. Procesów może działać wiele jednocześnie, jednak tylko jeden może być w danej chwili aktywny. Procedury pogrupowane są w **pakiety**. Składają się zaś z różnych **komend** tj. spanie, akcja urządzenia wskazującego, pętla, obsługa obrazu, obsługa dźwięku itd.

## Rozdział 2

# Informacje ogólne

Autorem programu jest Krzysztof Mroczek. W programie zostały jednak zawarte liczne biblioteki zewnętrzne, tworzone przez osoby trzecie. Są to m.in.:

- kSquaredHook - biblioteka do obsługi zdarzeń urządzeń wejściowych rozwijana przez Panów: Kristian Kraljic oraz Johannes Schüth.
- jLayer - biblioteka do obsługi plików mp3
- grammatica - biblioteka do parsowania wyrażeń regularnych

## Rozdział 3

# Licencja programu

Licencja na jakiej udostępniony jest program 'Lordui', zwanego dalej 'programem', składa się z poniższych punktów:

1. Ten program został stworzony przez Krzysztofa Mrocza. Krzysztof Mroczek jest jego autorem i do niego należą prawa autorskie programu Lordui. Autor zastrzega sobie możliwość zmian w programie, rozwijanie go, edytowanie, testowanie, zmiany funkcjonalne (w tym dodawanie oraz odejmowanie funkcjonalności).
2. Ten program można kopiować oraz instalować na różnych komputerach. Niedozwolone jest jednak kopiowanie wraz z programem kluczy licencyjnych - każda kopia programu oraz każdy użytkownik systemu operacyjnego wymaga osobnego wykupienia klucza licencyjnego.
3. Zabroniona jest dekompilacja, jakakolwiek zmiana, zastosowanie programu lub jego części w innym oprogramowaniu oraz dystrybucja zmienionej lub oryginalnej wersji programu Lordui. Wyjątek stanowią jedynie wykorzystane w programie biblioteki, których licencja stanowi inaczej.
4. Źródła programu oraz cały jego kod jest własnością autora programu. Zabronione jest ich używanie do jakichkolwiek celów - zarówno prywatnych jak i komercyjnych bez specjalnej zgody autora. Wyjątek stanowią jedynie użyte przy tworzeniu programu Lordui biblioteki, których licencja stanowi inaczej.
5. Niedozwolone jest czerpanie korzyści finansowych ze sprzedaży, dystrybucji, szkoleń czy udostępniania w dowolnych mediach programu Lordui.
6. Punkt 5) nie obowiązuje w przypadku specjalnego pozwolenia autora programu Lordui.

7. Ten program jest dystrybuowany taki jaki jest. Nie zostają udzielone żadne gwarancje jego poprawności. Wszelka odpowiedzialność za jakiegokolwiek szkody, braki czy straty powstałe w wyniku korzystania z niego należy do końcowego użytkownika programu. W szczególności dotyczy to odpowiedzialności za stracone dane, szkody wyrządzone na systemie plików czy też niemożliwość korzystania z programu.
8. Użytkownik ponosi pełną odpowiedzialność za korzystanie z programu. Autor programu nie ponosi żadnej odpowiedzialności za jakiegokolwiek łamanie prawa z wykorzystaniem programu Lordui.
9. Niedozwolona jest ingerencja w treść powyższej licencji oraz odrywanie programu od powyższej licencji - każda otrzymana kopia programu objęta powyższą licencją niezależnie od wykonywanych działań musi pozostać objęta powyższą licencją.

### **3.1 Sklep internetowy LordUI**

Podstawowa wersja programu LordUI jest jednocześnie wersją demonstracyjną i może być wykorzystywana tylko do użytku prywatnego. Wersja ta nie pozwala zapisywać projektów o więcej niż jednej procedurze, otwierać projektów w formacie XML, a dodatkowo wstrzymuje program co 10 minut. Aby pozbyć się powyższych ograniczeń, można wykupić kody w sklepie internetowym. Więcej informacji znajduje się na stronie [www.lordui.com/pl/sklep](http://www.lordui.com/pl/sklep).

### **3.2 Podpisywanie procedur**

Istnieje możliwość podpisania procedur kluczem. Podpisywaną procedurę można uruchamiać bez wstrzymywania co 10 minut na dowolnym komputerze. Dzięki temu można m.in. tworzyć procedury pełniące funkcjonalność instrukcji obsługi/tutorialu/demonstracji. Aby podpisać procedurę należy zaznaczyć procedurę i w sekcji „Kody procedur” wybrać opcję „Generuj”. W okienku pojawi się wygenerowany kod. Należy go wysłać do autora programu drogą mailową (kontakt znajduje się na stronie internetowej). Po otrzymaniu plątności oraz wygenerowanego kodu, zostanie wysłany mail zwrotny z „Kodem użytkownika”. Należy go wprowadzić używając opcji „Wprowadź kod”. Podpisanie procedury należy powtórzyć po każdej zmianie wykonanej w dowolnym elemencie procedury. Cena usługi jest ustalana indywidualnie. Autor programu zastrzega sobie prawo odmówienia podpisania procedury bez podania przyczyny.

Procedury podpisane według powyższego opisu można odtwarzać bez ograniczeń na dowolnym komputerze. Służy do tego specjalnie zaimplementowany odtwarzacz dostępny na stronie internetowej: [www.lordui.com/pl/pobieranie](http://www.lordui.com/pl/pobieranie).

Odtwarzacz jest lżejszy od edytora Lordui, a do tego da się go skonfigurować tak, by procedura była odtworzona bez pokazywania żadnych meta-elementów Lordui. Syntaktyka argumentów odtwarzacza jest następująca:

```
[-file <ścieżka do pliku *.lui>|-url <url pliku *.lui>] <Nazwa procedury do odtworzenia>
```

.



## Rozdział 4

# Instrukcja obsługi

### 4.1 Zapisywanie projektu

Dostępne są dwa różne formaty projektów LordUI: XML oraz lui. Projekt można zapisać w dowolnym z nich wybierając Plik->Zapisz (wybierz format z listy w dolnej części okna dialogowego). Pliki tych formatów można także wczytać wybierając Plik->Otwórz (Domyślnie w oknie dialogowym pokazują się tylko pliki \*.lui - zmień format za pomocą listy w dolnej części okienka). Przy zapisywaniu projektu możemy wybrać tylko część danych projektowych posługując się dostępnymi filtrami procedur i zmiennych globalnych. Dodatkowo możemy zapisać w projekcie aktualnie wybraną paletę oraz dołączyć do projektu skompilowaną wersję procedur.

#### 4.1.1 Pliki LUI

Pliki LordUI (LUI) są domyślnymi plikami do przechowywania projektów LordUI. Oprócz danych projektowych (procedury czy zmiennych globalnych) mogą (ale nie muszą) zawierać dodatkowo paletę komend procedury i skompilowaną wersję projektu. Skompilowana wersja projektu służy wystawianiu lżejszej wersji (o biblioteki służące do kompilacji) programu lordui przy odtwarzaniu procedur (np. przy udostępnianiu projektów Lordui przez internet).

#### 4.1.2 Pliki FUI

Format FUI nie jest już dostępny. Od Lordui wersji 3.0.4 został on wchłonięty przez format LUI.

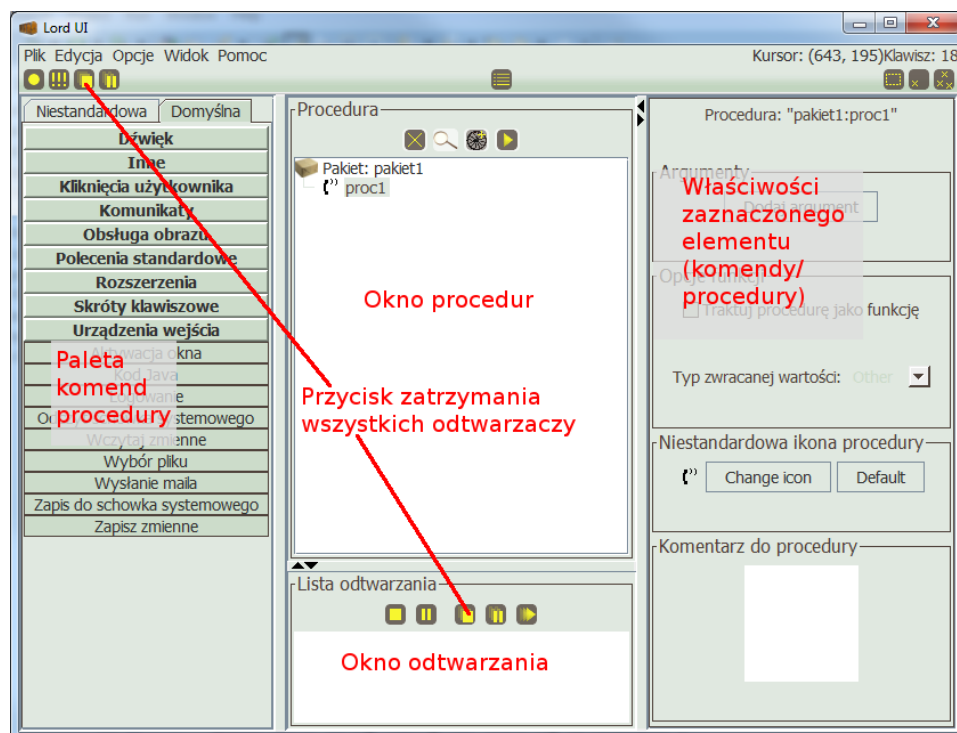
#### 4.1.3 Pliki XML

Pliki XML są plikami tekstowymi o określonej strukturze. Obsługa plików XML została zaimplementowana z myślą o użytkownikach ceniących szczegól-

nie możliwość programowania w notatniku, jednak nie jest on zalecany do codziennej pracy. Pliki XML są bardzo duże i wymagają wprowadzenia dużej ilości tekstu. Edytor tekstowy nie udostępnia także możliwości np. oglądania obrazków projektu.

## 4.2 Widok programu Lordui

Okno powitalne programu Lordui jest zarazem oknem głównym, który udostępnia większość funkcji potrzebnych przy pracy. W poniższym rozdziale zostaną zaprezentowane podstawowe funkcje dostępne w oknach programu. Okno









Rysunek 4.1: Podstawowe elementy okna głównego Lordui

główne składa się z kilku paneli. Są to:


- Paleta komend dostępna w lewej części ekranu. Są to poszczególne operacje, z których budowane będą procedury Lordui (Palety można edytować wybierając „Edycja” -> „Edytuj paletę komend”),
- Okno procedur, które prezentuje w formie drzewa cały projekt,
- Na dole znajduje się Lista Odtwarzania, pokazująca wszystkie aktywne procesy
- Właściwości wybranego obiektu (po prawej stronie)




### 4.2.1 Panel Listy odtwarzania

Lista odtwarzania składa się z paska narzędzi oraz listy aktywnych procesów. Aktywne procesy mogą zawierać uruchomione procedury. Gdy proces jest wstrzymany („zapauzowany”), klikając prawym przyciskiem myszy na dowolną jego procedurę możemy zobaczyć m.in. stos wywołań procedur czy podejrzeć pamięć procedury - czyli wszystkie obiekty dostępne w procedurze. Na pasku narzędzi znajdują się następujące przyciski:

-  - zatrzymanie zaznaczonego procesu (odtwarzacza procedur),
-  - wstrzymanie/wznowienie zaznaczonego procesu (odtwarzacza procedur),
-  - zatrzymanie wszystkich procesów,
-  - wstrzymanie wszystkich procesów,
-  - wznowienie wszystkich procesów,
-  - wykonanie pojedynczego kroku zaznaczonego procesu - dostępne, gdy zaznaczony proces jest wstrzymany.

### 4.2.2 Zmienne globalne







Zmienne globalne są to zmienne dostępne wszystkim procesom. Aby edytować zmienne globalne należy z paska narzędzi w górnej części okna głównego programu wybrać ikonkę .

Okno zmiennych globalnych składa się z drzewa zmiennych (zmienne mogą być grupowane w pakiety) z lewej strony okna oraz z okna edycji wybranej zmiennej z prawej strony okna. Zmienne globalne możemy dodawać, usuwać lub zmieniać ich nazwę klikając je prawym przyciskiem myszy, bądź za pomocą ikon z paska narzędzi nad drzewem zmiennych (Guziki: , , ).

Z prawej strony okna edycji zmiennych globalnych widać wartość wybranej zmiennej. Mogą to być liczby, napisy (Uwaga! Napisy wpisujemy otaczając je podwójnym cudzysłowem), wartości logiczne, ale także obrazki, dźwięki czy tablice składające się obiektów różnych typów. Np.

```
[1, false, ,,text'']
```


to tablica trzelementowa, składająca się z liczby (1), wartości logicznej (false) i napisu („text”). Niektóre typy obiektów (np. obrazek) mają możliwość edycji w dedykowanym widoku edycji, jednak podstawowy widok składa się z pola edycji oraz paska narzędzi nad polem edycji. Nad paskiem narzędzi znajdują się następujące przyciski:

-  - wstawia w miejsce kursora obrazek. Obrazek może zostać wczytany z pliku, bufora systemowego (tzw. clipboard) lub skopiowany z fragmentu wyświetlanego obrazu na ekranie komputera,
-  - wstawia w miejsce kursora dźwięk. Dźwięk można wczytać z pliku (wav, mp3) lub nagrać (tylko wav),
-  - wstawia w miejsce kursora punkt (współrzędne punktu). Współrzędne wprowadza się klikając wybrane miejsce ekranu. Współrzędne można wprowadzić też ręcznie w polu edycji w formacie (<wartość współrzędnej X>, <wartość współrzędnej Y>),
-  - wstawia w miejsce kursora obszar (współrzędne punktu). Współrzędne zaznaczając wybrany obszar myszką. Współrzędne można wprowadzić też ręcznie w polu edycji w formacie (<wartość współrzędnej X>, <wartość współrzędnej Y>, <szerokość>, <wysokość>),
-  - cofnięcie zmian edycji - przywrócenie wartości sprzed rozpoczęcia edycji zmiennej,
-  - przełączenie widoku edycji zmiennej pomiędzy standardowym, a dedykowanym typowi wprowadzonej zmiennej,
- Typ wprowadzonego wyrażenia.

Gdy wprowadzone wyrażenie nie jest obsługiwane, lub jest błędne program pokaże błąd w dolnej części okna edycji zmiennej

## 4.3 Narzędzia Lordui

### 4.3.1 Procedury

Aby móc pisać skrypty w programie Lordui należy najpierw stworzyć procedurę. Procedurę można utworzyć za pomocą przycisku , lub też klikając prawym przyciskiem myszy w okienku procedur (lewe górne białe okno) oraz wybierając opcję „Dodaj procedurę”. W nazwie procedur powinny znajdować się tylko litery, cyfry i dwukropki. Każdy dwukropek traktowany jest jako separator kolejnych składowych, przy czym ostatnia składowa jest nazwą procedury, a wcześniejsze składowe to nazwy kolejnych pakietów, w których procedura się znajduje np.: podając nazwę procedury „pakiet1:pakiet2:proc” stworzymy pakiet1, który będzie zawierał pakiet2, w którym zaś znajdzie się procedura o nazwie „proc”.

Procedury mogą mieć argumenty. Aby edytować argumenty procedury, należy zaznaczyć procedurę w oknie procedur. Po dodaniu argumentu należy podać jego typ oraz nazwę. **Zaleca się przypisywanie w całym projekcie do jednej zmiennej wartości tego samego typu.** Procedurę można


usuwać, zmieniać jej nazwę (czyli także przesuwać do innego pakietu), duplikować. Dostępna jest także opcja wyszukania wszystkich wywołań procedury.


W czasie, gdy uruchomiona jest dowolna procedura, dostępne są dwa bardzo ważne skróty klawiaturowe:


1. **Scroll lock - przerywa wykonanie wszystkich działających procesów (odtwarzaczy)**
2. **Pause/break - wstrzymuje lub ponownie uruchamia wykonywanie wszystkich procesów. (odtwarzaczy)**

## Uruchamianie procedury

Jest kilka sposobów uruchamiania procedur.

- Pierwszym sposobem jest wskazanie procedury, bądź jednej z jej komend w oknie procedury, a następnie kliknięcie przycisku  w toolbarze. Wyświetli się okno, w którym można ustawić liczbę wywołań procedury, minimalny i maksymalny odstęp czasowy pomiędzy wywołaniami procedury. Dostępna jest także nazwa odtwarzacza na którym procedura ma zostać uruchomiona. Czas do pierwszego wywołania procedury, a także wartości zmiennych wywoływanej procedury.
- Drugim sposobem wywołania procedur, jest zdefiniowanie skrótów klawiszowych: „Edycja” -> „Skróty klawiszowe”. Ustawienia zmiennej, liczby wywołań i odstępów pomiędzy wywołaniami znajdują się pod przyciskiem „Ust.” każdego wiersza okna edycji przycisków klawiszowych. Zwracam uwagę na opcję „Skróty aktywne podczas gdy uruchomiona jest którakolwiek procedura” w oknie edycji skrótów klawiszowych oraz opcję „Skróty klawiszowe aktywne” w menu „Edycja”. Jeśli którakolwiek z tych opcji nie jest spełniona, skrót klawiszowy nie będzie działać.
- Kolejną możliwością jest uruchomienie procedury przy pomocy innej procedury. Aby skorzystać z tej funkcji należy dodać odpowiednią komendę do procedury (patrz np. 4.6.17: „Wywołanie procedury”) oraz posiadać tą procedurę dodaną do odtwarzacza.
- Procedury mogą także być zlecane wątkom automatycznie na skutek akcji użytkownika gdy któraś z procedur wcześniej uruchomi nasłuchiwanie tej akcji (patrz rozdział 4.3.3: „Nasłuchiwanie”).

Procedury można w dowolnym momencie wstrzymać oraz zatrzymać. Aby zatrzymać wszystkie uruchomione procedury należy wcisnąć klawisz klawiatury **Scroll lock** lub przycisk . Można też zatrzymać wybrany proces, klikając go prawym przyciskiem myszy i wybierając opcję „Zatrzymaj ten proces”.

Aby wstrzymać wykonywanie procesów należy wcisnąć klawisz klawiatury **Pause/break** lub przycisk . Gdy procesy są wstrzymane dostępne są dodatkowe narzędzia debugowania. Wskaż prawym przyciskiem myszy proces na liście procesów.

1. Wybierz „Pokaż pamięć”, aby zobaczyć wartość wszystkich zmiennych trzymany w pamięci
2. Wybierz „Pokaż stos wywołań”, aby zobaczyć aktualny stos wywołań procedur.

#### 4.3.2 Odtwarzacze procedur

Procedury uruchamiane wykonywane są przez odtwarzacze. Każda procedura uruchamiana jest w osobnym wątku. Każda procedura może wykonywać jednocześnie tylko jeden wątek. Pozostałe wątki czekają aż poprzedni wątek zakończy wykonywanie lub wstrzyma pracę (patrz komenda 4.6.17 „Spanie”). Do odtwarzacza można dodać procedurę na wiele sposobów - zarówno ręcznie jak i automatycznie (patrz rozdział 4.3.1: „Uruchamianie procedury”).

Każdy wątek dodany do odtwarzacza dysponuje własną pamięcią podręczną. Jednocześnie jednak procedury mogą sięgać do głównej pamięci. Domyślnie odtwarzacz zostaje stworzony podczas dodawania do niego pierwszej procedury. Zostaje zaś zatrzymany po usunięciu ostatniego elementu. Niektóre odtwarzacze (np. niektóre nasłuchujące akcji użytkownika) nie zostaną zamknięte gdy nie mają żadnego zaplanowanego elementu i wymagają ręcznego zamknięcia (co jest jednoznaczne z zaprzestaniem nasłuchiwanie).

#### 4.3.3 Nasłuchiwanie

Nasłuchiwanie są to specjalne obiekty służące do obserwacji np. akcji użytkownika. Nasłuchiwanie można uruchomić np. za pomocą komendy 4.6.7: „Nasłuchiwanie klawiatury” czy komendy 4.6.7: „Nasłuchiwanie myszy”. Uruchomiony nasłuchiwanie jest przypisany do określonego odtwarzacza. Za każdym razem, gdy wystąpi zdarzenie na które czeka nasłuchiwanie, zostaje uruchomiona wybrana procedura (lub bardziej formalnie: do wybranego odtwarzacza zostanie dodany wątek wykonywania wybranej procedury). Aby zatrzymać nasłuchiwanie należy zatrzymać odtwarzacz, do którego jest podpięty nasłuchiwanie (uwaga: taki odtwarzacz nie zatrzymuje się automatycznie,

po zakończeniu wykonywania wątku) lub też wywołać komendę 4.6.16: „Zatrzymaj nasłuchiwanie”.

## 4.4 Lordui jako biblioteka Java

Lordui można zintegrować z dowolną aplikacją napisaną w języku Java. Umożliwia to obsługę elementów graficznych aplikacji za pomocą referencji pomiędzy obiektami, co z kolei przekłada się na łatwość implementacji, lepszą jakość projektów oraz szybsze działanie skryptów. Możliwa jest także komunikacja pomiędzy Lordui oraz aplikacją. Aplikacja obsługuje aktualnie jedynie interfejs biblioteki Swing. Na życzenie autor Lordui może rozważyć implementację dowolnej innej biblioteki graficznej.

### 4.4.1 Interfejs biblioteki Lordui

Interfejs biblioteki Lordui ogranicza się praktycznie do jednej klasy:

```
ktm.lordui.Lordui
```

. Instancję klasy Lordui uzyskuje sięwołając statyczną metodę *createInstance*. Można stworzyć tylko jedną instancję klasy Lordui - każde kolejne wywołanie metody *createInstance* zwróci ten sam obiekt co przy pierwszym wywołaniu. Po zakończeniu pracy należy zaś wywołać metodę *close*. Poniżej znajduje się lista dostępnych funkcji na obiekcie klasy Lordui:

1. *public final void loadProject(File luiFile) throws IOException* - wczytuje projekt z podanego pliku \*.lui,
2. *public final void setValue(String name, Object value)* - zapisuje w Lordui dany obiekt (tj. klasy Window, int, String czy Point) pod podaną zmienną globalną,
3. *public final Object getValue(String name)* - pobiera z Lordui obiekt z podanej zmiennej globalnej,
4. *public final void runProcedure(String procedureName)* - uruchamia podaną procedurę,
5. *public final void runProcedureAndWait(String procedureName)* - uruchamia podaną procedurę. Funkcja *runProcedureAndWait* zakończy działanie dopiero gdy odtwarzacz na którym została uruchomiona podana procedura zakończy działanie wszystkich procedur,
6. *public static final Lordui createInstance()* - tworzy instancję klasy Lordui,

7. *public final void close()* - zakończenie pracy z biblioteką Lordui. Bez wywołania tej metody, Lordui może nieprawidłowo zwolnić zasoby. Po wywołaniu tej procedury nie należy już więcej wykorzystywać obiektu Lordui, ani wywoływać funkcji *Lordui.createInstance*,
8. *public final void setVisible(boolean visible)* - pokazuje/ukrywa okno edytora Lordui.

W ten sposób przykładowy kod wykorzystujący Lordui jako bibliotekę ogranicza się do:

```
public class LorduiLibraryUsage {
    private Lordui lui;

    public void useLordui() {
        lui = Lordui.createInstance();
        try {
            lui.loadProject(new File("myLorduiProcedureFile.lui"));
        } catch (IOException e) {
            e.printStackTrace();
        }
        return;
    }

    //Tutaj należy umieścić kod wyświetlający naszą aplikację np:
    try {
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                JFrame fr = new JFrame("Test window");
                fr.setSize(800, 600);
                fr.setVisible(true);
            }
        });
    } catch (InvocationTargetException e) {
        e.printStackTrace();
    }
    return;
} catch (InterruptedException e) {
    e.printStackTrace();
}
return;
}

lui.runProcedureAndWait("LorduiProcedureName");
lui.close();
}
```



#### 4.4.2 Podłączanie biblioteki do różnych rodzajów aplikacji

Przed przeczytaniem tego rozdziału, proszę sprawdzić funkcjonalność 'Zdalna Java' (patrz: 4.10.1: "Moduł Zdalnej Java'y").

Istnieją różne możliwości podłączenia biblioteki Lordui do projektu. Aby podłączyć Lordui nie musimy modyfikować, ani nawet posiadać dostępu do kodu źródłowego naszej aplikacji. Zanim jednak podłączymy Lordui do naszego kodu należy upewnić się, że nie łamiemy w ten sposób żadnej z licencji.

##### Aplikacja desktop-owa/Aplikacja typu webstart

Przez aplikację desktopową rozumiem klasyczny plik \*.jar z zaszytymi informacjami nt klasy głównej. Taki plik można np. uruchomić wywołując komendę `java -jar mojPlik.jar`. Komenda ta uruchomi aplikację rozpoczynając wykonanie od określonej klasy zawierającej metodę `main(String[] args)` np.:

```
public class MainClass {
    public static main(String[] args) {
        //Run my app
    }
}
```

W takiej sytuacji należy stworzyć sztuczną klasę np.:

```
public class MainLorduiClass {
    public static main(String[] args) {
        ktm.lordui.Lordui lui = ktm.lordui.Lordui.createInstance();
        try {
            lui.loadProject(new File("myLorduiProcedureFile.lui"));
        } catch (IOException e) {
            e.printStackTrace();
            return;
        }
        lui.setVisible(true);
        MainClass.main(args);
        lui.runProcedureAndWait("LorduiProcedureName");
    }
}
```

Należy zwrócić uwagę na brak komendy `close()` wywołanej na końcu. Jeśli tylko istnieje możliwość, komenda ta powinna zostać na koniec pracy wywołana, zaś w przeciwnym przypadku zalecane jest ręczne zamknięcie okna edytora przed zamknięciem aplikacji (o ile aplikacja zamykana jest np. komendą

```
System.exit(0)
```

). Tak przygotowaną klasę należy opakować w plik jar i traktować jako główny plik uruchamialny aplikacji.

## Applet

W przypadku applet-ów metoda postępowania jest zupełnie analogiczna, jak w przypadku aplikacji. Założmy więc, że applet, do którego chcemy się podłączyć uruchamia się poprzez klasę:

```
public class EntryApplet extends JApplet {
    @Override
    public void init() {
        //Some init stuff
    }

    @Override
    public void destroy() {
        //Some destroy stuff
    }
}
```

W takim przypadku należy zrobić osobną klasę:

```
public class LorduiApplet extends EntryApplet {
    private ktm.lordui.Lordui lordui;
    @Override
    public void init() {
        lordui = ktm.lordui.Lordui.createInstance();
        try {
            lui.loadProject(new File("myLorduiProcedureFile.lui"));
        } catch (IOException e) {
            e.printStackTrace();
        }
        return;
    }
    super.init();
}

@Override
public void start() {
    super.start();
    lui.setVisible(true);
}

@Override
public void stop() {
```

```

lui.setVisible(false);
super.stop();
}

@Override
public void destroy() {
    lui.close();
    super.destroy();
}
}

```

Nową klasę należy opakować w plik jar i traktować jako główny plik uruchamialny appletu.

## Testy JUnit

Uruchamianie Lordui jako biblioteki mając dostęp do kodów źródłowych jest najwygodniejsze. Przykładowo możemy dodać na początku wykonanie kodu:

```

lui = Lordui.createInstance();
try {
    lui.loadProject(new File("myLorduiProcedureFile.lui"));
} catch (IOException e) {
    e.printStackTrace();
}
return;
}

```

oraz po stworzeniu okna aplikacji uruchomić procedurę Lordui, zaś na koniec bezpiecznie zwolnić zasoby:

```

lui.runProcedureAndWait("LorduiProcedureName");
lui.close();

```

Taki przykład wykorzystania Lordui znajduje się w rozdziale: 4.4.1: „Interface biblioteki Lordui”.

## 4.5 Wyrażenia

Program udostępnia wyrażenia oraz funkcje. Jako wyrażenie rozumiane jest tu działanie na obiektach. Obiektami tymi mogą być m.in. napisy, liczby, obrazki, punkty.

Zmienne, podobnie jak procedury, można grupować w kolekcje. Aby umieścić zmienną w kolekcji należy zmienić jej nazwę, dodając na jej początku


nazwę kolekcji i dwukropek jako separator (dwukropek jest w Lordui traktowany jako separator pakietów). Np. nadając zmiennej nazwę *a:b:c*, zmienna będzie się nazywała *c* i będzie umieszczona w kolekcji *b*, która z kolei będzie się znajdowała w kolekcji *a*.

Lordui udostępnia możliwość wyszukiwania użycia wyrażenia w procedurach. Aby uruchomić wyszukiwanie należy wybrać „Edycja” -> „Pokaż wywołania zmiennych”. W podanym okienku należy podać pełną nazwę zmiennej (wraz z pakietami).

#### 4.5.1 Typy wyrażeń

Program udostępnia między innymi następujące typy zmiennych:

- Listę,
- Punkt (punkt (0,0) to lewy górny róg),
- Napis,
- Liczbę,
- Obrazek,
- Pozycjonowany obrazek (czyli parę obrazek oraz punkt),
- Wartość logiczną (true/false - wielkość liter MA znaczenie),
- Obszar (punkt x, punkt y, szerokość, wysokość), gdzie punkt (0,0) to lewy-górny róg,
- Dźwięk,
- Inny obiekt.

Obiekty podanych typów można zapisywać w zmiennych globalnych, które współdzielone są ze wszystkimi procesami. Stałe można edytować przy pomocy narzędzi dostępnych na górnym pasku toolbar okna programu. Pod guzikiem  można znaleźć listę wszystkich zdefiniowanych zmiennych globalnych.

Obiekty te są także wyliczane dynamicznie z wyrażeń podczas wykonywania komend procesów. Każdy proces ma oddzielną pamięć, przez co procesy są od siebie zupełnie niezależne. Każde przypisanie zmiennej odbywa się w lokalnej pamięci procesu i nie ma wpływu na zmienne globalne (pamięć lokalna procesu ma pierwszeństwo nad zmiennymi globalnymi).







### 4.5.2 Edycja wyrażeń

Edycja wyrażeń dostępna jest przy edytowaniu komend procedur. Do poszczególnych pól można wpisywać proste wartości, ale też i skomplikowane wyrażenia. Wyrażeniami można posługiwać się także przy deklarowaniu zmiennych globalnych, jednak zostaną one wtedy zapisane jako wyniku wyrażenia. W definiowaniu wyrażeń, aby zdefiniować napis, należy użyć podwójnego cudzysłowu - napisy bez podwójnych cudzysłowów interpretowane będą jako inne obiekty (np. zmienne).



Rysunek 4.2: Przykładowe okienko edycji wyrażenia

Okno wyrażeń, zależnie od kontekstu, może zawierać m.in. następujące przyciski:

-  ,  - odpowiednio oznacza zmienną lokalną/globalną (kliknij aby przełączyć). Opcja dostępna tylko dla wyrażeń nazwy zmiennej pod którą wynik komendy ma zostać zapisany. Zmienna może zostać zapisana lokalnie (w pamięci wywołania procedury) lub do zmiennych globalnych,
-  - Podgląd obrazka - pozwala obejrzeć wybrany obrazek (pod warunkiem, że wyrażenie określa je jednoznacznie,
-  - Wybór obszaru/obrazka - kliknij aby zaznaczyć na ekranie obszar,
-  - Wybór punktu - kliknij aby wybrać punkt na monitorze,
-  - Wyświetlenie dostępnych funkcji i atrybutów zdefiniowanych.

### 4.5.3 Zmienne obiektów

Niektóre obiekty mają zdefiniowane swoje zmienne lokalne (pola). Aby odczytać takie pole, należy po wyrażeniu obiektu dopisać kropkę oraz nazwę pola. Dostępne są następujące pola:

- **function get(index)**  
Zwraca obiekt znajdujący się pod podanym indeksem w tablicy
  - **index** - Indeks obiektu w tablicy
- **function set(index, value)**  
Wstawia obiekt do tablicy. Zwraca nową tablicę (nowa i stara tablica mogą lecz nie muszą być tymi samymi obiektami)

- **index** - Indeks obiektu w tablicy
- **value** - Obiekt do umieszczenia w tablicy
- **function size()**  
Rozmiar tablicy
- **function trim()**  
Zwraca przycięty na początku i końcu napis - bez spacji, tabulatorów, znaków nowej linii
- **function replace(pattern, replace\_with)**  
Zwraca nowy napis - podmienia podany ciąg znaków
  - **pattern** - Wzorzec do wyszukania w napisie - dla pełnego opisu zobacz dokumentację klasy Pattern języka Java (\* oznacza dowolny napis)
  - **replace\_with** - Wartość jaką zastąpić wystąpienia wzorca.
- **function getPosition()**  
Zwraca punkt (pozycję) zapisany w pozycjonowanym obrazku
- **function split()**  
Rozrywa napis według podanego ciągu znaków i zwraca poszczególne składowe w tablicy - zobacz funkcję Java'ową String:split dla bardziej szczegółowego opisu
- **function pushEnd(object)**  
Dostawia obiekt na koniec tablicy. Zwraca nową tablicę
  - **object** - Obiekt do dostawienia
- **function pushBegin(object)**  
Dostawia obiekt na początek tablicy. Pozostałe obiekty zostają przesunięte o jedną pozycję w prawo. Zwraca nową tablicę
  - **object** - Obiekt do dostawienia
- **function push(object, index)**  
Dostawia obiekt do tablicy. Obiekty od podanego indeksu wzwyż są przesunięte o jedną pozycję w prawo. Zwraca nową tablicę
  - **object** - Obiekt do dostawienia
  - **index** - Indeks, na którym wstawić element
- **function toPoint()**  
Zwraca punkt zapisany w podanym obiekcie

- **function toArea()**  
Zwraca obszar zapisany w danym pozycjonowanym obrazku
- **function subImage(x, y, width, height)**  
Pozycjonowany obrazek danego pozycjonowanego obrazka. Zwracana pozycja to suma współrzędnych w głównym obrazku oraz podanych współrzędnych
  - **x** - Pozycja x obrazka względem lewej krawędzi obrazka oryginalnego
  - **y** - Pozycja y względem górnej krawędzi oryginalnego obrazka
  - **width** - Szerokość podobrazka
  - **height** - Wysokość podobrazka
- **function subImage(x, y, width, height)**  
Podobrazek danego obrazka
  - **x** - Pozycja x obrazka względem lewej krawędzi obrazka oryginalnego
  - **y** - Pozycja y względem górnej krawędzi oryginalnego obrazka
  - **width** - Szerokość podobrazka
  - **height** - Wysokość podobrazka
- **function location()**  
Pozycja (punkt - współrzędne lewego-górnego rogu) okna względem ekranu
- **function middlePoint()**  
Pozycja (punkt) - centralnego pixela okna względem lewego-górnego rogu ekranu.
- **function findUISubcomponent(functionName, value)**  
Zwraca obiekt interface'u użytkownika, będący podobieństwem podanego komponentu i zwracający odpowiednią wartość podanej funkcji
  - **functionName** - Nazwa funkcji, która będzie wywoływana an podobieństwach
  - **value** - Wartość funkcji jaką ma zwrócić poszukiwany obiekt
- **function readyForReading()**  
Zwraca wartość logiczną - prawdę, gdy z podanego gniazda (ang. Socket) można czytać dane (jeśli są jakieś komunikaty do przeczytania)

- **function mergeValues(separator)**  
Skleja wartości tablicy rzutując je jako napis oraz przedzielając podanym wzorcem.
  - **separator** - Wzorec do sklejenia elementów tablicy
- **function getColor(x, y)**  
Zwraca kolor określonego pixela na obrazku.
  - **x** - Współrzędna poprzeczna
  - **y** - Współrzędna pionowa
- **function subImage(area)**  
Zwraca wycinek oryginalnego obrazka. Wycinek obrazka będzie dzielił dane z oryginalnym obrazkiem - edycja któregokolwiek z nich będzie widoczna także na tym drugim
  - **area** - Obszar wycinka względem oryginału
- **function subImage(area)**  
Zwraca wycinek oryginalnego obrazka. Wycinek obrazka będzie dzielił dane z oryginalnym obrazkiem - edycja któregokolwiek z nich będzie widoczna także na tym drugim
  - **area** - Obszar wycinka względem oryginału
- **function copy()**  
Tworzy kopię obrazka
- **function copy()**  
Tworzy kopię obrazka

#### 4.5.4 Funkcje na wyrażeniach

Oprócz wyrażeń i pól, Lordui udostępnia także możliwość wykorzystywania funkcji. Zdefiniowane są funkcje dla poszczególnych obiektów, jak i funkcje systemowe.

##### Funkcje na obiektach

- **function get(index)**  
Zwraca obiekt znajdujący się pod podanym indeksem w tablicy
  - **index** - Indeks obiektu w tablicy
- **function set(index, value)**  
Wstawia obiekt do tablicy. Zwraca nową tablicę (nowa i stara tablica mogą lecz nie muszą być tymi samymi obiektami)



- **index** - Indeks obiektu w tablicy
- **value** - Obiekt do umieszczenia w tablicy
- **function size()**  
Rozmiar tablicy
- **function trim()**  
Zwraca przycięty na początku i końcu napis - bez spacji, tabulatorów, znaków nowej linii
- **function replace(pattern, replace\_with)**  
Zwraca nowy napis - podmienia podany ciąg znaków
  - **pattern** - Wzorzec do wyszukania w napisie - dla pełnego opisu zobacz dokumentację klasy Pattern języka Java (\* oznacza dowolny napis)
  - **replace\_with** - Wartość jaką zastąpić wystąpienia wzorca.
- **function getPosition()**  
Zwraca punkt (pozycję) zapisany w pozycjonowanym obrazku
- **function split()**  
Rozrywa napis według podanego ciągu znaków i zwraca poszczególne składowe w tablicy - zobacz funkcję Java'ową String:split dla bardziej szczegółowego opisu
- **function pushEnd(object)**  
Dostawia obiekt na koniec tablicy. Zwraca nową tablicę
  - **object** - Obiekt do dostawienia
- **function pushBegin(object)**  
Dostawia obiekt na początek tablicy. Pozostałe obiekty zostają przesunięte o jedną pozycję w prawo. Zwraca nową tablicę
  - **object** - Obiekt do dostawienia
- **function push(object, index)**  
Dostawia obiekt do tablicy. Obiekty od podanego indeksu wzwyż są przesunięte o jedną pozycję w prawo. Zwraca nową tablicę
  - **object** - Obiekt do dostawienia
  - **index** - Indeks, na którym wstawić element
- **function toPoint()**  
Zwraca punkt zapisany w podanym obiekcie

- **function toArea()**  
Zwraca obszar zapisany w danym pozycjonowanym obrazku
- **function subImage(x, y, width, height)**  
Pozycjonowany obrazek danego pozycjonowanego obrazka. Zwracana pozycja to suma współrzędnych w głównym obrazku oraz podanych współrzędnych
  - **x** - Pozycja x obrazka względem lewej krawędzi obrazka oryginalnego
  - **y** - Pozycja y względem górnej krawędzi oryginalnego obrazka
  - **width** - Szerokość podobrazka
  - **height** - Wysokość podobrazka
- **function subImage(x, y, width, height)**  
Podobrazek danego obrazka
  - **x** - Pozycja x obrazka względem lewej krawędzi obrazka oryginalnego
  - **y** - Pozycja y względem górnej krawędzi oryginalnego obrazka
  - **width** - Szerokość podobrazka
  - **height** - Wysokość podobrazka
- **function location()**  
Pozycja (punkt - współrzędne lewego-górnego rogu) okna względem ekranu
- **function middlePoint()**  
Pozycja (punkt) - centralnego pixela okna względem lewego-górnego rogu ekranu.
- **function findUISubcomponent(functionName, value)**  
Zwraca obiekt interface'u użytkownika, będący podobieństwem podanego komponentu i zwracający odpowiednią wartość podanej funkcji
  - **functionName** - Nazwa funkcji, która będzie wywoływana an podobieństwach
  - **value** - Wartość funkcji jaką ma zwrócić poszukiwany obiekt
- **function readyForReading()**  
Zwraca wartość logiczną - prawdę, gdy z podanego gniazda (ang. Socket) można czytać dane (jeśli są jakieś komunikaty do przeczytania)

- **function mergeValues(separator)**  
Skleja wartości tablicy rzutując je jako napis oraz przedzielając podanym wzorcem.
  - **separator** - Wzorec do sklejenia elementów tablicy
- **function getColor(x, y)**  
Zwraca kolor określonego pixela na obrazku.
  - **x** - Współrzędna poprzeczna
  - **y** - Współrzędna pionowa
- **function subImage(area)**  
Zwraca wycinek oryginalnego obrazka. Wycinek obrazka będzie dzielił dane z oryginalnym obrazkiem - edycja któregokolwiek z nich będzie widoczna także na tym drugim
  - **area** - Obszar wycinka względem oryginału
- **function subImage(area)**  
Zwraca wycinek oryginalnego obrazka. Wycinek obrazka będzie dzielił dane z oryginalnym obrazkiem - edycja któregokolwiek z nich będzie widoczna także na tym drugim
  - **area** - Obszar wycinka względem oryginału
- **function copy()**  
Tworzy kopię obrazka
- **function copy()**  
Tworzy kopię obrazka

## Funkcje systemowe

Aktualnie zdefiniowane są następujące funkcje systemowe:

- **static function activeWindowCoordinates()**  
Współrzędny aktywnego okna (pozycja i rozmiar). Pozycja podana jest relatywnie do górnego lewego rogu głównego monitora.
- **static function cursorPosition()**  
Pozycja kursora myszki względem górnego lewego rogu głównego monitora
- **static function getValue(index)**  
Zwraca wartość zmiennej o zadanej nazwie (to jest dynamiczna alternatywa dla odwołania się do zmiennej po nazwie)

- **index** - indeks obiektu w tablicy (licząc od zera)
- **static function createImage(width, height)**  
Tworzy nowy obiekt obrazka
  - **width** - Szerokość obrazka
  - **height** - Wysokość obrazka
- **static function captionImage(text, fontName, fontSize, fontStyle, color)**  
Rysuje text na obrazku. Tło pozostaje przezroczyste
  - **text** - Text do narysowania
  - **fontName** - Nazwa czcionki (nazwa całej rodziny czcionki, Z rozróżnieniem wielkości liter)
  - **fontSize** - Rozmiar czcionki
  - **fontStyle** - Styl czcionki - dla domyślnego stylu wpisz 0, dla pogrubionego użyj funkcji BOLD()
  - **color** - Kolor czcionki w formacie HTML-owym np.: FFFFFFFF
- **static function activeWindowTitle()**  
Tytuł aktywnego okna
- **static function screenSize()**  
Rozmiar całego obrazu (uwzględniając wszystkie wszystkie monitory i ich wzajemne położenie)
- **static function BOLD()**  
Stała zawierająca pogrubiony styl czcionki
- **static function compareImages(image1, image2)**  
Oblicza obszar na którym podane obrazki się różnią
  - **image1** - Pierwszy z obrazków do porównania
  - **image2** - Drugi z obrazków do porównania
- **static function screenArea()**  
Obszar całego obrazu uwzględniając wszystkie monitory. Współrzędne oznaczają lewy górny róg głównego monitora
- **static function concat(area1, area2)**  
Nowy obiekt zawierający część wspólną dwóch obszarów
  - **area1** - Pierwszy obszar do przecięcia
  - **area2** - Drugi obszar do przecięcia

- **static function popFirst(arrayName)**  
Zdejmuje pierwszy element tablicy przesuując wszystkie kolejne elementy o jeden w lewo. Zwracany jest zdjęty element.
  - **arrayName** - Nazwa zmiennej tablicy
- **static function popLast(arrayName)**  
Zdejmuje i zwraca ostatni element tablicy.
  - **arrayName** - Nazwa zmiennej tablicy
- **static function pop(arrayName, index)**  
Zdejmuje i zwraca podany element tablicy. Kolejne elementy są przesunięte o jeden w lewo.
  - **arrayName** - Nazwa zmiennej tablicy
  - **index** - Indeks elementu do zdjęcia
- **static function getClipboard()**  
Zwraca wartość zapisaną w schowku systemowym: obrazek, liczbę, wartość logiczną lub napis
- **static function readImage(url)**  
Wczytuje obrazek z podanego pliku lub adresu URL
  - **url** - Adres URL lub ścieżka pliku (względna lub pełna) pliku z obrazkiem
- **static function color(htmlColor)**  
Tworzy nowy obiekt koloru
  - **htmlColor** - Kolor w formacie HTML (np. 0xFFFFFFFF)
- **static function color(red, green, blue)**  
Tworzy nowy obiekt koloru
  - **red** - Czerwona wartość składowa koloru (0-255)
  - **green** - Zielona wartość składowa koloru (0-255)
  - **blue** - Niebieska wartość składowa koloru (0-255)
- **static function getColorArea(image, startPoint)**  
Oblicza obszar podanego koloru - znajduje najdalej wysunięte pixele do góry, w dół, w prawo i lewo, połączone z wybranym pixelem
  - **image** - Obrazek na którym ma zostać znaleziony obszar
  - **startPoint** - Pixel startowy

- **static function getBackgroundColor(image, area)**  
Zwraca kolor tła obrazka (kolor, którego jest najwięcej pixeli)
  - **image** - Obrazek do wykrycia koloru tła
  - **area** - Obszar na którym ma zostać wykryty kolor tła
- **static function getBackgroundColor(image)**  
Zwraca kolor tła obrazka (kolor, którego jest najwięcej pixeli)
  - **image** - Obrazek do wykrycia koloru tła
- **static function activeWindow()**  
Wskaźnik do aktywnego okna
- **static function getActiveJavaWindow()**  
Wskaźnik do aktywnego okna Java lub Null, jeśli nie ma takiego okna
- **static function callJavaStaticFunction(ClassName, Function, Args)**  
Wywołuje statyczną funkcję na podanej klasie. Zwraca zwróconą wartość lub null gdy funkcja zwraca void
  - **ClassName** - Nazwa klasy
  - **Function** - Nazwa funkcji do wywołania
  - **Args** - Tablica argumentów do użycia przy wywołaniu funkcji
- **static function callJavaFunction(object, Function, Args)**  
Wywołuje funkcję na podanym obiekcie. Zwraca zwróconą wartość lub null gdy funkcja zwraca void
  - **object** - Obiekt, na którym zostanie wywołana funkcja
  - **Function** - Nazwa funkcji do wywołania
  - **Args** - Tablica argumentów do użycia przy wywołaniu funkcji
- **static function variableExists(variableName, globalOnly)**  
Prawda, gdy istnieje zmienna o podanej nazwie
  - **variableName** - Nazwa sprawdzanej zmiennej
  - **globalOnly** - Prawda dla zmiennej globalnej (wyłącznie), fałsz dla zmiennej globalnej lub lokalnej dla aktualnego wywołania procedury
- **static function createScreenshot(area)**  
Wykonuje zdjęcie obrazu (tzw. screenshot)
  - **area** - Obszar zdjęcia obrazu

- **static function getWindowWithTitlePrefix(titlePrefix)**  
Zwraca wskaźnik do jednego z okien (dowolnej aplikacji), którego tytuł rozpoczyna się od podanego napisu
  - **titlePrefix** - Prefiks tytułu szukanego okna
- **static function getSystemProperty(propertyName)**  
Wartość zmiennej systemowej. Patrz wywołanie `Java System.getProperty(String)`
  - **propertyName** - Nazwa zmiennej systemowej do pobrania
- **static function isImage(object)**  
Zwraca `true` dla obiektów typu `image`
  - **object** - Obiekt do sprawdzenia, czy jest typu `Image`
- **static function abs(value)**  
Oblicza obszar podanego koloru - znajduje najdalej wysunięte pixele do góry, w dół, w prawo i lewo, połączone z wybranym pixelem
  - **value** - Obrazek na którym ma zostać znaleziony obszar

### Definiowanie własnych funkcji

Aby zdefiniować własną funkcję, w oknie ustawieniach procedury należy zaznaczyć pole „Funkcja”. Następnie zalecane jest także wybranie typu zwracanego obiektu. Zwracany obiekt jest ustalany podczas wywołania komendy „Return”. Własne funkcje mogą być używane w wyrażeniach zupełnie analogicznie do funkcji systemowych.

#### 4.5.5 Przykłady wyrażeń


Poniżej znajduje się kilka przykładów wyrażeń Lordui:


1. `false || true` - wylicza się do `true` - obiekt typu wartości logicznej,
2. `1 <= 2 && 2 != 3` - wylicza się do `true` - obiekt typu wartości logicznej,
3. `"Hello " + "world!!!"` - wylicza się do `"Hello world!!!"` - obiekt typu napis,
4. `7 + 8` - wylicza się do `15` - obiekt typu liczba,
5. `"a,b,c,d,e".split(", ").get(1)` - wylicza się do `"b"` - obiekt typu napis,
6. `screenSize().x + nazwaZmiennej` - zakładając, że pod `nazwaZmiennej` przypisana jest liczba - obiekt typu liczba,

## 4.6 Komendy procedur

Procedury, podstawowe elementy programu, są zbudowane z komend. Komendy stanowią podstawowe narzędzie programisty Lordui. Poniższy rozdział opisuje komendy dostępne w programie. Komendy można dodawać na kilka sposobów:

- klikając prawym przyciskiem myszy na procedurze lub komendzie oraz wybierając odpowiednią opcję (dodaj krok przed/po/do),
- edytując w notatniku w formacie XML, a następnie wklejając je do procedury,
- za pomocą lewego menu.

Większość komend ma dostępne parametry, które można edytować zaznaczając je w drzewie procedur lub otwierając w nowym oknie przy pomocy przycisku . Komendy można także usuwać, kopiować, przenosić - przy pomocy menu dostępnego pod prawym przyciskiem myszy oraz popularnych skrótów klawiszowych *ctrl+x*, *ctrl+c*, *ctrl+v*. Komendy przenoszone są w formacie XML.

Aby usunąć komendę z procedury, należy ją wskazać w drzewie procedur i nacisnąć przycisk  lub klawisz klawiatury *Delete*.

### 4.6.1 Strzałki

#### Ukrycie strzałki

Ukrycie strzałki zapisanej pod podaną zmienną.

Parametry:

- **Nazwa zmiennej strzałki** - Nazwa zmiennej - Nazwa zmiennej strzałkowej.

#### Pokazanie/przesunięcie strzałki

Pokazanie strzałki, która ma pokazywać konkretny punkt ekranu. Jeśli do podanej zmiennej jest już przypisana strzałka, zostanie ona przesunięta.

Parametry:

- **Nazwa zmiennej strzałki** - Nazwa zmiennej - Nazwa zmiennej strzałki. Zmienna ta zostanie zmodyfikowana lub utworzona,
- **Punkt wskazywany przez strzałkę** - Wyrażenie punktu - Pozycja na ekranie, która ma być wskazywana przez strzałkę.



### 4.6.2 Ikonki paska start

Ikonki paska start (tzw. Tray Icons) to ikonki na pasku start, położone obok zegarka. LordUI umożliwia pokazanie własnych ikonek oraz powiadomień ikonki. Dostępne są następujące operacje na ikonkach paska start.

#### Nowy element menu ikonki paska start

Stworzenie nowego elementu menu dostępnego po kliknięciu na ikonke paska start.

Parametry:

- **Nazwa zmiennej** - Nazwa zmiennej - Nazwa zmiennej z ikonką paska start,
- **Napis pozycji** - Wyrażenie napisowe - Napis do pokazania użytkownikowi,
- **Nazwa odtwarzacza** - Wyrażenie napisowe - Nazwa odtwarzacza, na którym zostanie uruchomiona procedura.

Podprocedury:

- **itemAction** - Akcja wykonana przy wybraniu pozycji.

#### Zmień obrazek ikonki paska start

Zmienia wyświetlaną ikonkę.

Parametry:

- **Nazwa zmiennej** - Nazwa zmiennej - Nazwa zmiennej ikonki paska start,
- **Wyrażenie ikonki** - Wyrażenie napisowe - Obrazek do wyświetlenia na ikonke paska start (zostanie przeskalowane, aby dostosować do rozmiaru ikonki).

#### Ukryj ikonkę paska start

Ukrywa wybraną ikonkę paska start.

Parametry:

- **Nazwa zmiennej** - Nazwa zmiennej - Nazwa zmiennej ikonki, która ma zostać ukryta.

### **Pokaż ikonkę paska start**

Pokazuje ikonkę na pasku start.

Parametry:

- **Nazwa zmiennej** - Nazwa zmiennej - Zmienna pod jaką zostanie zachowana ikonka paska start,
- **Wyrażenie ikonki** - Wyrażenie obrazka - Obrazek do umieszczenia na ikonke. Obrazek zostanie przeskalowany aby dostosować do rozmiaru ikonki.

### **Wiadomość ikonki paska start**

Pokazuje wiadomość w formie podpowiedź przy ikonke paska start.

Parametry:

- **Nazwa zmiennej** - Nazwa zmiennej - Nazwa zmiennej ikonki paska start, przy której ma zostać pokazana podpowiedź,
- **Nagłówek wiadomości** - Wyrażenie napisowe - Nagłówek wiadomości,
- **Treść wiadomości** - Wyrażenie napisowe - Treść wiadomości.

### **4.6.3 Kliknięcie użytkownika**

Blokada ekranu realizowana jest przy pomocy przezroczystego okna, które wyświetlone zostanie na całym ekranie. Blokadę tą można wyłączyć w dowolnym momencie naciskając *Escape*, o czym wyświetlana jest odpowiednia informacja.

### **Prośba użytkownika o kliknięcie**

Komendę należy wywoływać przed pokazaniem blokady, aby uniknąć zbędnego migania prostokątami na ekranie. Komenda dodaje obszar na blokadzie, w który użytkownik powinien kliknąć myszką. Akcję wykonywaną po kliknięciu należy zaimplementować w podprocedurze komendy.

Parametry:

- **Obszar kliknięcia użytkownika** - Wyrażenie obszaru - Obszar, który zostanie podświetlony na ekranie. Użytkownik będzie mógł go wskazać kliknięciem myszy,
- **Nazwa blokady** - Wyrażenie napisowe - Nazwa zmiennej z blokadą na której dodać obszar do kliknięcia,

- **Nazwa odtwarzacza** - Wyrażenie napisowe - Nazwa odtwarzacza, na którym uruchomić procedurę przy kliknięciu użytkownika.

Podprocedury:

- **OnClick** - Procedura uruchomiona po wskazaniu obszaru.

### Usunięcie akcji użytkownika

Usunięcie wszystkich aktualnie zdefiniowanych „*Prośb użytkownika o kliknięcie*”. Po wywołaniu tej komendy blokada (niezależnie czy jest pokazana czy nie) powinna być pusta. Należy więc zdefiniować po niej nowe obszary kliknięcia.

Parametry:

- **Nazwa blokady** - Wyrażenie napisowe - Nazwa blokady z której należy usunąć obszary do kliknięcia.

### Obsługa blokady ekranu

Włączenie/wyłączenie blokady ekranu.

Parametry:

- **Włącz/wyłącz** - Pole wyboru - Czy blokada ma zostać włączona czy wyłączona,
- **Nazwa blokady** - Wyrażenie napisowe - Nazwa zmiennej z blokadą.

## 4.6.4 GUI

GUI (ang. Graphical User Interface - graficzny interfejs użytkownika). Wszystkie operacje użytkownika tj. operacje na oknach, wyskakujących komunikatach, ikonkach itp.

### Aktywacja okna

Aktywuje podane okno. Gdy takie okno nie istnieje, nie wykona się żadna operacja.

Parametry:

- **Okno do aktywowania** - Wyrażenie obiektu - Okno, które ma zostać aktywowane.

### Wybór guzika

Zostanie pokazana krótka wiadomość tekstowa oraz zdefiniowane przyciski. Każdemu przyciskowi należy zaimplementować odpowiadającą mu procedurę.

Parametry:

- **Pokazany tekst** - Wyrażenie napisowe - Tekst do pokazania,
- **Nazwa zmiennej wyniku** - Nazwa zmiennej - Nazwa zmiennej pod jaką zostanie zachowany wynik wskazany przez użytkownika.

### Wybór pliku

Zostanie pokazane okno wyboru pliku. Nazwa pliku zostanie zapisana do podanej zmiennej.

Parametry:

- **Wyrażenie** - Nazwa zmiennej - Nazwa zmiennej pod jaką ma zostać zapisana nazwa pliku.

### Zmień rozmiar okna

Przesunięcie oraz zmiana rozmiaru okna

Parametry:

- **Okno** - Wyrażenie obiektu - Wskaźnik do okna, które ma zostać przesunięte,
- **Obszar** - Wyrażenie obszaru - Obszar, który okno ma zajmować po przesunięciu.

### Wiadomość tekstowa

Krótką wiadomość tekstową.

Parametry:

- **Wiadomość** - Wyrażenie napisowe - Tekst do pokazania.

### 4.6.5 Rysowanie

Poniższe komendy służą do rysowania prostych kształtów na obrazkach

### Rysuj obrazek na obrazku

Rysuje obrazek na obrazku.

Parametry:

- **Obraz do narysowania** - Wyrażenie obrazka - Obrazek który zostanie narysowany,
- **Obszar** - Wyrażenie obszaru - Obszar obrazka docelowego, gdzie zostanie narysowany obrazek,
- **Obraz** - Wyrażenie obrazka - Obrazek na którym zostanie narysowany drugi obrazek.

### Zamaluj prostokąt

Maluje prostokąt.

Parametry:

- **Kolor** - Wyrażenie koloru - kolor jaki ma zostać użyty do zamalowania pixela (aby uzyskać wartość koloru można użyć funkcji `color(r,g,b)`),
- **Współrzędne obszaru** - Wyrażenie punktu - współrzędne obszaru do zamalowania,
- **Nazwa obrazka** - Nazwa zmiennej - zmienna obrazka, na którym ma zostać namalowany prostokąt,
- **Wypełnij** - Wyrażenie logiczne - czy wypełnić obszar kolorem (w przeciwnym przypadku zostanie narysowana jedynie ramka).

### Ustaw kolor pixela

Zamalowuje wybrany pixel wybranym kolorem.

Parametry:

- **Współrzędne pixela** - Wyrażenie punktu - współrzędne pixela,
- **Kolor** - Wyrażenie koloru - kolor jaki ma zostać użyty do zamalowania pixela (aby uzyskać wartość koloru można użyć funkcji `color(r,g,b)`),
- **Nazwa obrazka** - Nazwa zmiennej - zmienna obrazka, na którym ma zostać zamalowany pixel.

## 4.6.6 Obsługa obrazu

### Kliknięcie obrazka

To jest skrót do operacji wyszukiwania obrazka na ekranie. Podczas tworzenia tego elementu programista zostanie poproszony o zaznaczenie na ekranie obrazka do wyszukania oraz podania jego nazwy. Obrazek zostanie zapisany jako wzorzec do wyszukania na ekranie. Jeśli będzie zaznaczona opcja „Dodaj komunikat błędu gdy obrazek nie zostanie znaleziony”, do podprocedury „notFound” zostanie wstawiona dodatkowo komenda pokazywania informacji o błędzie. Domyślnie gdy obrazek zostanie znaleziony, zostanie wykonane kliknięcie w centralnym jego punkcie, zaś gdy nie zostanie znaleziony, zostanie zatrzymane wykonywanie wszystkich procedur.

### Szukanie obrazu

Komenda wyszuka podany wzorzec. Przy pomocy tej komendy można wyszukać obrazek na ekranie lub na zmiennej obrazkowej i podjąć związaną z tym akcję (np. poczekać)

Parametry:

- **Wyrażenie obrazka wzorca** - Wyrażenie obrazka - Obrazek, który zostanie wyszukany,
- **Inwersja działania** - Pole wyboru - Inwersja powoduje, że komenda będzie czekała aż obrazek zniknie z ekranu. Bez włączonej inwersji komenda czeka aż obrazek się pojawi na ekranie,
- **Znajdź wszystkie** - Pole wyboru - Włącz by komenda zwracała tablicę wszystkich wystąpień wzoru na obrazie. Gdy opcja jest wyłączona zostanie zwrócona tylko jedna współrzędna obrazka,
- **Nazwa zmiennej pod wynikowej** - Nazwa zmiennej - nazwa zmiennej pod jakim zostanie zapisana współrzędna wyniku. Zwracane jest położenie lewego górnego rogu wzorca. Przypisanie zostaje wykonane każdorazowo za każdym razem zaraz po znalezieniu wzorca. Pozostaw puste, by nie zwracać żadnej wartości. Jeśli po zakończeniu wyszukiwania wzorzec nie został znaleziony, przypisanie nie zostanie wykonane. Jeśli opcja „Znajdź wszystkie” jest zaznaczona, zostanie zwrócona tablica współrzędnych,
- **Poziom porównywania obrazków** - Suwak - Jak bardzo wzorzec powinien się pokrywać z obrazkiem (pozostaw zero dla identyczności),
- **Ile milisekund czekać, aż obrazek się pojawi** - Wyrażenie liczbowe - Ile milisekund czekać gdy obrazek nie zostaje znaleziony,

- **Obszar/punkt poszukiwań wzorca** - Wyrażenie punktu/obszaru - Miejsce gdzie obszar ma zostać poszukiwany (obszar lub współrzędne lewego górnego rogu). Pozostawić puste, jeśli pełny ekran/cały obrazek ma zostać przeszukany,
- **Źródło obrazka** - Wyrażenie obrazka - Obrazek, na którym ma zostać wyszukany wzorec. Pozostawić puste, jeśli użyty ma być obraz monitora(ów).

Podprocedury:

- **FoundProcedure** - zostanie wywołana każdorazowo przy znalezieniu wzorca (tuż przed opuszczeniem komendy). Zostanie wywołana gdy obrazek zostanie znaleziony (gdy inwersja jest wyłączona) lub wystąpi tzw. timeout (gdy inwersja jest włączona),
- **NotFoundProcedure** - zostanie wywołana każdorazowo, gdy wzorec nie zostanie znaleziony (np. wystąpi tzw. timeout). Zostanie wywołana gdy obrazek nie zostanie znaleziony (gdy inwersja jest wyłączona) lub wystąpi timeout (gdy inwersja jest włączona),
- **OnSearching** - zostanie uruchomiona jeśli komenda nie znalazła obrazka (lub znalazła obrazek jeśli inwersja jest włączona). Jeśli wszystkie warunki są spełnione, pozycja znalezionej obrazka jest przypisana do podanej zmiennej.

Każdorazowo po szukaniu wzorca na ekranie, wywołana jest dokładnie jedna z powyższych trzech procedur.

### Zapisanie obrazu do pliku

Zapisanie obrazku z pamięci do pliku

Parametry:

- **Wybór nazwy pliku** - Opcje - dostępne są trzy możliwości:
  1. Nazwa pliku podana dokładnie - obrazek zostanie zapisany pod podaną nazwą także wtedy, gdy plik o tej nazwie już istnieje.
  2. Zapytaj się o nazwę pliku - Użytkownik zostanie zapytany o nazwę pliku podczas odtwarzania elementu,
  3. Dołącz suffix czyniąc nazwę pliku oryginalną - obrazek zostanie zapisany pod podaną nazwą, jednak do nazwy pliku zostanie dołączony suffix, by ucznić nazwę pliku unikalną
- **Domyślna nazwa pliku** - Wyrażenie napisowe - domyślna nazwa pliku lub nazwa pliku docelowego pod jakim zostanie zapisany obrazek,

- **Wyrażenie obrazka** - Wyrażenie obrazka - obrazek, który ma zostać zapisany po podanym plikiem.

### Zdjęcie obrazu

Zostanie wykonane zdjęcie obrazu (tzw. Screenshot).

Parametry:

- **Wybór nazwy pliku** - Opcje - dostępne są trzy możliwości:
  1. Nazwa pliku podana dokładnie - obrazek zostanie zapisany pod podaną nazwą także wtedy, gdy plik o tej nazwie już istnieje.
  2. Zapytaj się o nazwę pliku - Użytkownik zostanie zapytany o nazwę pliku podczas odtwarzania elementu,
  3. Dołącz sufix czyniąc nazwę pliku oryginalną - obrazek zostanie zapisany pod podaną nazwą, jednak do nazwy pliku zostanie dołączony sufix, by uczynić nazwę pliku unikalną
- **Nazwa zmiennej, pod jaką zapisane zostanie zdjęcie** - Wyrażenie napisowe - Nazwa zmiennej pod jaką zostanie zapisane zdjęcie obrazu,
- **Wyrażenie obrazka** - Wyrażenie obszaru - obszar, który ma zostać zapisany do pamięci. Jeśli niewypełnione, zostanie użyty cały ekran. Uwaga - lewy górny róg monitora nie zawsze ma współrzędne (0, 0) - zależnie od ustawień systemu operacyjnego.

### Szukanie napisu

Zostanie wyszukany na ekranie napis. Należy wybrać czcionkę, kolor, tekst napisu oraz nazwę zmiennej do zapisania wyniku. Napis wyszukiwany jest w formie obrazka - komenda wyszukuje zbiór pixeli, która układa się na napis. Jeśli na ekranie znajduje się np. duża plama o podanym kolorze, może ona zostać uznana za napis. Wyszukiwanie napisu nie będzie działało, przy jakimkolwiek rozmyciu obrazu - patrz Microsoft ClearType.

Parametry:

- **Szukany tekst** - Wyrażenie napisowe - Tekst do wyszukania,
- **Nazwa czcionki** - Wyrażenie napisowe - Nazwa czcionki szukanego tekstu,
- **Rozmiar czcionki** - Wyrażenie liczbowe - Rozmiar czcionki wykrywanego tekstu,



- **Styl czcionki** - Wyrażenie liczbowe - 0 dla domyślnego stylu lub funkcja BOLD() dla tekstu pogrubionego,
- **Nazwa zmiennej wyniku** - Nazwa zmiennej - Nazwa zmiennej pod którą ma zostać zapisana współrzędna znalezionego tekstu,
- **Kolor w formacie htmlowym** - Wyrażenie napisowe - Kolor w formacie HTMLowym. Rozpoznawany tekst musi być w całości zapisany podanym kolorem..

### **Pokazanie obrazu**

Pokaże obrazek w wyskakującym okienku.

Parametry:

- **Wyrażenie obrazka** - Wyrażenie obrazka - Obrazek do pokazania użytkownikowi.

### **OCR - Rozpoznanie napisu**

Należy podać obszar poszukiwań, czcionkę, kolor oraz zmienną, pod którą ma zostać zapisany wynik. W zadanym obszarze nie może znajdować się żaden szum (pixel o zadanym kolorze nie będący napisem), gdyż zaimplementowany OCR nie posiada żadnych metod heurystycznych. Dzięki temu jednak przy spełnieniu wszystkich warunków, wykazuje się on 100Parametry:

- **Obszar poszukiwań** - Wyrażenie obszaru - Obszar, na którym zostanie dokonanie odczytania tekstu. Może na nim się znajdować tylko pojedyncza linia tekstu. Tekst nie może wystawać poza ten obszar,
- **Nazwa czcionki** - Wyrażenie napisowe - Nazwa czcionki wykrywanego tekstu,
- **Rozmiar czcionki** - Wyrażenie liczbowe - Rozmiar czcionki wykrywanego tekstu,
- **Styl czcionki** - Wyrażenie liczbowe - 0 dla domyślnego stylu lub funkcja BOLD() dla tekstu pogrubionego,
- **Nazwa zmiennej wyniku** - Nazwa zmiennej - Nazwa zmiennej pod którą ma zostać zapisany wykryty tekst,
- **Kolor w formacie htmlowym** - Wyrażenie napisowe - Kolor w formacie HTMLowym. Rozpoznawany tekst musi być w całości zapisany podanym kolorem. Dodatkowo na pokazanym obszarze nie może się znajdować żaden dodatkowy pixel o zadanym kolorze.

### Użytkownik oznacza obrazek

Pokazanie prośby o zaznaczenie obszaru na ekranie. Jeśli użytkownik zrezygnuje i nie oznaczy obrazu zostanie wywołana podprocedura „User cancelled”

Parametry:

- **Nazwa zmiennej** - Nazwa zmiennej - Nazwa zmiennej pod jaką zostanie zapamiętany obrazek.

Podprocedury:

- **OnCancel** - Uruchomiona gdy użytkownik zrezygnuje z wyboru.

### 4.6.7 Urządzenia wejścia (operacje myszy i klawiatury)

#### Nasłuchiwanie klawiatury

Zostanie rozpoczęte nasłuchiwanie klawiatury. Każde wciśnięcie przycisku klawiatury będzie skutkowało uruchomieniem podprocedury na wskazanym odtwarzaczu. Dostępny jest filtr przycisków (wyrażenie logiczne) - do odtwarzacza zostaną dodane tylko te wciśnięcia przycisku, które spełniają warunki filtru. Uruchomiony odtwarzacz będzie aktywny aż nasłuchiwanie nie zostanie przerwane (można też go zatrzymać np. za pomocą komend - tj komenda „Zatrzymaj nasłuchiwanie”)

Parametry:

- **Filtr** - Wyrażenie logiczne - wyrażenie logiczne określające czy dla danego wciśnięcia przycisku klawiatury ma zostać uruchomiona podprocedura,
- **Nazwa zmiennej przycisku** - Nazwa zmiennej - Pod tą zmienną zostanie zapisany kod przycisku. Kodu tego można używać także już w filtrze,
- **Nazwa odtwarzacza** - Wyrażenie napisowe - Nazwa odtwarzacza na którym mają zostać uruchamiane wątki,
- **Akcja klawiatury** - Opcje - Możesz nasłuchiwać jednej z dwóch akcji: wciśnięcia lub zwolnienia klawisza klawiatury,
- **Stempel czasowy** - Nazwa zmiennej - Zmienna będzie zawierała stempel czasowy - ilość milisekund od 1’go stycznia 1970.

Podprocedury:

- **KeyPressAction** - Akcja uruchamiana na zdarzenie wciśnięcia przycisku klawiatury.

## Zdarzenie klawiatury

Zależnie od wybranej opcji, zostanie wykonane wciśnięcie klawisza, puszczenie klawisza lub kliknięcie klawisza (czyli wciśnięcie oraz puszczenie). Poniżej należy podać w postaci liczby kod klawisza klawiatury. Kod ten można poznać wciskając ten klawisz. Jego kod pojawi się na pasku menu. Musi być wtedy jednak zaznaczona opcja: „*Opcje*” -> „*Pokazuj wartości urządzeń wskazujących*”

Parametry:

- **Operacja przycisku** - Opcje - dostępne są trzy operacje przycisku klawiatury:
  1. Wciśnięcie klawisza klawiatury
  2. Kliknięcie klawisza klawiatury (wciśnięcie + puszczenie)
  3. Puszczenie klawisza klawiatury,
  - **Kod przycisku** - Liczba - tzw. Keycode klawisza do wciśnięcia.

## Zdarzenie myszy

Zależnie od wybranej opcji można zlecić kliknięcie, przeciągnięcie, wciśnięcie lub puszczenie przycisku myszy. Należy także współrzędne kursora myszy oraz przycisk myszy (nie dotyczy przesunięcia).

Parametry:

- **Pozycja kursora** - Wyrażenie punktu - Współrzędne docelowe kursora myszy,
- **Przycisk myszy** - Opcje - Dostępne są trzy przyciski myszy:
  1. Lewy przycisk myszy
  2. Środkowy przycisk myszy
  3. Prawy przycisk myszy,
  - **Akcja guzika** - Opcje - Dostępne są następujące operacje guzika myszy:
    1. Wciśnięcie guzika
    2. Kliknięcie (wciśnięcie + puszczenie)
    3. Puszczenie guzika myszy
    4. Przesunięcie kursora myszy

## 5. Przeciągnięcie kursora myszy

- **Rodzaj ruchu kursora** - Opcje - Sposób przesuwania kursora myszki do pozycji docelowej. Dostępne są:

1. Skok myszy (kursor natychmiast znajdzie się w pozycji docelowej)
2. Ruch stały myszy (można podać prędkość kursora - wyrażoną w ilości pixeli na 1/100 sekundy)

## Nasłuchiwanie myszy

Zostanie rozpoczęte nasłuchiwanie myszy. Każde wciśnięcie przycisku myszy wywoła uruchomienie podprocedury na wskazanym odtwarzaczu. Dostępny jest filtr (wyrażenie logiczne) - do odtwarzacza zostaną dodane tylko te kliknięcia, które spełniają warunki filtru. Uruchomiony odtwarzacz będzie aktywny aż nasłuchiwanie nie zostanie przerwane (można też go zatrzymać np. za pomocą komend - tj komenda „Zatrzymaj nasłuchiwanie”)

Parametry:

- **Filtr** - Wyrażenie logiczne - Wyrażenie logiczne określające czy dla danego kliknięcia ma zostać uruchomiona podprocedura,
- **Nazwa zmiennej numeru przycisku** - Nazwa zmiennej - Pod tą zmienną zostanie zapisany numer przycisku myszy. Kodu tego można używać także już w filtrze. Zmienna może przyjąć następujące wartości:
  1. 3001 - lewy przycisk mysz
  2. 3002 - środkowy przycisk myszy (lub kliknięcie rolką myszy)
  3. 3003 - prawy przycisk myszy
  4. 3004 - brak wskazanego przycisku myszy
- **Nazwa zmiennej pozycji** - Nazwa zmiennej - Pod tą zmienną zostanie współrzędny wskazanego punktu. Kodu tego można używać także już w filtrze,
- **Nazwa odtwarzacza** - Wyrażenie napisowe - Nazwa odtwarzacza na którym mają zostać uruchamiane wątki,
- **Akcja kursora myszy** - Opcje - Możesz nasłuchiwać jednej z trzech akcji kursora myszy: wciśnięcie przycisku myszy, zwolnienie przycisku lub przesunięcie/przeciągnięcie kursora,

- **Stempel czasowy** - Nazwa zmiennej - Zmienna będzie zawierała stempel czasowy - ilość milisekund od 1'go stycznia 1970.

Podprocedury:

- **ClickAction** - Akcja uruchomiona po kliknięciu myszy (o ile spełnia podany filtr).

### Wskaż kursorem

Kursor myszki wskaże wybrany punkt ekranu poprzez zakreślanie okręgów wokół niego.

Parametry:

- **Wskazany punkt** - Wyrażenie punktu - Kursor będzie się poruszał wokół tego punktu,
- **Promień okręgu** - Wyrażenie napisowe - Promień z jakim będzie się poruszał kursor,
- **Długość wskazywania** - Wyrażenie liczbowe - Czas (w milisekundach), jaki będzie wskazywany wybrany punkt.

### Wpisanie tekstu

Wpisanie tekstu przy pomocy klawiatury.

Parametry:

- **Tekst, który ma zostać wpisany** - Wyrażenie napisowe - Podany tekst zostanie wpisany automatycznie,
- **Czas spania po wciśnięciu/puszczeniu klawisza** - Wyrażenie liczbowe - ile milisekund ma minąć każdorazowo po każdym emulowanym wciśnięciu/puszczeniu klawisza klawiatury.

### Czekanie na click myszy

Czekanie aż zostanie wykonane kliknięcie myszką

## 4.6.8 Baza danych

Za pomocą połączeń bazodanowych można połączyć się z bazą danych oraz wykonywać na niej operacje. Aby skorzystać z tej funkcjonalności należy dodać do bibliotek sterowniki Java określonej bazy.

## Połączenie do Bazy Danych

Nawiązanie połączenia do bazy danych. Aby móc się połączyć do bazy danych należy do programu dołączyć odpowiedni sterownik bazy danych. Wszelkie niestandardowe ustawienia połączenia należy ustawić używając właściwości połączenia. Wartości domyślne podane są dla sterownika bazy danych Postgres. Przykładowe wartości kluczy (jak np. "user" czy "password") mogą różnić się w zależności od użytej bazy danych.

Parametry:

- **Nazwa zmiennej wyniku** - Wyrażenie napisowe - Nazwa zmiennej pod jaką zostanie zapisane połączenie,
- **Sterownik** - Wyrażenie napisowe - Nazwa sterownika (pełna ścieżka do klasy sterownika),
- **Protokół** - Wyrażenie napisowe - Protokół komunikacji z bazą danych,
- **Adres serwera** - Wyrażenie napisowe - Adres URL serwera,
- **Nazwa bazy danych** - Wyrażenie napisowe - Nazwa bazy danych.

## Rozłączenie Bazy Danych

Zamknięcie połączenia do bazy danych.

Parametry:

- **Nazwa zmiennej połączenia** - Wyrażenie obiektu - Nazwa zmiennej, pod którą znajduje się połączenie do bazy danych.

## Operacja bazodanowa

Wykonanie operacji bazodanowej (np. takich komend jak select, update, truncate czy drop).

Parametry:

- **Połączenie bazodanowe** - Wyrażenie obiektu - Połączenia, które ma zostać użyte do wykonania operacji,
- **Operacja (komenda)** - Wyrażenie napisowe - Pełna komenda do wykonania na bazie danych,
- **Nazwa zmiennej wynikowej** - Wyrażenie napisowe - Nazwa zmiennej, pod którą ma zostać zapisany wynik operacji - dwuwymiarowa tablica (jednowymiarowa tablica jednowymiarowych tablic).

#### 4.6.9 Wsparcie obiektów Java

Dzięki wsparciu obiektów Java można lepiej kontrolować aplikacje Java. Wymagane jest jednak załączenie Lordui jako biblioteki aplikacji (lub przeciwnie - aplikacji jako biblioteki Lordui).

#### Znalezienie obiektu UI Java

Na podanym obiekcie nadrzędnym interface'u użytkownika zostanie wyszukany element interface'u. Obiekt ten zostanie wyszukany zgodnie z zapisanym wzorcem - przechodząc warstwa po warstwie.

Parametry:

- **Nazwa zmiennej wyniku** - Wyrażenie napisowe - Nazwa zmiennej pod jaką ma zostać zapisany znaleziony obiekt,
- **Nadrzędny obiekt UI** - Wyrażenie obiektu - Wskaźnik do obiektu Java (Swing) - obiektu UI na którym ma zostać wykonane wyszukiwanie.

#### 4.6.10 Komenda SO

Wywołania komend systemu operacyjnego tj. kopiowanie pliku

##### Kopiuje plik(i)

Kopiuje wybrany plik lub katalog (rekurencyjnie)

Parametry:

- **Ścieżka pliku źródłowego** - Wyrażenie napisowe - Nazwa pliku/katalogu źródłowego,
- **Ścieżka pliku docelowego** - Wyrażenie napisowe - Nazwa pliku/katalogu docelowego.

##### Usuń pliki(i)

Usunięcie pliku lub katalogu.

Parametry:

- **Ścieżka pliku** - Wyrażenie napisowe - Nazwa (ścieżka) pliku/katalogu do usunięcia.

### Otworzenie pliku dokumentu

Otwarcie pliku zawierającego dokument bądź inny materiał plikowy. Plik zostanie otwarty przez domyślną przeglądarkę lub przez program przypisany domyślnie rozszerzeniu podanego pliku

Parametry:

- **URI pliku** - Wyrażenie napisowe - Ścieżka pliku (lokalnego bądź zdalnego).

### Uruchom komendę SO

Uruchamia komendę systemu operacyjnego (tak jak z konsoli systemowej). Aby uruchomić komendę jak poprzez narzędzie "Uruchom jako"(ang. 'Run as') należy wprowadzić (na przykładzie kalkulatora) [cmd.exe, "/C", śtart", calc"]

Parametry:

- **Komenda do wykonania** - Wyrażenie tablicowe - Tablica komendy do uruchomienia. Pierwszym elementem powinna być komenda, natomiast kolejnymi argumenty. Np. aby uruchomić komendę „Java – version” należy podać jako argument: [”java”, ”-version”],
- **Uruchom w tle** - Pole wyboru - Czy element powinien czekać na zakończenie komendy.

### Pisanie do pliku

Zapisanie napisu do pliku

Parametry:

- **Napis do zapisania** - Wyrażenie napisowe - Napis, który ma zostać zapisany do pliku.,
- **Ścieżka pliku** - Wyrażenie napisowe - Ścieżka pliku.,
- **Czyść przed zapisem** - Wyrażenie logiczne - Czy plik ma zostać wyczyszczony przed wykonaniem zapisu (w przeciwnym przypadku napis zostanie doklejony na końcu)..

#### 4.6.11 Socket

Operacje na gniazdach (ang. Socket)



### **Zamknij połączenie do gniazda**

Zamknięcie połączenia do gniazda (ang. Socket). Zmienna połączenia nie będzie odtąd już dostępna.

Parametry:

- **Gniazdo (socket)** - Wyrażenie napisowe - Połączenie do zamknięcia.

### **Otwórz gniazdo**

Otwiera gniazdo (ang. Socket) to wykonywania operacji czytania i pisania.

Parametry:

- **Host** - Wyrażenie napisowe - Host (IP lub nazwa sieciowa hosta),
- **Port** - Wyrażenie napisowe - Port połączenia,
- **Zmienna docelowa** - Nazwa zmiennej - Zmienna pod którą zostanie zapisane połączenie..

### **Wyślij wiadomość na port**

Wysyła wiadomość na podany port odbiorcy. Jeśli wysłanie wiadomości się nie powiedzie, zostanie rzucony wyjątek.

Parametry:

- **Gniazdo (socket)** - Wyrażenie napisowe - Nazwa połączenia, to którego będzie wysłana wiadomość.,
- **Wiadomość** - Wyrażenie napisowe - Wiadomość do wysłania.

### **Czytaj z gniazda**

Czytanie wiadomości z gniazda (ang. Socket). Zostanie pobrana wiadomość (w formacie tekstowym) i zapisana pod podaną zmienną

Parametry:

- **Gniazdo (socket)** - Wyrażenie napisowe - Połączenie, z którego zostanie przeczytana wiadomość,
- **Zmienna docelowa** - Nazwa zmiennej - Zmienna pod którą zostanie zapisana otrzymana wiadomość.

Podprocedury:

- **Operacja na upłynięcie czasu** - Procedura uruchomiona po upływie limitu czasu przydzielonego na odczytanie wiadomości.

#### 4.6.12 Integracja

Zestawa narzędzi do komunikowania się z systemem operacyjnym/bazami danych/serwerami/aplikacjami/...

##### Kod Java

Kod Java to zdefiniowany przez użytkownika kod Java. Jest on podany jako przeciążona klasa. Dlatego też każdy taki element musi być oparty o poniższy szkielet:

```
import ktm.lordui.nativeOperations.dynamicJavaCode.DynamicClassPrototype;
import ktm.lordui.data.Memory;

public class LorduiDynamicClass extends DynamicClassPrototype {
    public void run(Memory state) {
    }
}
```

Aby móc skompilować kod, należy mieć zainstalowane Java JDK 1.6 lub nowsze. Należy także mieć poprawnie ustawione ścieżki systemowe m.in. do zmiennej Path (wskazującej na katalog z JDK).

Parametry:

- **Kod Java** - Napis - Kod w Języku Java.

##### Odczyt schowka systemowego

Przez schowek systemowy rozmiany jest tutaj bufor systemu operacyjnego, czyli fragment pamięci gdzie np. trafiają dane po wciśnięciu *ctrl+c* w wielu programach. Wartość przechowywana w pamięci operacyjnej systemu operacyjnego zostanie zapisana pod podaną zmienną.

Parametry:

- **Nazwa zmiennej wyniku** - Nazwa zmiennej - Nazwa zmiennej pod którą zostanie zapisana zmienna.

##### Zapis do schowka systemowego

Zapisuje wartość do schowka systemowego (tak, jakby została wywołana operacja *ctrl+c*).

Parametry:

- **Wartość do zapisania** - Wyrażenie obiektu - Wartość do zapisania w schowku systemowym.

## Wysłanie maila

Wysłany mail nie jest anonimowy, lecz zostaje wysłany w imieniu konkretnego konta mailowego. Aby więc wysłać maila należy podać namiary na konto mailowe (adres SMTP, adres serwera i inne pola - patrz właściwości połączenia. Jako klucze podawaj klucze domyślnego biblioteki mailowej dla Java'y) oraz dane dotyczące maila (od, do, tytuł i treść). Wielu dostawców internetu oraz administratorów blokuje możliwość wysyłania maili przez aplikacje java'owe, więc pomimo poprawnego skonfigurowania komendy, może ona nie zadziałać poprawnie.

Parametry:

- **Mail od** - Wyrażenie napisowe - Adres mailowy nadawcy maila,
- **Mail do** - Wyrażenie napisowe - Adres mailowy adresata maila,
- **Tytuł maila** - Wyrażenie napisowe - Tytuł maila,
- **Treść maila** - Wyrażenie napisowe - Treść maila.

## 4.6.13 Dźwięk

### Odtwórz dźwięk

Zostanie odegrany dźwięk. Dźwięk ten może zostać wczytany w czasie developmentu komendy (opcja „*Odtwórz dźwięk ze zmiennej*”) lub załadowany z podanego adres url dopiero w czasie wykonywania procesu z procedurą. Komenda uruchamia odtwarzanie dźwięku i natychmiast kończy działanie dając możliwość odtworzenia kolejnej komendy. Nie należy uruchamiać dwóch dźwięków jednocześnie. Zatrzymanie dźwięku czy jakakolwiek inna kontrola nad nim odbywa się przy pomocy innych komend. Procedura potrafi uruchomić odtwarzanie dźwięku w formacie mp3, wav oraz zależnie od implementacji języka Java także czasem au, snd, aiff, aiffc.

Parametry:

- **Nazwa odtwarzacza** - Napis - Nazwa odtwarzacza dźwięku, na którym zostanie uruchomione odtwarzanie materiału dźwiękowego.,
- **Źródło pliku muzycznego** - Wyrażenie napisowe - Źródło pliku muzycznego - może to być plik na dysku, bądź zmienna w programie Lordui.

### Zatrzymanie odtwarzania

Zatrzymanie odtwarzania nagrania.

Parametry:

- **Nazwa odtwarzacza** - Napis - Nazwa odtwarzacza dźwięku, na którym zatrzymać odtwarzanie dźwięku..

### Czekanie na zakończenie dźwięku

Komenda zablokuje odtwarzanie do momentu zakończenia odtwarzania dźwięku.

Parametry:

- **Nazwa odtwarzacza** - Napis - Nazwa odtwarzacza dźwięku, na którym rozpocząć nasłuchiwanie..

### 4.6.14 Sound and Video

Narzędzia do nagrywania filmików i dźwięku

#### Dodaj obrazki do filmiku

Dodaje obrazki(pojedyncze klatki) do filmiku. Dźwięk (o ile obsługiwany) powinien być obsługiwany osobno.

Parametry:

- **Zmienna nagrywacza filmiku** - Wyrażenie napisowe - Nazwa obiektu nagrywającego filmik, do którego zostaną dodane obrazki,
- **Obrazek** - Wyrażenie obrazka - Obrazek dodany do filmiku. Musi być rozmiaru obrazu ustawionego w filmiku.

#### Zamknij nagrywacza

Zamyka nagrywacza filmiku - po wywołaniu tej komendy filmik będzie gotowy.

Parametry:

- **Zmienna nagrywacza filmiku** - Wyrażenie napisowe - Nazwa obiektu nagrywającego filmik.

#### Inicjowanie nagrywacza

Inicjuje nagrywacza filmiku. Ta komenda musi zostać wywołana przed rozpoczęciem jakiegokolwiek innej akcji - np. nagrywania.

Parametry:

- **Zmienna nagrywacza filmiku** - Wyrażenie napisowe - Nazwa obiektu nagrywającego filmik,
- **Plik docelowy** - Wyrażenie napisowe - Plik w którym zostanie zapisany filmik (format \*.mov),
- **Rozmiar klatki** - Wyrażenie punktu - Rozmiar klatki filmu (width, height) - format punktu jest akceptowany,

- **Częstotliwość** - Wyrażenie liczbowe - Częstotliwość klatek na (podana w ilości klatek na sekundę),
- **Ustawienia dźwięku** - Ustawienia dźwięku - Ustawienia dźwięku (format \*.wav).

### Rozpocznij nagrywanie

Rozpoczyna nagrywanie filmiku. Obiekt nagrywający powinien być zainicjowany. Parametry:

- **Zmienna nagrywacza filmiku** - Wyrażenie napisowe - Nazwa obiektu nagrywającego filmik,
- **Obszar nagrywania** - Wyrażenie punktu/obszaru - Lewy-górny róg lub cały obszar ekranu, który obszar nagrywany,
- **Źródło dźwięku** - Wyrażenie napisowe - Nazwa źródła dźwięku. Możesz sprawdzić jakie źródła są dostępne lokalnie na Twoim komputerze,
- **Uruchamiaj procesor obrazu** - Wyrażenie logiczne - Każda klatka może być dodatkowo obrabiana każdorazowo przy dodaniu do filmiku.

Podprocedury:

- **Przerabianie klatek** - Jeśli przerabianie klatek jest włączone, procedura zostanie uruchomiona dla każdej pojedynczej klatki. Może ona np. dorysowywać kursor lub dorabiać ramkę..

### Zatrzymaj nagrywanie

Zatrzymuje nagrywanie obrazu z monitora. Filmik po wywołaniu tej funkcji nie będzie jeszcze gotowy. Należy wywołać wcześniej obiekt zamykający (lub np. nagrać kolejny fragment ekranu).

Parametry:

- **Zmienna nagrywacza filmu** - Wyrażenie napisowe - Nazwa obiektu nagrywającego filmik.

#### 4.6.15 Tworzenie obiektów lordui

Zestaw elementów pozwalających dynamicznie modyfikować procedury Lordui. Twórz obiekty za pomocą ich definicji XML (Definicje obiektów w formie XML możesz zobaczyć eksportując projekt w formacie XML lub kopiując poszczególne elementy do schowka systemowego i wklejając do notatnika)

### Tworzenie komendy

Dodaje nową komendę do procedury

Parametry:

- **Nazwa procedury** - Wyrażenie napisowe - Pełna nazwa procedury (łącznie ze ścieżką pakietów),
- **Definicja w formacie XML** - Wyrażenie napisowe - Definicja komendy w formacie XML. Definicje obiektów w formie XML możesz zobaczyć eksportując projekt w formacie XML lub kopiując poszczególne elementy do schowka systemowego i wklejając do notatnika,
- **Pozycja** - Wyrażenie liczbowe - Numer pozycji (licząc od zera) na której wstawić nowy obiekt. Dla wartości -1 dokleja komendę na koniec procedury..

### Tworzenie procedury

Stworzenie nowej, pustej procedury Lordui. Jeśli procedura o podanej ścieżce nie istnieje, zostanie rzucony wyjątek.

Parametry:

- **Nazwa procedury** - Wyrażenie napisowe - Pełna nazwa procedury (łącznie ze ścieżką pakietów),
- **Typ zwracanej wartości** - Wyrażenie napisowe - Pozostaw pusty, jeśli procedura ma nie zwracać wartości. Możliwe typy:
  - Point
  - PointList
  - String
  - Integer
  - Image
  - PositionedImage
  - Boolean
  - Rectangle
  - MediaPlayer
  - Other
  - Array
  - Collection
  - Sound
  - Expression

- Color
- ScreenBlockade
- Null
- JavaUIObject
- Long
- Map

,

- **Argumenty** - Dynamiczna lista argumentów procedury - Argumenty procedury. Lista możliwych typów argumentów jest taka sama jak lista typów zwracanych wartości.

### Usunięcie komendy

Usuwa komendę procedury. Jeśli procedura o podanej ścieżce nie istnieje, zostanie rzucony wyjątek. Jeśli procedura nie ma komendy o podanym numerze, nic nie zostanie wykonane.

Parametry:

- **Nazwa procedury** - Wyrażenie napisowe - Pełna nazwa procedury (łącznie ze ścieżką pakietów),
- **Numer komendy** - Wyrażenie liczbowe - Numer komendy procedury (pomiędzy 0, a ilość komend-1).

### Usunięcie procedury

Usuwa procedurę o podanej ścieżce (uwzględniając nazwy pakietów). Jeśli procedura o podanej ścieżce nie istnieje, zostanie rzucony wyjątek.

Parametry:

- **Nazwa procedury** - Wyrażenie napisowe - Pełna nazwa procedury (łącznie ze ścieżką pakietów).

## 4.6.16 Lordui meta-elementy

Za pomocą meta-poleceń można kontrolować odtwarzanie procedur lordui.

### Zamknięcie Lordui

Zostaną zwolnione wszystkie zasoby używane przez Lordui. Program zostanie zamknięty.

## **Pauza**

Wstrzymuje wykonywanie procedur. Więcej informacji na temat uruchamiania procedur znajduje się w rozdziale 4.3.1: „Uruchamianie procedury”

## **Zatrzymaj wszystko**

Zatrzymanie wszystkich uruchomionych procedur. Komenda wykonuje tą samą operację co wciśnięcie klawisza *Scroll lock*

## **Zatrzymaj nasłuchiwanie**

Zatrzyma wszystkie nasłuchiwanie podłączone do podanego odtwarzacza. Np. w przypadku nasłuchiwanie urządzeń wejścia - po uruchomieniu go można zakończyć nasłuchiwanie urządzeń wejścia, by nie reagować już na akcje urządzenia wejścia, jednak odtwarzacz dokończy wszystkie uruchomione dotąd wątki.

Parametry:

- **Nazwa odtwarzacza** - Wyrażenie napisowe - Nazwa odtwarzacza, dla którego mają zostać zatrzymane wszystkie nasłuchiwanie.

## **Zatrzymaj odtwarzacz**

Zatrzyma wybrany odtwarzacz i wszystkie zlecone mu wątki.

Parametry:

- **Nazwa odtwarzacza** - Wyrażenie napisowe - Nazwa odtwarzacza do zatrzymania.

## **4.6.17 Polecenia standardowe**

Klasyczne polecenia języka programowania takie jak warunek „if”, pętla, przypisanie zmiennej czy spanie.

## **Nowy nasłuchiwanie Lordui**

Tworzy nowego nasłuchiwanie zdarzeń Lordui. Nasłuchiwanie będzie uruchamiać wprowadzoną procedurę na zdarzenie zatrzymania wykonywania procedury Lordui.

Parametry:

- **Nazwa nasłuchiwanie** - Wyrażenie napisowe - Nazwa nasłuchiwanie zdarzeń zatrzymania procedur,



- **Uruchom tylko raz** - Wyrażenie logiczne - Czy zatrzymać nasłuchiwacz po pierwszym otrzymanym zdarzeniu zatrzymania procedur,
- **Nazwa odtwarzacza** - Wyrażenie napisowe - Nazwa odtwarzacza, który ma być obserwowany pod kątem sygnału zatrzymania.

Podprocedury:

- **Uruchamiana procedura** - Procedura uruchomiona przy każdym zdarzeniu zatrzymania procedury.

### Dodaj do tablicy

Wstawia wartość do tablicy.

Parametry:

- **Nazwa zmiennej** - Nazwa zmiennej - Nazwa zmiennej tablicy. Jeśli pod podaną zmienną nie ma obiektu, zostanie stworzona nowa tablica,
- **Typ wstawienia** - Opcje - Jest kilka możliwości wstawienia wartości:
  1. Ustaw wartość (żadne przesunięcie nie zostanie wykonane). Jeśli pod podanym indeksem znajduje się już jakiś obiekt, zostanie on nadpisany,
  2. Wstaw wartość - wszystkie elementy od podanego indeksu zostaną przesunięte o jedną pozycję dalej,
  3. Wstaw na koniec - skrót syntaktyczny do „wstaw wartości” (przy tej opcji indeks nie jest brany pod uwagę).
- **Indeks** - Wyrażenie liczbowe - Indeks pod którym ma zostać wstawiona wartość,
- **Wstawiana wartość** - Wyrażenie obiektu - Wartość do wstawienia do tablicy.

### Wywołanie procedury

Zostanie uruchomiona nowa podprocedura.

Parametry:

- **Nazwa wywołanej procedury** - Napis - Pełna nazwa procedury, która ma zostać uruchomiona np.: *pakiet1:pakiet2:nazwa\_procedury*,
- **Liczba powtórzeń procedury** - Wyrażenie liczbowe - Ile razy procedura zostanie uruchomiona,

- **Uruchom w oddzielnym wątku** - Pole wyboru - Czy procedura powinna zostać w oddzielnym wątku - procedura będzie kontynuowała działania równoległe z działaniem procesu uruchomionego w tle,
- **Argumenty procedury** - Wstawiane automatycznie - Każdy argument musi zostać uzupełniony. Liczba argumentów i ich typy są zdefiniowane w nagłówku procedury.

### **Dynamiczne wywołanie procedury**

Zostanie uruchomiona podprocedura. Należy podać jej nazwę oraz listę argumentów. Oba pola są dynamicznie obliczane w czasie działania procesów. Dostępna jest opcja uruchomienia procedury w nowym wątku (w jednym momencie tylko jeden wątek jest aktywny). Jeśli procedura ma zostać uruchomiona w nowym wątku, rozpocznie wykonywanie dopiero gdy aktualny wątek zakończy działanie lub wykona spanie z opcją „Możliwości zmiany wątku”.

Parametry:

- **Nazwa wywołanej procedury** - Wyrażenie napisowe - Nazwa zmiennej tablicy. Jeśli pod podaną zmienną nie ma obiektu, zostanie stworzona nowa tablica,
- **Argumenty procedury** - Wyrażenie tablicowe - Tablica z argumentami wywołania procedury,
- **Uruchom w oddzielnym wątku** - Pole wyboru - Czy procedura powinna zostać w oddzielnym wątku - procedura będzie kontynuowała działania równoległe z działaniem procesu uruchomionego w tle.

### **Zamknięcie Lordui**

Zostaną zwolnione wszystkie zasoby używane przez Lordui. Program zostanie zamknięty.

### **Instrukcja IF**

Instrukcja działa dokładnie tak, jak w większości popularnych języków programowania. Jeśli warunek (wyrażenie logiczne) zostanie spełniony, wywołana jest procedura „*On true*”, w przeciwnym razie wywołana jest procedura „*On false*”. Procedury te można edytować w oknie procedur, jako podelementy komendy IF. Warunek instrukcji if musi być zdefiniowany np.:  
`val1=2 || val2 = „aa” || true`

Parametry:

- **Wyrażenie** - Wyrażenie logiczne - wyrażenie obliczane jako warunek operacji „if”.

Podprocedury:

- **OnTrue** - Uruchomiona gdy wyrażenie logiczne wylicza się do prawdy,
- **OnFalse** - Uruchomiona gdy wyrażenie logiczne wylicza się do fałszu.

### Pętla While

Pętla while działa jak w większości popularnych języków programowania. Sprawdzany jest warunek (wyrażenie) pętli, a gdy jest prawdziwe, zostaje wywołana podprocedura komendy „*Loop body*”. Komenda pętli while zostanie zakończona dopiero, gdy wyrażenie warunku pętli wyliczy się do wartości false. Podprocedurę można edytować w oknie procedur, jako podelement pętli while. Warunek instrukcji if musi być zdefiniowany np.: `val1=2 // val2 = „aa” // true`

Parametry:

- **Warunek pętli while** - Wyrażenie logiczne - wyrażenie logiczne pętli while.

Podprocedury:

- **LoopBody** - Uruchomione, gdy wyrażenie logiczne jest spełnione.

### Usuń nasłuchiawcz Lordui

Usuwa nasłuchiawczą operację Lordui. Nasłuchiawcz nie będzie odtąd śledził zatrzymania procedur Lordui.

Parametry:

- **Nazwa nasłuchiawcza** - Wyrażenie napisowe - Nasłuchiawcz do zatrzymania.

### Return

Komenda przerywa wykonywanie aktualnej procedury i zwraca podaną wartość. Zwracana wartość powinna być zgodna z typem podanym w nagłówku procedury

Parametry:

- **Zwracana wartość** - Wyrażenie obiektu - Wartość zwracana przez funkcję.

## Spanie

Spanie czyli czekanie/zawieszenie wykonywania procedury

Parametry:

- **Długość spania** - Wyrażenie liczbowe - Długość spanie wyrażona w milisekundach,
- **Pozwól na zmianę wątku** - Pole wyboru - Czy w trakcie spania inna procedura może wznowić działanie.

## Try ... catch

Przechwytywanie wyjątków

Parametry:

- **Wzór nazwy wyjątków** - Wyrażenie napisowe - Nazwa klasy wyjątku do przechwycenia..

Podprocedury:

- **Body** - Kod do uruchomienia, którego wyjątki zostaną przechwyczone,
- **Obsługa wyjątków** - Kod uruchomiony po przechwyceniu wyjątku.

## Przypisanie zmiennej

Przypisanie zmiennej (lub uusnięcie, gdy wyrażenie jest puste) służy przypisaniu wartości wyliczonej w danym momencie do nazwy zmiennej. **Zalecane jest przypisywanie do jednej zmiennej zawsze wartości tego samego typu.**

Parametry:

- **Nazwa zmiennej** - Nazwa zmiennej - Nazwa zmiennej pod jaką ma zostać zapisana wartość. Jeśli zmienna istnieje, zostanie nadpisana. W przypadku usuwania zmiennej usunięcie zmiennej wywołania procedury lub zmiennej globalnej. Aby usunąć obie, należy użyć dwóch komend procedury „przypisanie zmiennej”,
- **Wyrażenie** - Wyrażenie obiektu - Wartość do zapisania. Pozostaw puste, aby usunąć zmienną.

## Wczytaj zmienne

Wczytanie zmiennych z pliku XML.

Parametry:

- **Ścieżka pliku** - Wyrażenie napisowe - Ścieżka pliku XML do wczytania.

## Logowanie

Dopisanie wpisu do logu.

Parametry:

- **Logowana wartość** - Wyrażenie napisowe - Tekst do wpisania do logu.

## Zapisz zmienne

Zapisanie zmiennych do pliku XML podczas wykonywania procedury. Zapisane zmienne mogą później zostać otwarte np. jako projekt (procedury nie są jednak zapisywane).


Parametry:

- **Ścieżka pliku** - Wyrażenie napisowe - Nazwa pliku do którego zostaną zapisane zmienne,
- **Nazwa zmiennej** - Dynamiczna lista zmiennych - Nazwy zmiennych do zapisania w pliku.

### 4.6.18 Rozszerzenia


Do Lordui można implementować własne komendy. Dzięki temu Lordui może m.in. zostać użyte jako biblioteka przy programie zewnętrznym.

## 4.7 Tworzenie klasycznych makr

LordUI pozwala także tworzyć klasyczne makra. Aby nagrać makro, wybierz . Do wyboru będzie kilka opcji. Możesz zdecydować, ile ruchów myszy zostanie nagranych (możesz nie zapisywać żadnych, traktować wszystkie ruchy myszy pomiędzy poszczególnymi kliknięciami jako pojedynczy ruch myszy lub zapisać każde zdarzenie ruchu myszy osobno) oraz ile przeciągnięć myszy zostanie nagranych. Pomiedzy każdą operację można wstawić stałą długość spania. Dostępne są także skróty do tworzenia zdjęć ekranu (oraz fragmentu zdjęć), a także wybierania punktów na ekranie. Można także się przełączyć do małego widoku okna nagrywania.

Aby nagrać makro, należy kliknąć przycisk nagrywania. Uwaga: operacje wykonywane na oknie nagrywania nie są nagrywane. Nagrywanie można wstrzymać lub zatrzymać. Po zatrzymaniu, powstanie nowa procedura zawierająca akcje nagranych makra (procedura przyjmie nazwę Proc z sufiksem jej numeru).

## 4.8 Przygotowywanie zmiennych obrazkowych

LordUI jest wyposażone w narzędzie do obróbki obrazków, by mogły one lepiej służyć przy wyszukiwaniu wzorców. Aby zobaczyć czy wzorec odbiega od fragmentu obrazu wyświetlanego aktualnie na ekranie, w liście zmiennych, wybierz zmienną obrazkową a następnie kliknij przycisk . Pokaże się okno ze zrzutem ekranu, zaś w lewym górnym rogu wyświetlony zostanie wzorec. Wzorec można przeciągać myszką oraz przesuwac strzałkami kursora klawiatury. Przy pomocy przycisków na górnym pasku okna można zobaczyć w którym miejscu wzorec różni się od fragmentu zrzutu ekranu, wypalić różniące się pixle, lub zapisać otrzymany wzorec jako zmienną. Najedź myszką na poszczególne przyciski, by zobaczyć podpowiedzi. Przy pomocy suwaka można wyświetlić wzorec częściowo przezroczystym, by lepiej umiejscowić go na zrzucie ekranu.

## 4.9 Natywna część Lordui

Program Lordui napisany jest w języku Java, dzięki czemu jego silnik może działać na różnych platformach systemów operacyjnych. Jednak ze względu na specyfikę programu, używany jest szereg komend specyficznych dla systemu operacyjnego. Sprawia to, że tworząc projekty w Lordui trzeba dołożyć specjalnej staranności, by działały one na możliwie szerokiej gamie systemów operacyjnych. W poniższym rozdziale opisane zostaną zagadnienia specyficzne dla poszczególnych modułów oraz systemów operacyjnych

### 4.9.1 Rozmycie obrazu

OCR programu Lordui oraz wyszukiwanie wzorca zaimplementowane zostały 100% w języku Java. Niestety jednak, w niektórych systemach operacyjnych (np. Microsoft Windows) używane jest rozmycie obrazu, które w teorii powinno wygładzić napisy i ułatwić pracę z komputerem. W niektórych systemach operacyjnych (np. Microsoft Windows Vista i nowszych) rozmycie obrazu domyślnie jest włączone. Microsoft nazywa takie rozmycie jako „*ClearType*”. Przy tworzeniu projektów należy zwrócić uwagę na wszelkie rozmycie (zarówno tekstu, jak i całego obrazu) z m.in. dwóch powodów:

1. Rozmycie obrazu sprawia, że moduł OCR nie będzie działać
2. Wzorce zdefiniowane w projekcie na komputerze z włączonym „*ClearType*” mogą nie pasować do wzorców zdefiniowanych na komputerze bez włączonej tej opcji, lub inną jej konfiguracją.

**Dlatego zalecane jest wyłączenie opcji „*ClearType*”, lub zachowanie wszelkiej ostrożności w przypadku niemożliwości jej wyłączenia.**

### 4.9.2 Operacje natywne

Niektóre operacje - tj. odczytywanie zdarzeń systemowych czy obsługa okien aplikacji zewnętrznych obsługiwane są przy pomocy komend natywnych. Autor programu sprawdził możliwość uruchomienia tych komend pod

- Microsoft Windows XP service pack 3, wersji 32 bitowej
- Microsoft Windows 7, wersji 64 bitowej


Autor będzie wdzięczny za informacje zarówno o pomyślnym uruchomieniu programu na różnych środowiskach, jak i wszelkich problemach. Ważne będą wszelkie informacje nt. - nazwy i wersji systemu operacyjnego oraz wersji środowiska Java.

### 4.9.3 Przezroczyste okna

Program Lordui w wielu miejscach korzysta z przezroczystości. Wszystkie nowsze systemy operacyjne z rodziny Microsoft Windows obsługują wyświetlanie przezroczystości. Autor programu spotkał się z problemami przy obsłudze przezroczystości z poziomu Java'y w przypadku niektórych systemów operacyjnych z rodziny Linux'a. W takim przypadku praca z programem będzie bardzo utrudniona i jest odradzana.

## 4.10 Moduły Lordui

Lordui udostępnia zestaw opcjonalnych modułów. Moduły udostępnione na oficjalnej liście są zarządzane przez autora programu i tylko on może zmieniać zbiór dostępnych pozycji. Jednocześnie istnieje jednak możliwość zaimplementowania własnego modułu. Głównym celem funkcjonalności modułów jest wyjęcie z Lordui sporej części funkcjonalności, które byłyby zbyt rzadko używane przez większość użytkowników.

Zestawem modułów możesz zarządzać w oknie zarządzania modułami. Znajduje się ono pod „Pomoc” -> „Moduły”. Moduły można przeglądać/odinstalować lub instalować zarówno z serwera, jak i lokalnego pliku. Domyślnie używany jest oficjalny serwer modułów Lordui, jednak można to zmienić używając przycisku: .

Aby zainstalować moduł, wybierz go na liście po lewej stronie okna modułów oraz wciśnij „Instaluj” w górnej części okna. W oknie tym można także aktualizować moduły, których aktualizacja dostępna jest na serwerze.

### 4.10.1 Moduł Zdalnej Java'y

Za pomocą modułu Zdalnej Java'y można podłączyć się do zdalnej aplikacji Java. Tymsamym można kontrolować aplikację nie tylko za pomocą obrazków (screenshot), ale także i odwołując się poprzez referencję.

## Podłączenie klienta zdalnej Java'y

Po zainstalowaniu modułu „Remote Java Module”, Lordui będzie służyło jako Server Zdalnej Java'y. Na pojedynczym komputerze może być uruchomiony tylko jeden taki server. Każda inna instancja Lordui uruchamiana na Lordui uruchomi się bez włączonego Serwera Zdalnej Java'y. Serwer używa technologii Java RMI na porcie 7231.

Każda aplikacja Java może być podłączona jako klient. Aby to zrobić, należy wykonać dwa kroki (przy czym, jak piszę później - pierwszy krok nie zawsze jest konieczny)

1. dodaj bibliotekę jar Zdalnego Klienta Java do ścieżki (tzw. classpath),
2. uruchom klienta.

Aby załączyć klienta w ścieżce (tzw. classpath), pobierz najnowszą wersję ze strony [www.lordui.com](http://www.lordui.com). Są dwie polecane drogi dołączenia bibliotki Zdalnego Klienta Java:

- dodanie go bezpośrednio do ścieżki, jako zwykłą bibliotekę \*.jar,
- dodanie biblioteki do ścieżki kompilacyjnej wirtualnej maszyny Java tak, by każda aplikacja Java uruchomiona na komputerze uruchomiła się z bibliotekę Lordui.


Mając załadowaną bibliotekę Zdalnego Klienta Java, ostatnim krokiem jest uruchomienie klienta, aby skomunikował się z serwerem. Są dwie drogi: Pierwszy polega na uruchomieniu klienta wprost z kodu za pomocą komendy:

```
LorduiRMIClient.execute();
```


Drugi sposób jest trochę bardziej skomplikowany, ale za to nie wymaga dorzucania pliku Jar do ścieżki programu. Uruchomimy klienta, który natychmiast połączy się z serwerem, a następnie sam uruchomi Twoją Aplikację Java. Zdalnego Klienta Java uruchomimy z linii komend. Będziemy musieli przekazać ścieżkę do wszystkich bibliotek. Klasą startową Zdalnego Klienta Java jest `modules.ktm.autoclicker.remoteJava.client.JavaRemoteClient`. Jako argumenty należy podać: „main\_class”, „pełna.ścieżka.Twojej.klasy.głównej”, „argument pierwszy Twojej aplikacji”, „argument drugi Twojej aplikacji”, ... . Przykładowo założmy, że Twoja aplikacja jest uruchamiana komendą: `java -jar my_app.jar "arg1"`. Patrząc do pliku Manifest w `my_app.jar` wiemy, że główną klasą jest `com.application.Main` (definiuje ona funkcję `main(String[])`). Aby uruchomić program za pomocą Zdalnego Klienta Java możemy użyć np.:

```
java -cp "my_app.jar;RemoteJavaClient.jar"  
modules.ktm.autoclicker.remoteJava.client.JavaRemoteClient  
main_class com.application.Main arg1
```



. Ta komenda uruchomi Zdalnego Klienta Java, a następnie uruchomi Twoją aplikację tak, jak wcześniej była ona uruchamiana z linii komend. Wśród ikonek paska start, pojawi się ikonka , która informuje o działającym Zdalnym Kliencie Java.

### Konsola Zdalnej Java'y

W aplikacji Lordui, po zainstalowaniu Modułu Zdalnej Java'y, na pasku ikon po prawej stronie dostępna będzie nowa ikonka: . Kliknij ją, aby otworzyć Konsolę Zdalnej Java'y.

Zakładka Klienci pokazuje listę podłączonych aktualnie klientów. Zaznaczając opcję „Uruchom klienta lokalnie”, możesz podłączyć aplikację Lordui samą do siebie jako Zdalnego Klienta Java. Opcja ta jednak jest zawsze wyłączona przy każdym uruchomieniu programu. Zakładka „Badaj” pozwala zbadać obiekty GUI Zdalnych Klientów Java. Wybierz przycisk „Wybierz zdalny obiekt”. Następnie przesunij myszkę nad obiekt GUI (Twojej podłączonej aplikacji), który chcesz zbadać. Okno powinno się przyszarzyć. Kliknij obiekt (zdarzenie kliknięcia nie zostanie wysłane do aplikacji). W oknie konsoli zobaczysz podstawowe właściwości obiektu. Możesz zapisać każdą z wyświetlonych wartości - np. możesz zapisać obrazki klikając prawym przyciskiem myszki na nim. Za pomocą konsoli możesz także zbudować formułę, za pomocą której procedury będą mogły pobierać obiekt GUI. W tym celu zaznacz właściwości obiektu, które wyznaczają obiekt jednoznacznie. Następnie zaznacz „Zbuduj formułę”. Zostanie wyświetlona informacja ile obiektów okna ma takie właściwości. Formuła będzie podana nad klikniętym guzikiem. Możesz skopiować ją przy pomocy guzika „Kopiuj do schowka”. Na zakładce „Stos” możesz zobaczyć pełną listę obiektów-właścicieli od okna, aż do wskazanego obiektu.

Konsolę Java znajdziesz także znaleźć w właściwościach obiektów (np.: „Wywołaj metodę Java”), czym można łatwo wyklikać formułę wskazującą obiekt GUI.