Java

GUI

FSR Informatik

October 11, 2016

Overview

This lecture covers the basic principles of how to program Graphical User Interfaces (GUI) in Java.

We will use a lightweight and simple package called *Simple GUI*. You can download it here:
`http://users.ifsr.de/~fredo/2013/Java/SimpleGui/simple_gui.jar`

JAR

JAR stands for Java Archive. A `*.jar`-file ...

- contains `*.class` files
- may contains `*.java` files
- may be digital signed
- may be compressed

`*.class` are the compiled Java classes.
`*.java` are the sourcecode files.

JARs often used as single file Java executables.

Import JAR

1. open Project > Properties
2. select Java Build Path
3. select the tab Libaries
4. click Add External JARs...
5. choose the location to your JAR

The imported package can be found on *Referenced Libaries* in the Eclipse *Package Explorer*. Your own *default package* is located in *src*.

Do not move the JAR after the import. Eclipse will always reference the given location.

Javadoc

The Javadoc for Simple Gui is online:
`http://users.ifsr.de/~fredo/2013/Java/SimpleGui/doc/`

You see there are three classes and one interface.

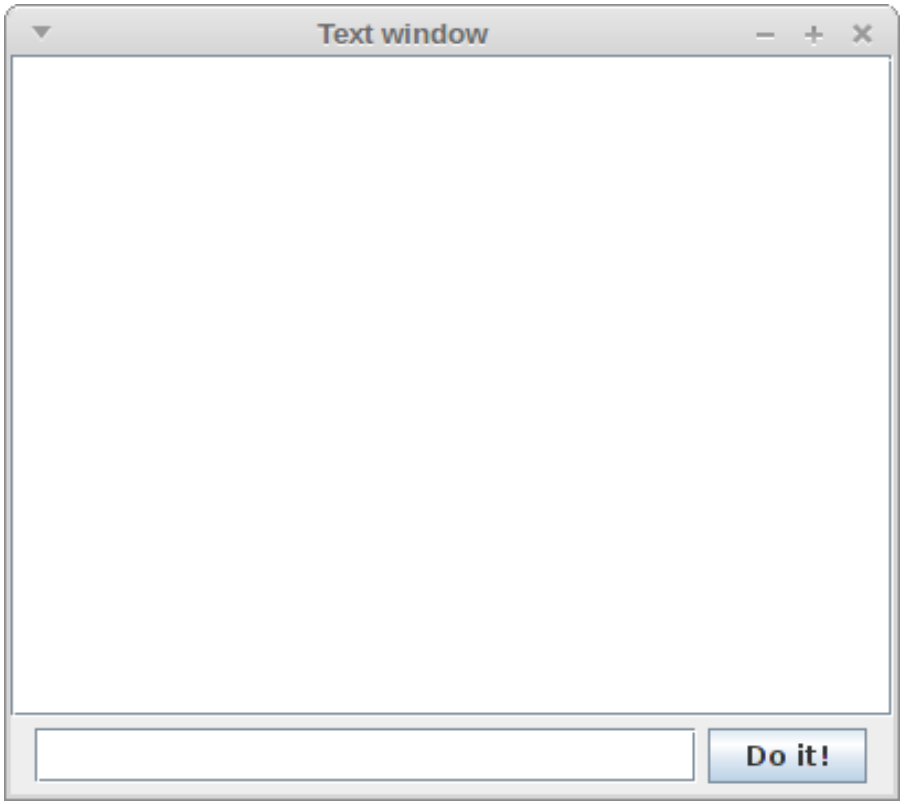
- Interface ButtonConfiguration
- Class ButtonWindow
- Class DrawingWindow
- Class TextWindow

The first Window

Windows in Java are treated as normal objects.

```
1   import simple_gui.*;
2
3   public class Example {
4
5       public static void main(String[] args) {
6
7           TextWindow window = new TextWindow();
8       }
9   }
10
```

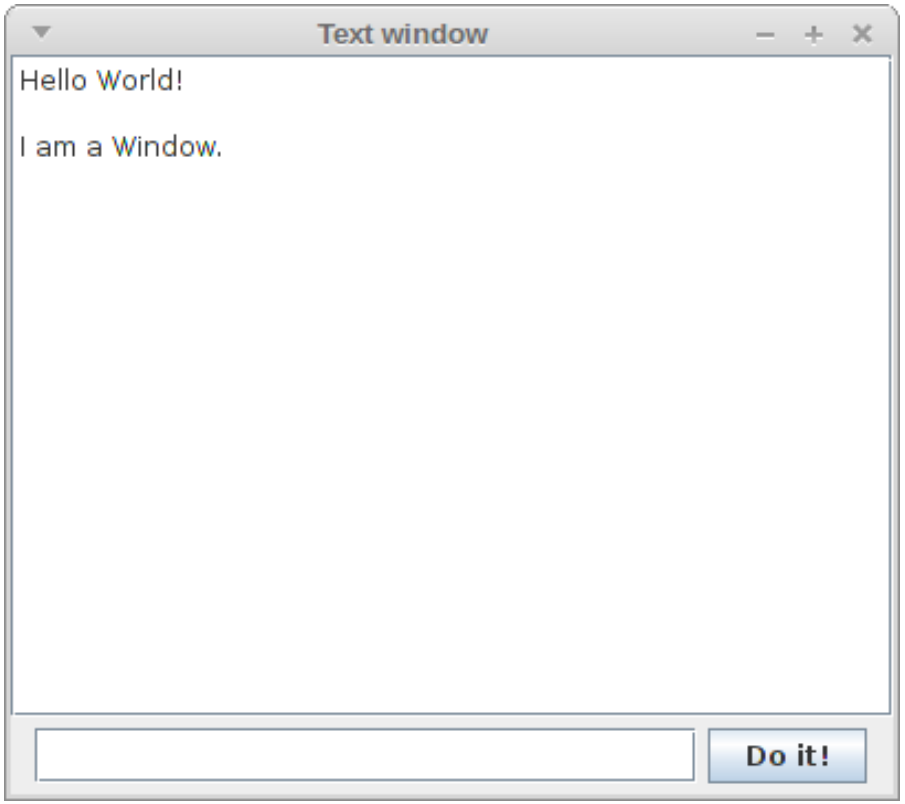
The first Window - Screenshot



Hello World

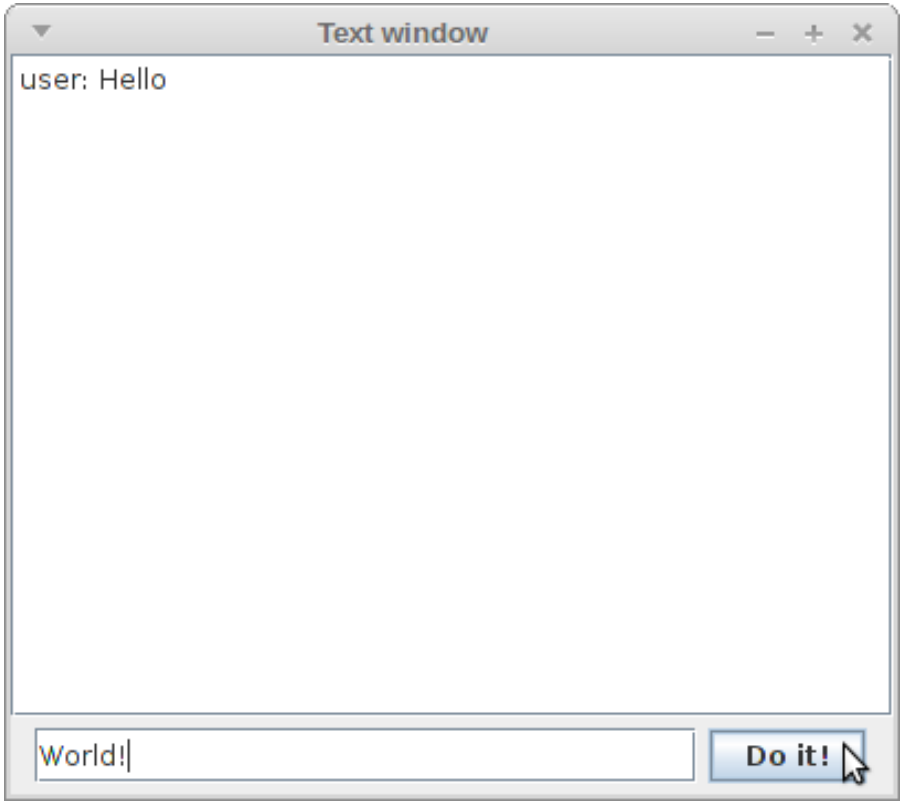
```
1   import simple_gui.*;
2
3   public class Example {
4
5       public static void main(String[] args) {
6
7           TextWindow window = new TextWindow();
8
9           window.addOutputLine("Hello World!");
10          window.addOutputLine("");
11          window.addOutputLine("I am a Window.");
12      }
13  }
14
```

Hello World - Screenshot



Input - Processing - Output - Loop

```
1       public static void main(String[] args) {
2
3           TextWindow window = new TextWindow();
4
5           while(true) {
6               if (window.isInputAvailable()) {
7                   // input:
8                   String str = window.getNextInputLine()
    ;
9                   // processing:
10                  str = "user: " + str;
11                  // output:
12                  window.addOutputLine(str);
13              }
14          }
15      }
16
```



During the second input.
First input was *Hello*. Second input will be *World!*.

Exit the Program

The `while(true)` statement implies the program will run forever. So you need a way to exit the program.

```
1      while(true) {
2          if (window.isInputAvailable()) {
3              // input:
4              String str = window.getNextInputLine();
5              // processing:
6              if (str.compareTo("exit") == 0) {
7                  window.close();
8                  return;
9              }
10             str = "user: " + str;
11             // output:
12             window.addOutputLine(str);
13         }
14     }
15
```

Buttons

```
1   import simple_gui.*;
2
3   public class ButtonTest {
4
5       public static void main(String[] args) {
6
7           // make a new window with 1 x 3 buttons
8           ButtonWindow window = new ButtonWindow(1,
    3);
9       }
10  }
11
```

Buttons - Screenshot

We build a window with three buttons which have no function.

ButtonWindow

The Javadoc says `configureButton(int buttonNumber, ButtonConfiguration config)` adds some configuration to the button.

ButtonConfiguration is an Interface that describes two methods:

- `String getButtonText()`
- `void onClickAction()`

We have to write a class that implements *ButtonConfiguration* and pass an instance to the method `configureButton(...)`.
The configured button will be named with the String the method *getButtonText()* will return. A clicked configured button will call the method *onClickAction()*.

CloseButton

```
1   import simple_gui.*;
2
3   public class CloseButton implements ButtonConfiguration
    {
4
5       @Override
6       public String getButtonText() {
7           return "Close";
8       }
9
10      @Override
11      public void onClickAction() {
12          // implementation follows
13      }
14  }
15
```

CloseButton - Test

```
1   import simple_gui.*;
2
3   public class ButtonTest {
4
5       public static void main(String[] args) {
6
7           // make a new window with 1 x 3 buttons
8           ButtonWindow window = new ButtonWindow(1,
    3);
9
10          // the numeration starts with 0
11          // so the right button is no. 2
12          window.configureButton(2, new CloseButton
    ());
13      }
14  }
15
```

CloseButton - Screenshot

The button is successfully renamed.

CloseButton - onClickAction()

The method `window.close()` closes the window. Unfortunately the class *CloseButton* can not access the reference `window`.

```
    @Override
    public String getButtonText() {
        return "Close";
    }

    @Override
    public void onClickAction() {
        // ...
    }

```

CloseButton - onClickAction()

The class receives the reference through the constructor.

```
1       private ButtonWindow window;
2
3       public CloseButton(ButtonWindow window) {
4           this.window = window;
5       }
6
7       @Override
8       public String getButtonText() {
9           return "Close";
10      }
11
12      @Override
13      public void onClickAction() {
14          this.window.close();
15      }
16
```

CloseButton - Final Test

```
1   import simple_gui.*;
2
3   public class ButtonTest {
4
5       public static void main(String[] args) {
6
7           // make a new window with 1 x 3 buttons
8           ButtonWindow window = new ButtonWindow(1, 3);
9
10          // pass the reference window as an argument
11          window.configureButton(2, new CloseButton(window
    ));
12      }
13  }
14
```

AbstractButton

```
1   import simple_gui.*;
2
3   public abstract class AbstractButton implements
    ButtonConfiguration {
4
5       protected ButtonWindow window;
6       private String label;
7
8       public AbstractButton(ButtonWindow window, String
    label) {
9           this.window = window;
10          this.label = label;
11      }
12
13      @Override
14      public String getButtonText() {
15          return label;
16      }
17  }
18
```

CloseButton

```
1   import simple_gui.*;
2
3   public class CloseButton extends AbstractButton {
4
5       public CloseButton(ButtonWindow window, String label
    ) {
6           super(window, label);
7       }
8
9       @Override
10      public void onClickAction() {
11          this.window.close();
12      }
13  }
14
```

Test

```
1   import simple_gui.*;
2
3   public class ButtonTest {
4
5       public static void main(String[] args) {
6
7           // make a new window with 1 x 3 buttons
8           ButtonWindow window = new ButtonWindow(1, 3);
9
10          window.configureButton(2,
11              new CloseButton(window, "Close"));
12      }
13  }
14
```