Graphical Interface

- GUI : Graphic User Interfac. Composition of predefined generic components.
- Dynamicity of program comes from events definition/analysis.
- All of GUI library take into account the window manager system: Gnome, KDE ... In Java, it is operating system independant.
- Main concept: the **Widget object**.

Graphical Interface

A lot of graphical libraries

- In C or C++ : tk, Gtk, tkinter, OpenGL, QT, wx ....
- Example in QT:

```
#include <qapplication.h>

#include <qpushbutton.h>

int main(int argc, int **argv)

{

     QApplication a(argc, argv);

     QPushButton hello(''Hello World !'', 0);

     hello.resize(100,30);

     a.setMainWidget(&hello);

     hello.show();

     return a.exec();

}
```

Basic Elements for GUI

Widget concepts

- Always create a main window which is in charge to link the application with the environment.
- Container widget which are able to arrange the components.
- A menu bar composed of buttons.
- Working area to display drawings, texts ....
- Daughter windows with *popup* behavior.

Organisation

- The main window is :
  - the root window of the windows hierarchy.
  - Has link with the display (your screen).
- Create composition :
  - The others windows/widgets are arranged inside the main window - see `JFrame` in Swing, `Panel`, `Canvas`...
  - Parameters allow to attach the widget from their sides to another one, to position them relatively to the *daughters* windows.

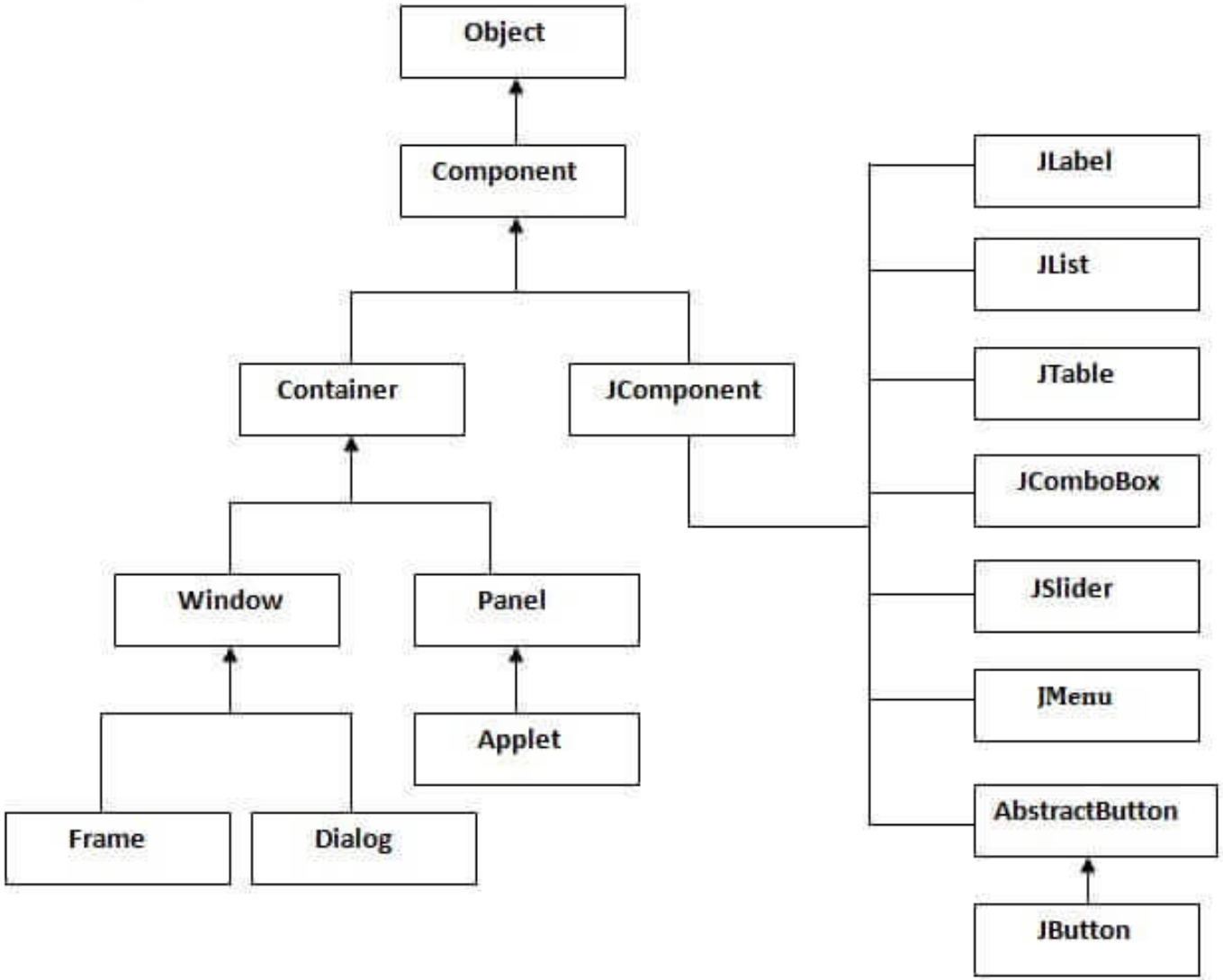
Java Libraries

- AWT : Abstract Windowing Toolkit, package `java.awt` the first one in Java.
- JFC/Swing : newest one, containing AWT. package `javax.swing`
- java3D : for example `javax.media.j3d`, that to have to add by hands in your library repository.

Swing

- Large number of components/widget types.
- Belong to the Java Fundation Classes (JFC), an API to provide GUI to java programs.
- Include the older toolkit AWT.

Swing Library Hierarchy



Container in Swing

- Container widgets can have other components on it.
- At least, one container is mandatory to create an application.
- Three main types of containers:
  - **Frame**: a fully functioning window with its title and icons (linked to the window manager application).
  - **Panel**: is not a window - just provide functionality to organize other widget on it.
  - **Dialog**: kind of popup window but not fully functional as Frame. Able to display message.

First Window

```
import javax.swing.*;

public class FirstSwingExample {
  public static void main(String[] args) {

      JFrame f=new JFrame();//creating instance of JFrame

      JButton b=new JButton("click");//creating instance of JButton
      b.setBounds(130,100,100, 40);//x axis, y axis, width, height

      f.add(b);//adding button in JFrame

      f.setSize(400,500);//400 width and 500 height
      f.setLayout(null);//using no layout managers
      f.setVisible(true);//making the frame visible
  }
}
```

By using the constructor

```
import javax.swing.*;
public class Simple {
  JFrame f;
  Simple(){
  f=new JFrame();//creating instance of JFrame

  JButton b=new JButton("click");//creating instance of JButton
  b.setBounds(130,100,100, 40);

  f.add(b);//adding button in JFrame

  f.setSize(400,500);//400 width and 500 height
  f.setLayout(null);//using no layout managers
  f.setVisible(true);//making the frame visible
 }

 public static void main(String[] args) {
  new Simple();
 }
}
```

Same with inheritance

```
import javax.swing.*;

public class Simple2 extends JFrame{//inheriting JFrame
  JFrame f;
  Simple2(){
     JButton b=new JButton("click");//create button
     b.setBounds(130,100,100, 40);

     add(b);//adding button on frame
     setSize(400,500);
     setLayout(null);
     setVisible(true);
  }
  public static void main(String[] args) {
     new Simple2();
  }
}
```

Swing Library Hierarchy

Basic ... istn't it !

SwingSet Demo

A full example

- All the widget types are presented.
- Code available
- Use both from `.jar` or as *web application*.
- Applet widget
  - Declare a class that `extends JApplet`
  - Call to the methods `init`, `start`, `stop` and `destroy` are most often implicit.
  - Two steps :
    - Java virtual machine instanciates the Applet object by calling default constructor.
    - Java virtual machine send the message `init` to the Applet object.
  - A method `update` is called each time the screen is refreshed. It calls the `paint` method.
  - Redefining `public void update(Graphics g){paint(g)}` suppress the automatic refresh of the screen.

Write HTML file

```
<html>
<head>
  <title>My application</title>
</head>

<body>
    <h1>My nice application</h1>
    <applet code="MyAppli.class"
            width=695 height=525>
    </applet>
</body>
</html>
```

SwingSet2

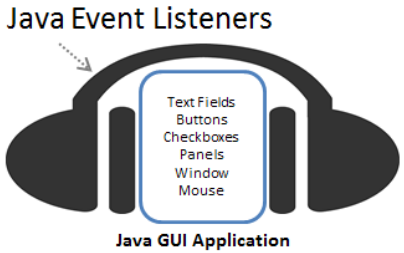
```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html>
  <head>
    <title>SwingSet demo</title>
  </head>

  <body>
      <h1>SwingSet demo</h1>
      <applet code=SwingSet2Applet
              archive="SwingSet2.jar"
             width=695 height=525>
      </applet>
  </body>
</html>
```

A lot of new version of browser do not support any more Applet - this way to do application will decline strongly !

Event-Driven Programming

Concepts

- Paradigm of programming opposite to the imperative programming.
- When an event happens a function is called. This change the program state.
- An event can be a user *button click*.
- Create *listener* objects to collect events through a loop of events.
- Graphical interface are based on this paradigm.



Event-Driven Programming

Java Context

- *Event* are objects - see the interface `Event` and its subclasses `UIEvent`, `MouseEvent` ...
- Listener has to be attached to widget.
- List of listener types :

Class name	Action
ActionListener	Responds to actions (e.g., button click, text field change)
ItemListener	Listens for individual item changes (e.g., a checkbox)
KeyListener	Listens for keyboard interaction
MouseListener	Listens for mouse events (double-click, right-click, etc.)
MouseMotionListener	Listens for mouse motion
WindowListener	Acts on events in the window
ContainerListener	Listens for container (JFrame, JPanel) events
ComponentListener	Listens for changes to components (e.g., a label being moved, hidden, shown, or resized in the program)
AdjustmentListener	Listens for adjustments (e.g., a scroll bar being engaged)

Example from the SwingSet

```
public JMenuItem createMenuItem(JMenu menu, String label, String mnemonic,
                                 String accessibleDescription, Action action) {
    JMenuItem mi = (JMenuItem) menu.add(new JMenuItem(getString(label)));
    mi.setMnemonic(getMnemonic(mnemonic));
    mi.getAccessibleContext().setAccessibleDescription(getString(accessibleDescription));
    mi.addActionListener(action);
    if(action == null) {
         mi.setEnabled(false);
    }
   return mi;
}
```

- See the file `EventDriven.java` to analyse another example.

A simple complete application

```
import javax.swing.*;
import java.awt.event.*;

public class swing1 extends JFrame {

   public swing1() {
      super("Application title");

      WindowListener l = new WindowAdapter() {
              public void windowClosing(WindowEvent e){
                        System.exit(0);
              }
           };

      addWindowListener(l);
      setSize(200,100);
      setVisible(true);
   }

   public static void main(String [] args){
      JFrame frame = new swing1();
   }
}
```

Now with a button

```
import javax.swing.*;
import java.awt.event.*;

public class swing2 extends JFrame {

   public swing2() {

     super("Application Title");

      WindowListener l = new WindowAdapter() {
         public void windowClosing(WindowEvent e){
           System.exit(0);
         }
      };
     addWindowListener(l);

     JButton bouton = new JButton("my button");
     JPanel panneau = new JPanel();
     panneau.add(bouton);

     setContentPane(panneau);
     setSize(200,100);
     setVisible(true);
   }

   public static void main(String [] args){
      JFrame frame = new swing2();
   }
}
```

Adding a Menu Bar

```
import javax.swing.*;
import java.awt.*;

public class TestJFrame6 {

  public static void main(String argv[]) {

    JFrame f = new JFrame("my window");
    f.setSize(300,100);
    JButton b =new JButton("my button");
    f.getContentPane().add(b);

    JMenuBar menuBar = new JMenuBar();
    f.setJMenuBar(menuBar);

    JMenu menu = new JMenu("File");
    menu.add(menuItem);
    menuBar.add(menu);

    f.setVisible(true);
  }
}
```

Java FX

New tool

- As the usage of Applet declines, the new way to developp web application emerge.
- JavaFx allows to developp RIA (Rich Internet Application) containing video, music, grapphical effects ....
- Web applications can be used with computer but also with mobile phone and even TV.
- To use it you need to download an additional module of the JRE.

New possibilities

- In addition to the classical objects, new objects :
  - Building diagrams(statistics).
  - Drawing shapes.
  - New layout.
  - Effects, annimations, media connexions ...

New Concepts

- A FxApplication is based on the concept of *scene*... as in a theater - it is represented by the type `Stage`.
- The first class, containing the `main`, has to inherit from the class `Application`.
- An `Application` define the `Stage`, which contains the *scene* where all the other objects are positioned.
- In a scene, a *root* object has to be defined, Only one root group can be defined in one scene.
- The root object contains all the other graphical objects of the application.
- An application has a method `launch` that calls the method `start` to run your graphical application.

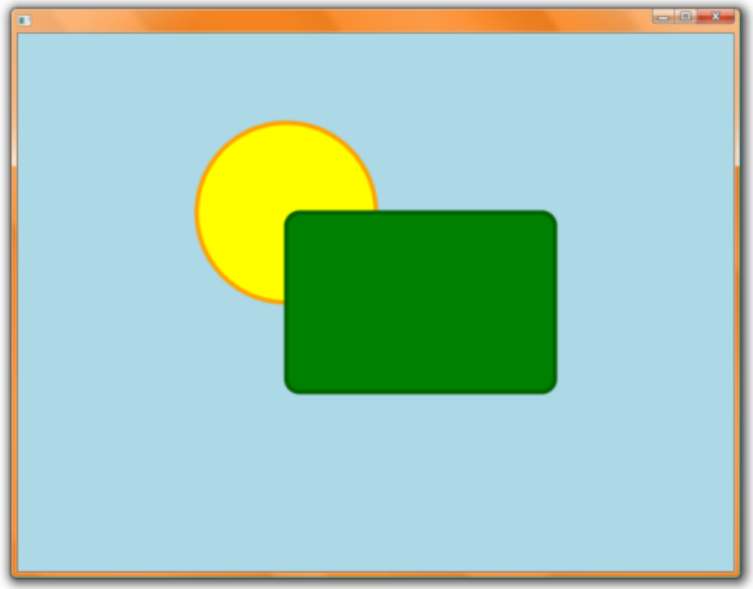
First lines

```
public class Test extends Application {

    public static void main(String[] args) {
        Application.launch(Test.class, args);
    }
    public void start(Stage primaryStage) {
        primaryStage.setTitle("Hello World");
        Group root = new Group();
        Scene scene = new Scene(root, 300, 250, Color.LIGHTGREEN);
        Button btn = new Button();
        btn.setLayoutX(100);
        btn.setLayoutY(80);
        btn.setText("Hello World");
        btn.setOnAction(new EventHandler<ActionEvent>() {

            public void handle(ActionEvent event) {
                System.out.println("Hello World");
            }
        });
        root.getChildren().add(btn);
        primaryStage.setScene(scene);
        primaryStage.setVisible(true);
    }
}
```

Display Shapes

```
public void start(Stage primaryStage) {
      Group root = new Group();
      Scene scene = new Scene(root, 800, 600, Color.LIGHTBLUE);
      primaryStage.setScene(scene);

      Circle cercle = new Circle();
      cercle.setCenterX(300);
      cercle.setCenterY(200);
      cercle.setRadius(100);
      cercle.setFill(Color.YELLOW);
      cercle.setStroke(Color.ORANGE);
      cercle.setStrokeWidth(5);

      Rectangle rectangle = new Rectangle();
      rectangle.setX(300);
      rectangle.setY(200);
      rectangle.setWidth(300);
      rectangle.setHeight(200);
      rectangle.setFill(Color.GREEN);
      rectangle.setStroke(Color.DARKGREEN);
      rectangle.setStrokeWidth(5);
      rectangle.setArcHeight(30);
      rectangle.setArcWidth(30);

      root.getChildren().add(cercle);
      root.getChildren().add(rectangle);//On ajoute le rectangle après le cercle
      primaryStage.setVisible(true);
  }
```

Result



Create calculator

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import fr.ubordeaux.miage.s7.calculette.model.Calculator;
import fr.ubordeaux.miage.s7.calculette.model.StackCalculator;
import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;
public class Main extends Application implements EventHandler<ActionEvent> {

  Calculator calculator = new StackCalculator();
   TextField text;

   public static void main(String[] args) {
      launch(args);

   }
   public void handle(ActionEvent event) {
      calculator.pressButton(((Button)event.getSource()).getId());
      text.setText(calculator.getResult());
   }
```

Start method

```
public void start(Stage primaryStage) throws Exception {
   Stage window = primaryStage;
   window.setTitle("Calculette");
   List<Button> buttons = new ArrayList<Button>();
   String[] buttonsArray = {
         "MC", "M+", "M-", "/",
         "7", "8", "9", "x",
         "4", "5", "6", "-",
         "1", "2", "3", "+",
         "0", ".", "="};
   for (String b : buttonsArray)
      buttons.add(new Button(b));

   GridPane grid = new GridPane();
   grid.setAlignment(Pos.CENTER);
   grid.setHgap(4);
   grid.setVgap(4);

   text = new TextField();
   grid.add(text, 0, 0, 4, 1);

   Iterator<Button> iterator = buttons.iterator();
   int lineIndex = 2;
   int columnIndex = 0;
```

Start method

```
    while (iterator.hasNext()) {
        Button button = iterator.next();
        button.setOnAction(this);
        button.setId(button.getText());
        button.setMaxWidth(Double.MAX_VALUE);
        button.setMaxHeight(Double.MAX_VALUE);
        grid.add(button, columnIndex++, lineIndex);
        if (columnIndex==4) {
            lineIndex++;
            columnIndex=0;
        }
    }

    if (calculator.pressButton("0"))
        text.setText(calculator.getResult());

    Scene scene = new Scene(grid);
    window.setScene(scene);
    window.setResizable(false);
    window.show();

}//end of start
```

Generic Calculator

```
public interface Calculator {

   boolean pressButton(String button);
   String getResult();
   void calculate();
   void clear();

}
```

StackCalculator

```
import java.util.Deque;
import java.util.LinkedList;

public class StackCalculator implements Calculator {


  private Deque<Object> stack;
  private Double result;
  private double multiplicator = 10;
  private double divisor = 1;

  public StackCalculator() {
     stack = new LinkedList<Object>();
  }

  private void pushOperator(String op){
     stack.push(op);
     stack.push(new Double(0));
  }

  private void pushNop(){
     stack.push("NOP");
  }
```

StackCalculator

```
private void pushDigit(int digit) {
   double nb;
   if (!stack.isEmpty())
      nb = multiplicator * (Double)(stack.pop());
   else
      nb = 0;
   if (multiplicator == 1)
      divisor = divisor / 10;
   stack.push(new Double(digit * divisor + nb));
}

private void resetDigit() {
   this.multiplicator = 10;
   this.divisor = 1;
}

private boolean topIsNop() {
   return !stack.isEmpty()
         && (stack.getFirst() instanceof String)
         && (((String)stack.getFirst()).equals("NOP"));
}
```

StackCalculator

```
public boolean pressButton(String button) {
   switch (button) {
   case "0" :
   case "1" :
   case "2" :
   case "3" :
   case "4" :
   case "5" :
   case "6" :
   case "7" :
   case "8" :
   case "9" :
      if (topIsNop())
         clear();
      pushDigit(Integer.parseInt(button));
      this.result = (Double)stack.getFirst();
      return true;
```

StackCalculator

```
    case "+" :
    case "-" :
    case "x" :
    case "/" :
       if (topIsNop())
          stack.pop();
       if (stack.size()>=3) {
          this.calculate();
          this.result = (Double)stack.getFirst();
       }
       pushOperator(button);
       resetDigit();return false;
    case "=" :
       if (topIsNop())
          stack.pop();
       if (stack.size()>=3) {
          this.calculate();
          this.result = (Double)stack.getFirst();
       }
       pushNop();
       resetDigit();return true;
    case "MC" :
       clear();
       this.result = (Double)stack.getFirst();
       resetDigit();return true;
    case "." :
       this.multiplicator = 1;return false;
    }
    return false;
} //end of switch
```

StackCalculator

```
    public String getResult() {
        return String.valueOf(this.result);
    }

    public void calculate() {
        if (stack.size()>=3){
            double nb2 = (Double)(stack.pop());
            String op = (String)(stack.pop());
            double nb1 = (Double)(stack.pop());
            switch (op){
            case "+":
                stack.push(new Double(nb1 + nb2));
                break;
            case "-":
                stack.push(new Double(nb1 - nb2));
                break;
            case "x":
                stack.push(new Double(nb1 * nb2));
                break;
            case "/":
                stack.push(new Double(nb1 / nb2));
                break;
            }
        }
    }

    public void clear() {
        stack.clear();
        stack.push(new Double(0));
    }
}// end of class
```

FXML file

Web Application

- Other way to create web application with `javafx` is to create a `FXML`.
- Markup Language (ML) are language to describe data structure in a lot of different domains.
- Main principles : using tags to describe a tree of keywords (tags).
- HTML belongs to this category. XML is the generic ML used to describe new ML by defining new keywords.
- You are not supposed to create or to use by hands data files writing in any ML. These files are huge !
- The FXML file is loaded in the `initRootLayout` method using the class `FXMLLoader`
- To begin consult : https://docs.oracle.com/javafx/2/get_started/fxml_tutorial.htm

FXML example

```
<?xml version="1.0" encoding="UTF-8"?>

<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<AnchorPane id="AnchorPane" prefHeight="400.0" prefWidth="600.0"
                xmlns:fx="http://javafx.com/fxml">
</AnchorPane>
```

```
public final class Main extends Application {

  @Override
  public void start(final Stage primaryStage) {
    try {
      // Localisation du fichier FXML.
      final URL url = getClass().getResource("test.fxml");
      // Création du loader.
      final FXMLLoader fxmlLoader = new FXMLLoader(url);
      // Chargement du FXML.
      final AnchorPane root = (AnchorPane) fxmlLoader.load();
      // Création de la scène.
      final Scene scene = new Scene(root, 300, 250);
      primaryStage.setScene(scene);
    } catch (IOException ex) {
      System.err.println("Erreur au chargement: " + ex);
    }
    primaryStage.setTitle("Test FXML");
    primaryStage.show();
  }

  public static void main(String[] args) {
    launch(args);
  }
}
```

Result