Advanced Topics in Programming

LAB 10 – JAVA FX (CONT.)

Outline

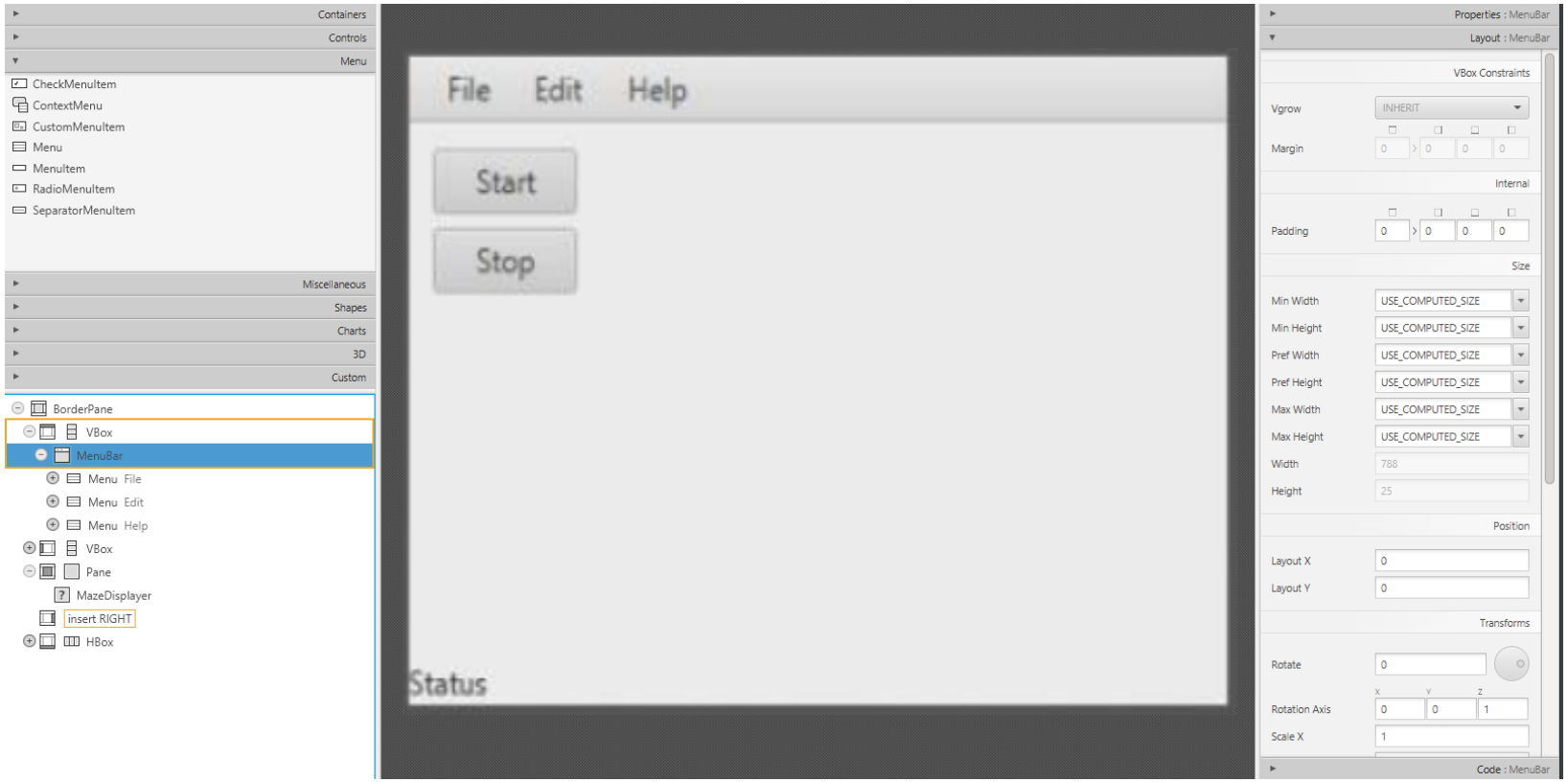
- FXML – Scene Builder
- Observer design pattern
- Continue the code from Lab 9

Java FX Example

JavaFX

```
public class Main extends Application {
 @Override
 public void start(Stage primaryStage) {
  try {
   BorderPane root=(BorderPane)FXMLLoader.load(getClass().getResource("MainWindow.fxml"));
   Scene scene = new Scene(root,400,400);
   scene.getStylesheets().add(getClass().getResource("application.css").toExternalForm());
   primaryStage.setScene(scene);
   primaryStage.show();
  } catch(Exception e) {
   e.printStackTrace();
  }
 }

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

JavaFX - SceneBuilder



https://gluonhq.com/products/scene-builder/

JavaFX – Output to FXML file

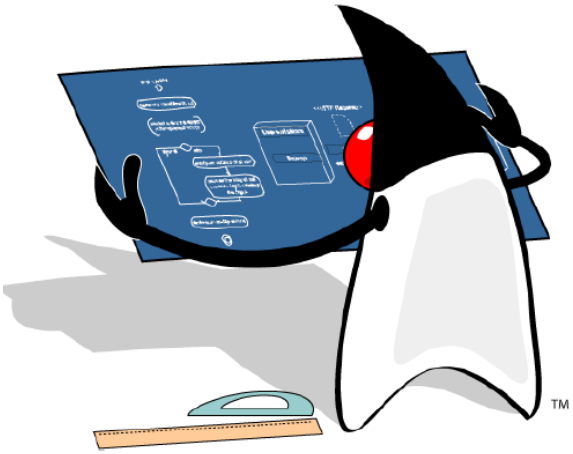
```
<BorderPane prefHeight="395.0" prefWidth="607.0" xmlns="http://javafx.com/javafx/8.0.111"
xmlns:fx="http://javafx.com/fxml/1"
fx:controller="view.MainWindowLogic">
   <top>
      <MenuBar BorderPane.alignment="CENTER">
        <menus>
          <Menu mnemonicParsing="false" text="File">
            <items>
              <MenuItem mnemonicParsing="false" onAction="#openFile" text="open" />
              <MenuItem mnemonicParsing="false" text="Close" />
            </items>
          </Menu>
          <Menu mnemonicParsing="false" text="Edit">
            <items>
              <MenuItem mnemonicParsing="false" text="Delete" />
            </items>
          </Menu>
          <Menu mnemonicParsing="false" text="Help">
            <items>
              <MenuItem mnemonicParsing="false" text="About" />
            </items>
          </Menu>
        </menus>
      </MenuBar>
   </top>
   <left> ...
```

Link to presentation logic

JavaFX – MainWindowLogic

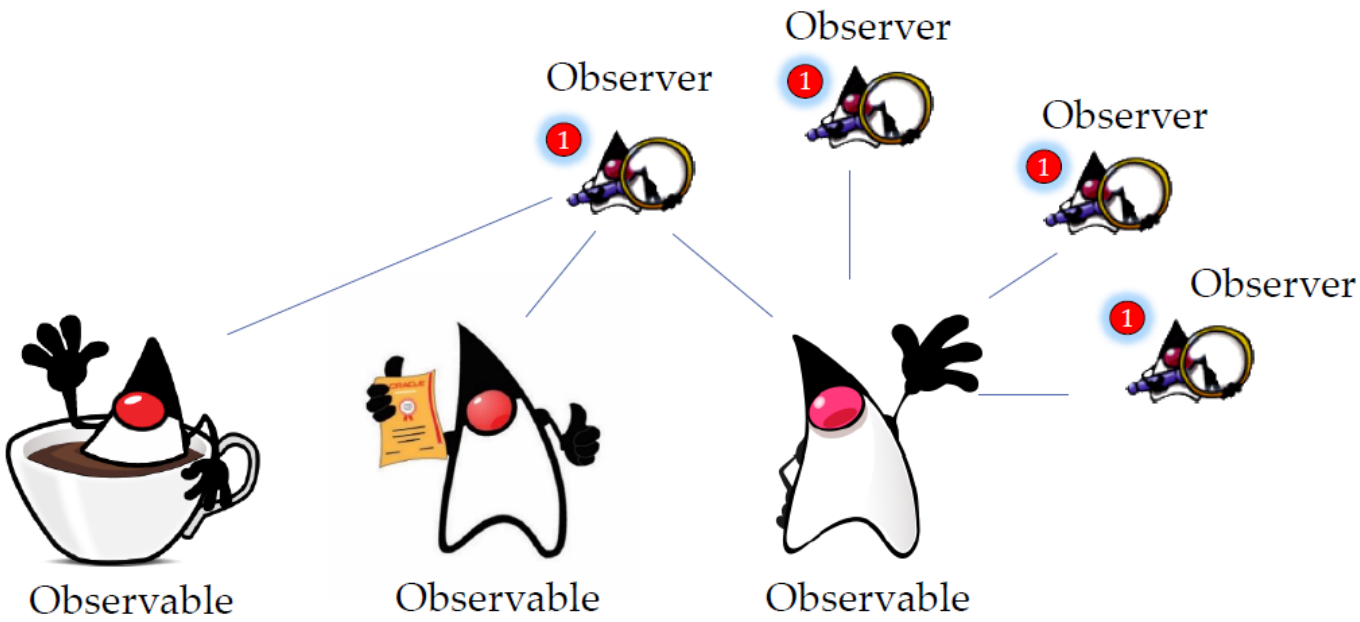
```
public class MainWindowLogic {


 public void openFile(){
   FileChooser fc=new FileChooser();
   fc.setTitle("open level");
   fc.setSelectedExtensionFilter(new ExtensionFilter("*.xml", "*.obj"));
   fc.setInitialDirectory(new File("./resources"));
   File chosen=fc.showOpenDialog(null);
 }
 //...
```

This event handler dependency was injected to the MenuItem object. It was done by JavaFX behind the scenes.



Observer Design Pattern

Observer Design Pattern



Observer Design Pattern

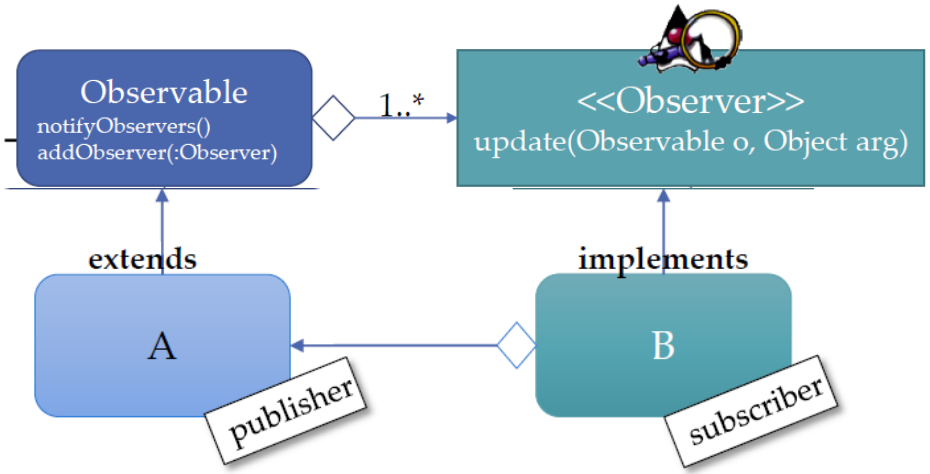
- An ***Observable*** can **notify** many ***Observers***
- An ***Observer*** can **subscribe** to many ***Observables***

```
for(Observer o : observersList){
        o.update(this, args);
}
```

B should subscribe as A's observer

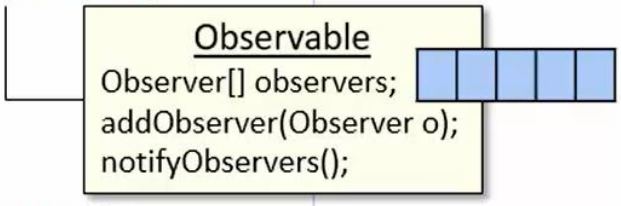
A can update B without knowing B!

If we want, B can tell A what to do



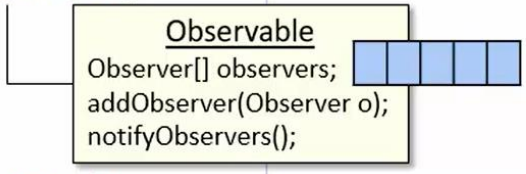
Observer Design Pattern

```
public class A extends Observable{
  int x,y;
  public A() {
    x=0;
    y=0;
  }
  public void setXY(int x,int y){
    this.x=x;
    this.y=y;
      // actively notify all observers
      // and invoke their update method
         notifyObservers();
  }

  public int getX(){return x;}
  public int getY(){return y;}
}
```



Observer Design Pattern

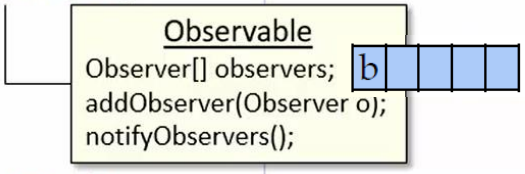
```java
public class A extends Observable{
  int x,y;
  public A() {
    x=0;
    y=0;
  }
  public void setXY(int x,int y){
    this.x=x;
    this.y=y;
      // actively notify all observers
      // and invoke their update method
         notifyObservers();
  }

  public int getX(){return x;}
  public int getY(){return y;}
}
```




```java
public class B implements Observer{
  A a0,a1,a2;
  public B(A a0, A a1, A a2) {
    this.a0=a0; this.a1=a1; this.a2=a2;
  }

  @Override
  public void update(Observable o, Object arg) {
    // this is invoked upon any change to object "a"
    // now we can actively get the state of object "a"
    if(o == a0){
     System.out.println("a change has occurred");
     System.out.println("X="+a0.getX()+"Y="+a0.getY());
    }
  }
}
```

Observer Design Pattern

```java
public class A extends Observable{
  int x,y;
  public A() {
    x=0;
    y=0;
  }
  public void setXY(int x,int y){
    this.x=x;
    this.y=y;
      // actively notify all observers
      // and invoke their update method
        notifyObservers();
  }

  public int getX(){return x;}
  public int getY(){return y;}
}
```




```java
public class B implements Observer{
  A a0,a1,a2;
  public B(A a0, A a1, A a2) {
    this.a0=a0; this.a1=a1; this.a2=a2;
  }

  @Override
  public void update(Observable o, Object arg) {
    // this is invoked upon any change to object "a"
    // now we can actively get the state of object "a"
    if(o == a0){
     System.out.println("a change has occurred");
     System.out.println("X="+a0.getX()+"Y="+a0.getY());
    }
  }
}
```

```java
public static void main(String[] args) {
  A a=new A();
  B b=new B(a,null,null);
  // inherited from Observable
  // add b to the a's list of observers
  a.addObserver(b);

  a.setXY(5,5);
}
```

Lab 10

- Use Scene Builder
- Properties & Biding
- Add Image wall & character on the maze
- https://openjfx.io/openjfx-docs/#javaFX-and-Intellij
  - Go to “JavaFX and Intellij”

Lab 10 Task

- Make the image character move