Cours M2105
Introduction aux Interfaces Homme-Machine

Cours 3 : Widgets et événements (1/2)

Géry Casiez https://cristal.univ-lille.fr/~casiez/
Département informatique - IUT A - Université de Lille

Plan du cours en 9 semaines

1. Introduction à l'interaction, placement
2. Programmation événementielle
3. **Widgets et événements (1/2)**
4. Widgets et événements (2/2)
5. Conception et prototypage (1/2)
6. Conception et prototypage (2/2)
7. Heuristiques et recommandations
8. Modèles et théories
9. Méthodes d'évaluation des IHM

Compteur

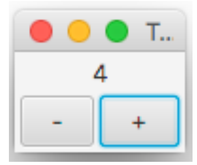
1ère solution : une classe externe par bouton


```
public class TestButtonIncDec extends Application {
    public void start(Stage stage) {
        Label label = new Label("0");
        Button bmoins = new Button("  -  ");
        Button bplus = new Button("  +  ");

        bplus.addEventHandler(ActionEvent.ACTION, new ClicListenerInc(label));
        bmoins.addEventHandler(ActionEvent.ACTION, new ClicListenerDec(label));

        VBox vbox = new VBox(3);
        vbox.setPadding(new Insets(3, 3, 3, 3));
        vbox.setAlignment(Pos.CENTER);
        HBox hbox = new HBox(3);
        hbox.getChildren().addAll(bmoins, bplus);
        vbox.getChildren().addAll(label, hbox);

        Scene scene = new Scene(vbox);
        stage.setTitle("TestButtonIncDec");
        stage.setScene(scene);
        stage.show();
    }

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


```
public class ClicListenerInc implements EventHandler<ActionEvent> {
    Label label;

    ClicListenerInc(Label lbl) {
        label = lbl;
    }

    public void handle(ActionEvent event) {
        int newValue = Integer.parseInt(label.getText()) + 1;
        label.setText("" + newValue);

    }
}
```

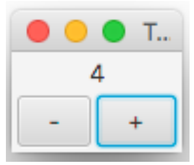
2e solution : une classe interne


```
public class TestButtonIncDec2 extends Application {
    Label label;
    Button bmoins, bplus;

    class ClicListenerIncDec implements EventHandler<ActionEvent> {
        public void handle(ActionEvent event) {
            int currentValue = Integer.parseInt(label.getText());
            if (event.getTarget() == bplus) {
                label.setText("" + (currentValue + 1));
            } else {
                label.setText("" + (currentValue - 1));
            }
        }
    }

    public void start(Stage stage) {
        label = new Label("0");
        bmoins = new Button("  -  ");
        bplus = new Button("  +  ");

        bplus.addEventHandler(ActionEvent.ACTION, new ClicListenerIncDec());
        bmoins.addEventHandler(ActionEvent.ACTION, new ClicListenerIncDec());

        VBox vbox = new VBox(3);
        vbox.setPadding(new Insets(3, 3, 3, 3));
        vbox.setAlignment(Pos.CENTER);
        HBox hbox = new HBox(3);
        hbox.getChildren().addAll(bmoins, bplus);
        vbox.getChildren().addAll(label, hbox);

        Scene scene = new Scene(vbox);
        stage.setTitle("TestButtonIncDec2");
        stage.setScene(scene);
        stage.show();
    }

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



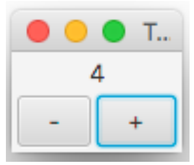
3e solution : deux classes internes anonymes


```
public class TestButtonIncDec3 extends Application {

    public void start(Stage stage) {
        Label label = new Label("0");
        Button bmoins = new Button("  -  ");
        Button bplus = new Button("  +  ");

        bplus.addEventHandler(ActionEvent.ACTION, new EventHandler<ActionEvent>(){
            public void handle(ActionEvent event) {
                int newValue = Integer.parseInt(label.getText()) + 1;
                label.setText("" + newValue);
            }
        });
        bmoins.addEventHandler(ActionEvent.ACTION, new EventHandler<ActionEvent>(){
            public void handle(ActionEvent event) {
                int newValue = Integer.parseInt(label.getText()) - 1;
                label.setText("" + newValue);
            }
        });

        VBox vbox = new VBox(3);
        vbox.setPadding(new Insets(3, 3, 3, 3));
        vbox.setAlignment(Pos.CENTER);
        HBox hbox = new HBox(3);
        hbox.getChildren().addAll(bmoins, bplus);
        vbox.getChildren().addAll(label, hbox);

        Scene scene = new Scene(vbox);
        stage.setTitle("TestButtonIncDec2");
        stage.setScene(scene);
        stage.show();
    }

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



4e solution : implémenter directement EventHandler<ActionEvent>


```
public class TestButtonIncDec4 extends Application implements EventHandler<ActionEvent> {
    Label label;
    Button bmoins, bplus;

    public void start(Stage stage) {
        label = new Label("0");
        bmoins = new Button("  -  ");
        bplus = new Button("  +  ");

        bplus.addEventHandler(ActionEvent.ACTION, this);
        bmoins.addEventHandler(ActionEvent.ACTION, this);

        VBox vbox = new VBox(3);
        vbox.setPadding(new Insets(3, 3, 3, 3));
        vbox.setAlignment(Pos.CENTER);
        HBox hbox = new HBox(3);
        hbox.getChildren().addAll(bmoins, bplus);
        vbox.getChildren().addAll(label, hbox);

        Scene scene = new Scene(vbox);
        stage.setTitle("TestButtonIncDec4");
        stage.setScene(scene);
        stage.show();
    }

    public void handle(ActionEvent event) {
        int currentValue = Integer.parseInt(label.getText());
        if (event.getTarget() == bplus) {
            label.setText("" + (currentValue + 1));
        } else {
            label.setText("" + (currentValue - 1));
        }
    }

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



5e solution : lambda expressions


```
public class TestButtonIncDec5 extends Application {

    public void start(Stage stage) {
        Label label = new Label("0");
        Button bmoins = new Button("  -  ");
        Button bplus = new Button("  +  ");

        bplus.addEventHandler(ActionEvent.ACTION, e -> {
            int currentValue = Integer.parseInt(label.getText());
            label.setText("" + (currentValue + 1));
        });
        bmoins.addEventHandler(ActionEvent.ACTION, e -> {
            int currentValue = Integer.parseInt(label.getText());
            label.setText("" + (currentValue - 1));
        });

        VBox vbox = new VBox(3);
        vbox.setPadding(new Insets(3, 3, 3, 3));
        vbox.setAlignment(Pos.CENTER);
        HBox hbox = new HBox(3);
        hbox.getChildren().addAll(bmoins, bplus);
        vbox.getChildren().addAll(label, hbox);

        Scene scene = new Scene(vbox);
        stage.setTitle("TestButtonIncDec5");
        stage.setScene(scene);
        stage.show();
    }

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

Syntaxe des lambda expressions


```
e -> {
    int currentValue = Integer.parseInt(label.getText());
    label.setText("" + (currentValue - 1));
}
```

```
e -> System.out.println(e.getEventType())
```

Abonneurs / abonnés

Deux catégories d'entités :

- Un abonneur, la source d’information, le lieu où sont gérés les événements
- Un **ou des** abonnés, utilisant l’information produite

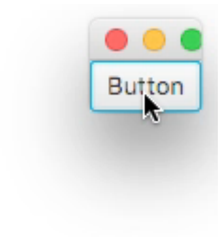
Résumé des différentes étapes :

- Abonnement de l’abonné auprès de l’abonneur
- Un événement se produit au niveau de l’abonneur
- L’abonneur notifie l’ensemble des abonnés en leur transmettant un événement
- Chaque abonné peut traiter l’événement

Illustration


```java
public class PlusieursHandlers extends Application {
    public void start(Stage stage) {
        VBox root = new VBox();
        Button b = new Button("Button");
        b.addEventHandler(ActionEvent.ACTION, e -> System.out.println("Event handler 1"));
        b.addEventHandler(ActionEvent.ACTION, e -> System.out.println("Event handler 2"));
        b.addEventHandler(ActionEvent.ACTION, e -> System.out.println("Event handler 3"));
        root.getChildren().add(b);

        Scene scene = new Scene(root);
        stage.setTitle("Plusieurs handlers");
        stage.setScene(scene);
        stage.show();
    }

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

```
Event handler 1
Event handler 2
Event handler 3
```



Utilisation de setOnXXX

Méthode setOnXXX pour enregistrer **un** event handler sur un noeud

```
public class TestButtonIncDec6 extends Application {

    public void start(Stage stage) {
        Label label = new Label("0");
        Button bmoins = new Button("  -  ");
        Button bplus = new Button("  +  ");

        bplus.setOnAction(e -> {
            int currentValue = Integer.parseInt(label.getText());
            label.setText("" + (currentValue + 1));
        });

        bmoins.setOnAction(e -> {
            int currentValue = Integer.parseInt(label.getText());
            label.setText("" + (currentValue - 1));
        });

        VBox vbox = new VBox(3);
        vbox.setPadding(new Insets(3, 3, 3, 3));
        vbox.setAlignment(Pos.CENTER);
        HBox hbox = new HBox(3);
        hbox.getChildren().addAll(bmoins, bplus);
        vbox.getChildren().addAll(label, hbox);

        Scene scene = new Scene(vbox);
        stage.setTitle("TestButtonIncDec5");
        stage.setScene(scene);
        stage.show();
    }

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

Utilisation de setOnXXX

De la même manière:

setOnMouseClicked

setOnMouseDragged

setOnKeyPressed

setOnTouchPressed

…

Recommandations

Privilégier l'utilisation des méthodes setOnXXX

Peu de code : utilisation des lambda expressions

Ne pas utiliser les classes internes anonymes

Plus de code : classe interne ou implémentation EventHandler

Encore plus de code : classe externe

Toujours privilégier la lisibilité du code