



Interfaces graphiques

martine.gautier@univ-lorraine.fr

▲ Fenêtre(s) + Composants graphiques.

- texte, images, menus, boutons, listes déroulantes, boîtes de dialogue, barres de défilement, etc.

▲ Programmation événementielle

- L'ordre d'exécution des instructions est imprévisible, puisqu'il dépend des actions de l'utilisateur.
- Une action de l'utilisateur engendre un événement, rangé dans une file d'attente.
- Chaque événement de la file est traité dans l'ordre d'arrivée.

▲ API `java.awt`

- Indépendance vis à vis de la machine sur laquelle s'exécute le programme
- Gestionnaires de mise en page

▲ API `java.swing` (à partir de Java 2)

- Ecrite en Java, sans code spécifique à la plateforme
- S'utilise avec `awt` pour la gestion des événements

▲ API `java.javaFX` (à partir de Java 8)

- Refonte totale, destinée à remplacer `Swing`
- Possibilité de décrire l'interface en XML

▲ Mode procédural

- Utilisation de l'API pour créer/manipuler des fenêtres, des boutons, ...
- Application totalement écrite en Java

▲ Mode déclaratif

- Description de l'interface en XML par le biais de SceneBuilder
- Design accessible à un non-programmeur

▲ Le mélange des deux modes est possible.

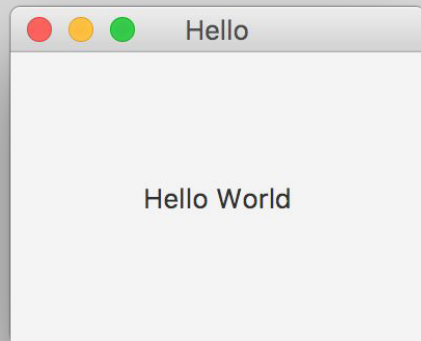


Hello
World

```
public class HelloWorld extends Application {  
    @Override  
    public void start(Stage primaryStage){  
        primaryStage.setTitle("Hello");  
        BorderPane root = new BorderPane();  
        root.setCenter(new Label("Hello World"));  
        primaryStage.setScene(new Scene(root, 300, 275));  
        primaryStage.show();  
    }  
    public static void main(String[] args) {  
        launch(args);  
    }  
}
```

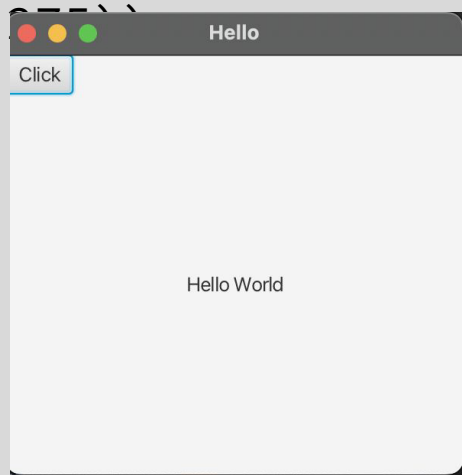
Hello
World

```
public class HelloWorld extends Application {  
    @Override  
    public void start(Stage primaryStage) {  
        primaryStage.setTitle("Hello");  
        BorderPane root = new BorderPane();  
        root.setCenter(new Label("Hello World"));  
        primaryStage.setScene(new Scene(root, 300, 275));  
        primaryStage.show();  
    }  
    public static void main(String[] args) {  
        launch(args);  
    }  
}
```

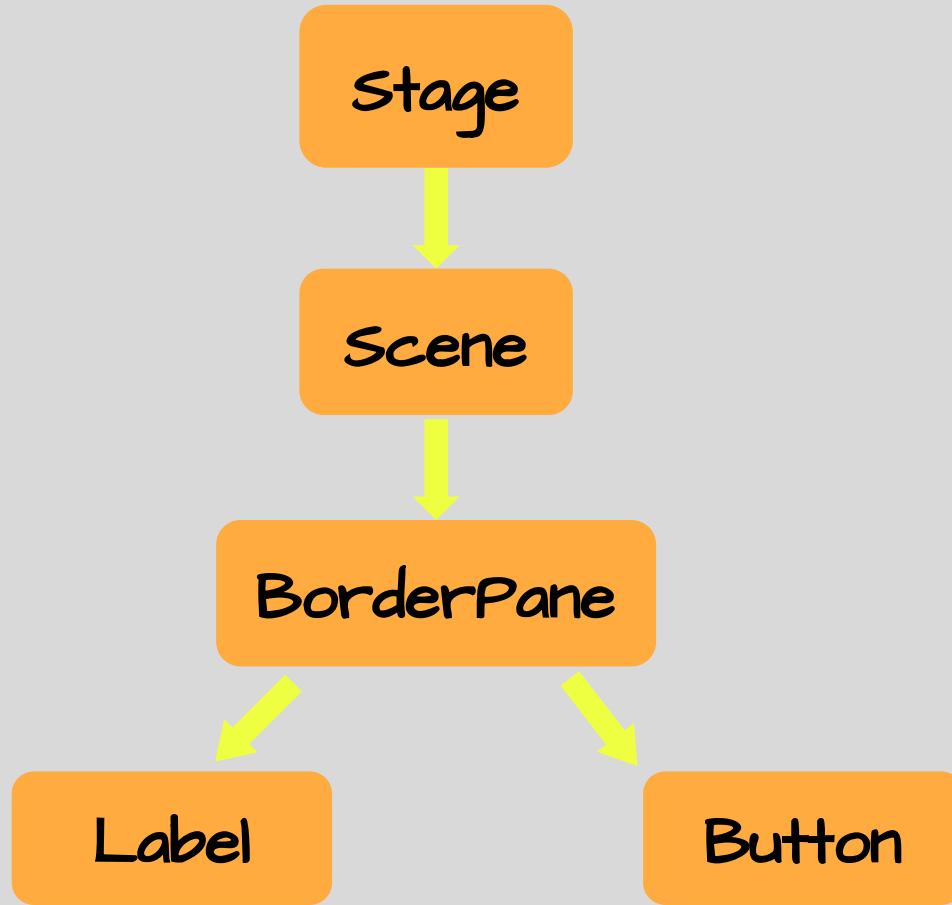


Hello
World

```
public class HelloWorld extends Application {  
    @Override  
    public void start(Stage primaryStage) {  
        primaryStage.setTitle("Hello");  
        BorderPane root = new BorderPane();  
        root.setCenter(new Label("Hello World"));  
        root.setTop(new Button("Click !"));  
        primaryStage.setScene(new Scene(root, 300,  
            primaryStage.show());  
    }  
    public static void main(String[] args) {  
        launch(args);  
    }  
}
```



Arbores-
cence des
composants



- ▲ Point d'entrée : sous-classe de `Application`
- ▲ Lancement par l'intermédiaire de `Application.launch`
 1. Appel de `init()`
 2. Appel de `start()`, avec la fenêtre principale en paramètre
 3. Sommeil, en attendant que l'application se termine
 - Fermeture de la dernière fenêtre
 - Exécution de `Platform.exit()`
 4. Appel de `stop()`

```
public class AppliGraphique extends Application {  
    @Override  
    public void init(){  
        ...  
    }  
    @Override  
    public void start(Stage primaryStage){  
        ...  
    }  
    @Override  
    public void stop() {  
        ...  
    }  
    public static void main(String[] args) { launch(args); }  
}
```

Paramètres d'appel

```
java hello.HelloWorld --texte="Bonjour le monde"
```

```
public void start(Stage primaryStage){  
    primaryStage.setTitle("Hello");  
    BorderPane root = new BorderPane();  
    Parameters param = getParameters() ;  
    String texte = param.getNamed().get("texte");  
    root.setCenter(new Label(texte));  
    ...  
}
```

Réagir à une action

- ▲ Les composants sont réactifs.
- ▲ Une action de l'utilisateur engendre la création d'un événement.
- ▲ La gestion de la file d'attente des événements est réalisée par un thread spécifique.
- ▲ Programmer le traitement de l'événement.

Réagir à une action

- ▲ Le traitement est réalisé par un écouteur.
- ▲ Un écouteur est un objet attaché au composant lors de sa création ; il sera averti lorsque le composant sera source d'un événement.
- ▲ De multiples sortes d'événements, donc d'écouteurs

Attacher l'
écouteur
au bouton

@Override

```
public void start(Stage primaryStage) {  
    primaryStage.setTitle("Hello");  
    BorderPane root = new BorderPane();  
    Button bouton = new Button("Click !");  
    root.setCenter(bouton);  
    bouton.setOnAction(new Ecouteur()) ;  
    ...  
}
```

```
import javafx.event.EventHandler;

public class Ecouteur implements EventHandler<ActionEvent> {

    public Ecouteur () { }

    @Override
    public void handle(ActionEvent event) {
        System.out.println("Click click");
    }

}
```

Transmettre
des données à
l'écouteur

```
@Override  
public void start(Stage primaryStage) {  
    Label label = new Label(" ") ;  
    Button bouton = new Button("Click !");  
    bouton.setOnAction(new Ecouteur(label)) ;  
    ...  
}
```


Ecouteur avec données

```
import javafx.event.EventHandler;

public class Ecouteur implements EventHandler<ActionEvent> {

    private Label label ;

    public Ecouteur (Label label) { this.label = label ; }

    public void handle(ActionEvent event) {

        label.setText("Bonjour !") ;           // Provoque le rafraichissement

    }

}
```

Alternative
à
bannir

```
public class AppliGraphique extends Application
    implements EventHandler<ActionEvent>{

    public void start(Stage primaryStage) {
        Label label = new Label(" ");
        Button bouton = new Button("Click !");
        bouton.setOnAction(this) ;
        ...
    }

    void handle(ActionEvent event) {
        label.setText("Bonjour !") ;
    }
}
```

Attacher l'
écouteur au
bouton,
version
courte ...

@Override

```
public void start(Stage primaryStage) {  
    primaryStage.setTitle("Hello");  
    BorderPane root = new BorderPane();  
    Button bouton = new Button("Click !");  
    root.setCenter(bouton);  
    bouton.setOnAction(event->System.out.println("Bonjour !")) ;  
    ...  
}
```

IntelliJ

