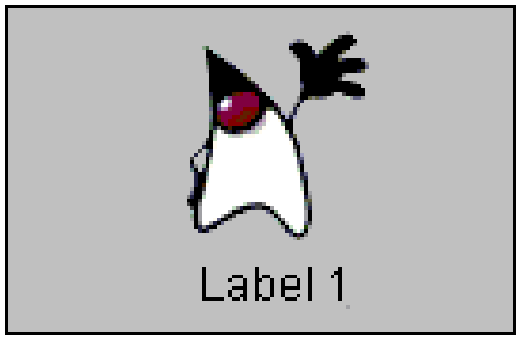
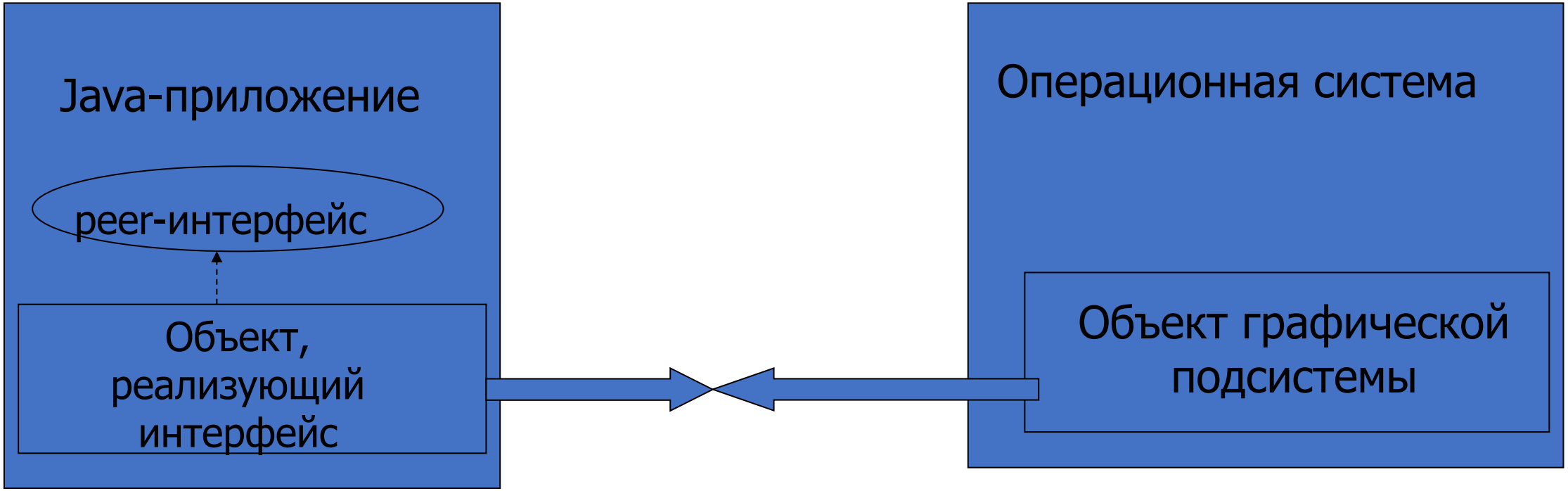
Средства создания графического интерфейса в Java



Библиотека AWT

- peer-to-peer интерфейсы («тяжелые» компоненты)



«Легкие» (lightweight) компоненты

- Сохранение заданного при создании вида (look and feel)
- Возможность изменить вид в любой момент работы приложения (PL&F)
- Библиотека «легких» компонентов Swing

JavaFX

- 2007г. – альтернатива Flash (Sun Microsystems)
- JavaFX 8 вошла в JDK(Oracle)
- Использует язык разметки FXML
- Начиная с Java 11 версии, The Java FX не входит в JavaSE (компания Gluon)

- Ресурсы

  https://openjfx.io/
  https://gluonhq.com/products/javafx/
  https://habr.com/ru/articles/474292/

Приложение JavaFX

```
public class Main extends Application{
   public static void main(String[] args) {
                launch(args);
}
 @Override
  public void start(Stage stage)  {
              Label text = new Label("Hello !");
              Pane root  new Pane(text);
               Scene scene = new Scene(root,300,250);
               stage.setScene(scene);
               stage.setTitle("JavaFX Application");
              stage.show();
  }
}
```

Подключаемые классы

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.layout.Pane;
import javafx.stage.Stage;
```

javafx.application.Application

- init(): инициализирует приложение до его запуска. Метод не должен использоваться для создания графического интерфейса или отдельных его частей.
- start(Stage stage): здесь определяется графический интерфейс.
- stop(): вызывается после закрытия приложения

Потоки

- Ни метод init, ни конструктор класса, который наследуется от Application, не подходит и не должен использоваться для создания графического интерфейса. Потому что и метод init, и конструктор запускаются в потоке, который называется потоком запуска или launcher thread.
- А создание и изменение графического интерфейса должно производиться в потоке приложения или application thread. Именно в таком потоке и запускается метод start (и метод stop).

Потоки выполнения

```
public static void main(String[] args) {
    System.out.println("Метод main()" + " " +
        Thread.currentThread().getName());
    Application.launch(args);
  }
```

```
@Override
  public void init() throws Exception {
    System.out.println("Метод init()" + " " +
        Thread.currentThread().getName());
  }


@Override
  public void stop() throws Exception {
    System.out.println("Метод stop()" + " " +
        Thread.currentThread().getName());
  }
```

```
@Override
  public void start(Stage stage) throws Exception {

    System.out.println("Метод start()" + " " +
        Thread.currentThread().getName());

. . .
}
```

Структура приложения

- Каждое JavaFX приложение состоит из иерархии нескольких основных компонентов:
  - Stage (окна или подмостки)
  - Scene (сцены)
  - Node (узлы)

Stage

Stage -> Window->Object

Конструкторы

Stage()

Stage(StageStyle s)

Stage

В JavaFX application создается первичное окно, которое передается в метод start()

public void start(Stage stage) . . .

При закрытии первичного окна вызывается метод stop() и приложение завершает работу

Программным путем можно создавать дополнительные окна

Окно по нажатию кнопки

```
// вносим изменения в метод start

Pane root= new Pane();
Button bt = new Button(“Create Window”);
root.getChildren().add(btn);

// добавляем обработку события «нажатие кнопки»
btn.setOnAction(event-> {
        newWind();
});
```

Окно по нажатию кнопки

```
// добавляем метод  создания окна, вызываемый в обработчике


public void newWind() {
        Stage wind = new Stage();
        wind.setTitle(“New window”);
        wind.show();
}
```

Стили

StageStyle.DECORATED
StageStyle.UNDECORATED
StageStyle.TRANSPARENT
StageStyle.UTILITY

```
Stage w = new Stage(StageStyle.DECORATED);
```

```
Stage w = new Stage();
w.initStyle(StageStyle.UTILITY);
```

Управление

https://openjfx.io/javadoc/18/javafx.graphics/javafx/stage/Stage.html

По умолчанию открывается документация для 21

Модальность

Modality.NONE

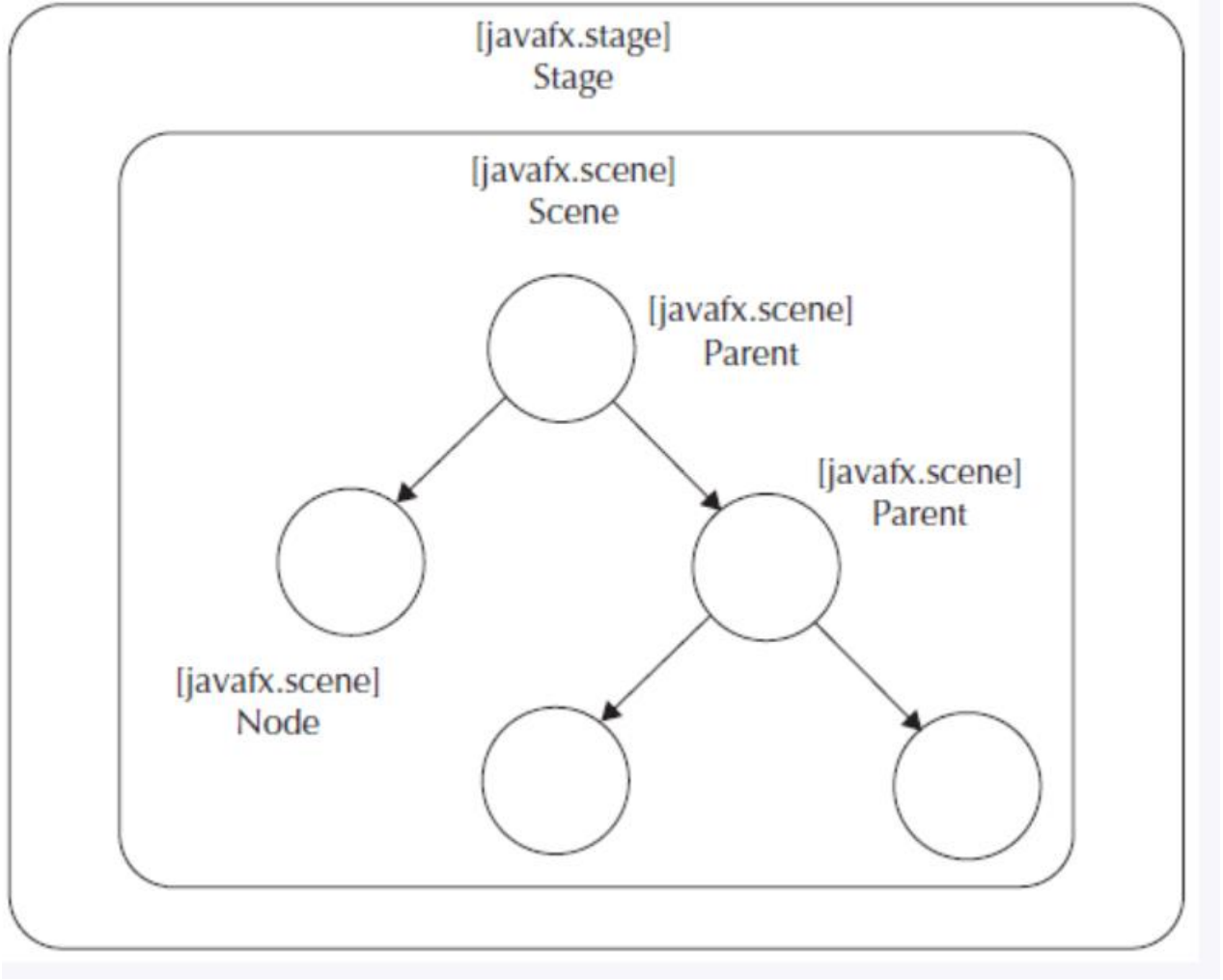
Modality.WINDOW_MODAL

Modality.APPLICATION_MODAL

```
Stage s = new Stage();
s.initModality(Modality. WINDOW_MODAL);
s.initOwner(parentStage);
```

Scene

- javafx.scene.Parent
- javafx.scene.Node



- https://openjfx.io/javadoc/18/javafx.graphics/javafx/scene/package-summary.html

Scene

Scene(Parent root)

Scene(Parent root, double width, double height)

Scene(Parent root, Paint fill)

Scene(Parent root, double width, double height, Paint fill)

Несколько примеров

```
@Override
    public void start(Stage stage) {
        Group root = new Group();
        Scene scene = new Scene(root, 400, 150, Color.BLUE);
        //scene.setFill(Color.BLUE); альтернативная установка цвета
        stage.setScene(scene);
         stage.setTitle("Hello JavaFX");
         stage.show();
    }
```

```
@Override
    public void start(Stage stage) {
        Label label = new Label("Hello");                    // текстовая метка
        Button button = new Button("Button");                // кнопка
        Group group = new Group(button);                     // вложенный узел Group
        FlowPane root = new FlowPane(label, group);          // корневой узел
        Scene scene = new Scene(root, 300, 150);             // создание Scene
        stage.setScene(scene);                               // установка Scene для Stage
        stage.setTitle("Hello JavaFX");
        stage.show();
    }
```

```
@Override
public void start(Stage stage) {
        Group root = new Group();
        Scene scene = new Scene(root, 200, 150);
        scene.setFill(Color.LIGHTGRAY);
        Circle circle = new Circle(60, 40, 30, Color.GREEN);
        Text text = new Text(10, 90, "JavaFX Scene");
        text.setFill(Color.DARKRED);
        Font font = new Font(20);
        text.setFont(font);
        root.getChildren().add(circle);
        root.getChildren().add(text);
        stage.setScene(scene);
        stage.show();
    }
```