Wojskowa Akademia Techniczna

WYDZIAŁ ELEKTRONIKI
WOJSKOWA AKADEMIA TECHNICZNA

Zaawansowane programowanie w języku Java

Laboratorium 3

Interfejsy i wyrażenia lambda.

2016

LABORATORIUM 3

Zagadnienia:

- ✓ Interfejsy
- ✓ Wyrażenia lambda
- ✓ Java FX – eventy
- ✓ Java FX – Scene Builder

PUNKTY:

1 pkt – obecność
2 pkt - Aplikacja 01 - panel logowania
1 pkt - Aplikacja 02 - Scene Builder - pierwsza aplikacja
1 pkt - Aplikacja 03 - Scene Builder - eventy
1 pkt - Aplikacja 04 - Scene Builder - zaawansowany interfejs

Java – interfejsy

Interfejs (protokół) – w programowaniu obiektowym definicja metod, których obiekt używa **do komunikacji** z innym obiektem.

```
public interface MojInterfejs {
    int dodaj();

    int odejmij();

    int pomnoz(int liczba);
}
```

Java – interfejsy - przykłady

```
@FunctionalInterface
public interface Runnable {
    /**
     * When an object implementing interface <code>Runnable</code> is used
     * to create a thread, starting the thread causes the object's
     * <code>run</code> method to be called in that separately executing
     * thread.
     * <p>
     * The general contract of the method <code>run</code> is that it may
     * take any action whatsoever.
     *
     * @see     java.lang.Thread#run()
     */
    public abstract void run();
}
```

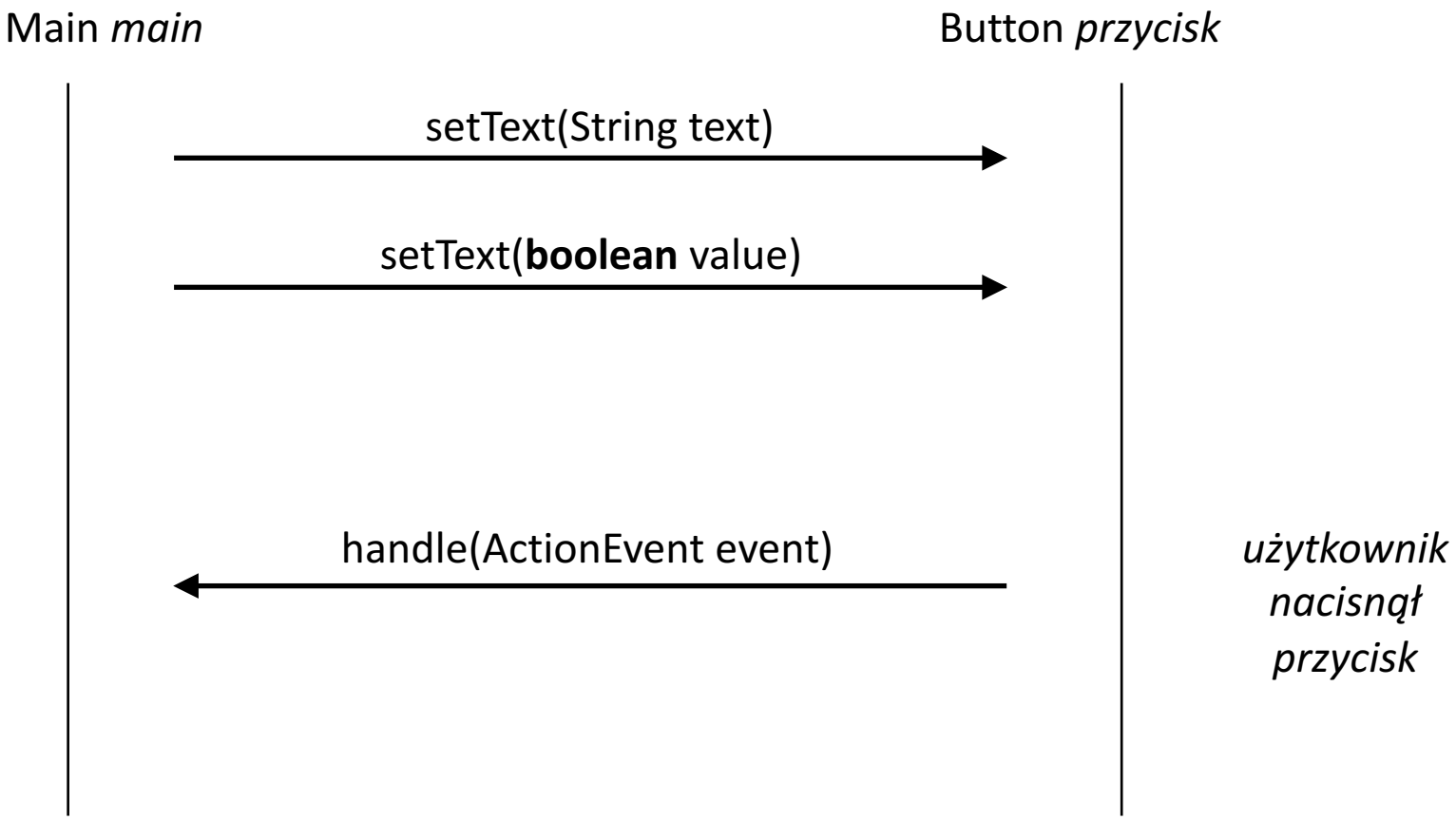
Java – interfejsy - przykłady

```
@FunctionalInterface
public interface EventHandler<T extends Event> extends EventListener {
    /**
     * Invoked when a specific event of the type for which this handler is
     * registered happens.
     *
     * @param event the event which occurred
     */
    void handle(T event);
}
```

Java – interfejsy – przykład komunikacji

```
public class Main extends Application {

        (...)

        public void start(Stage primaryStage) {

                (...)

                Button przycisk = new Button();
                przycisk.setText("ok");
                przycisk.setDefaultButton(true);

                (...)
        }
}
```

Java – interfejsy - przykład komunikacji



Java – interfejsy - implementacja

Opcja 1 – Klasa implementująca

a) Utworzyć klasę, która implementuje interfejs

```
public class ImplementationOfHandler implements EventHandler<ActionEvent> {
    @Override
    public void handle(ActionEvent event) {
        System.out.print(event.getEventType().getName());
    }
}
```

b) Utworzyć obiekt tej klasy

```
button.setOnAction(new ImplementationOfHandler());
```

Java – interfejsy - implementacja

Opcja 2 – Anonymouse Classes

```
button.setOnAction(new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent event) {
        System.out.print(event.getEventType().getName());
    }
});
```

Java – interfejsy - implementacja

Opcja 3 – Implementacja w danej klasie

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

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

    @Override
    public void start(Stage primaryStage) throws Exception {
        Button button = new Button( text: "ok");

        button.setOnAction(this);

        VBox vBox = new VBox();
        vBox.getChildren().addAll(button);

        Scene scene = new Scene(vBox,  width: 300,  height: 300);
        primaryStage.setScene(scene);

        primaryStage.show();
    }

    @Override
    public void handle(ActionEvent event) {
        System.out.print(event.getEventType().getName());
    }
}
```

Java – interfejsy - implementacja

Opcja 4* – Wyrażenia lambda

```
button.setOnAction((ActionEvent event) -> {
        System.out.print(event.getEventType().getName());
});
```

** tylko dla interfejsów funkcyjnych (Functional Interface)*

Java – wyrażenia lambda

```
button.setOnAction(new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent event) {
        System.out.print(event.getEventType().getName());
    }
});
```

==

```
button.setOnAction((ActionEvent event) -> {
        System.out.print(event.getEventType().getName());
});
```

==

```
button.setOnAction((ActionEvent event) -> System.out.print(event.getEventType().getName()));
```

==

```
button.setOnAction(event -> System.out.print(event.getEventType().getName()));
```

JavaFX - literatura

https://docs.oracle.com/javase/tutorial/java/concepts/interface.html
https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html
http://docs.oracle.com/javase/8/scene-builder-2/user-guide/intro.htm#sthref6