Building GUIs

Definition: Stage

The window, along with decorations like a menu bar and window controls

Definition: Scene

The area inside the window in which to put the content

1 Creating a stage

```
public void initialiseGUI1(){
    Stage stage = new Stage();
    stage.setTitle("Hello World");
    stage.show();
}
```

2 Launching the GUI

- Usually just extending the application class, but it is slightly different in Bluej

```
public void launchFX(){
    new JFXPanel();
    Platform.setImplicitExit(false);
    Platform.runLater(() -> initialiseGUI1());
}
```

2.1 Lambda expressions

- They were added to Java 8 for defining anonymous methods
- e.g

```
() -> initialiseGUI1()
```

3 Adding content to the scene

```
public void initialiseGUI2(){
    Stage stage = new Stage();
    stage.setTitle("Hello World");
    Button btn = new Button();
    btn.setText("Say 'hello world'");
    StackPane root = new StackPane();
    root.getChildren().add(btn);
    stage.setScene(new Scene(root, 300, 250));
    stage.show();
}
```

4 Event Handling

- Events correspond to use interactions with components
- Clicking on a button causes an ActionEvent
- An object implements the EventHandler interface
  - Defines a handle method
- It registers as a handler with setOnAction

5 Nested class syntax

Class definitions may be nested

```
public class enclosing
{
    private class Innner
    {
        ...
    }
}
```

5.1 Inner classes

- Instances of the inner class are localised within the enclosing class
- Instances of the inner class have access to the private members of the enclosing class

5.1.1 Anonymous inner classes

- Obey the rules of inner classes
- Used to create one-off objects for which a class name is not required
- Use a special syntax
- The instance is always referenced by its supertype, as it has no subtype name

6 Anonymous event handler

```
btn.setOnAction(new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent event){
        System.out.println("Hello World");
    }
});
```

- Creates object with new
- Overrides methods
- Can be used with interfaces (only way to use interfaces with new)
- @Override annotation is checked by compiler