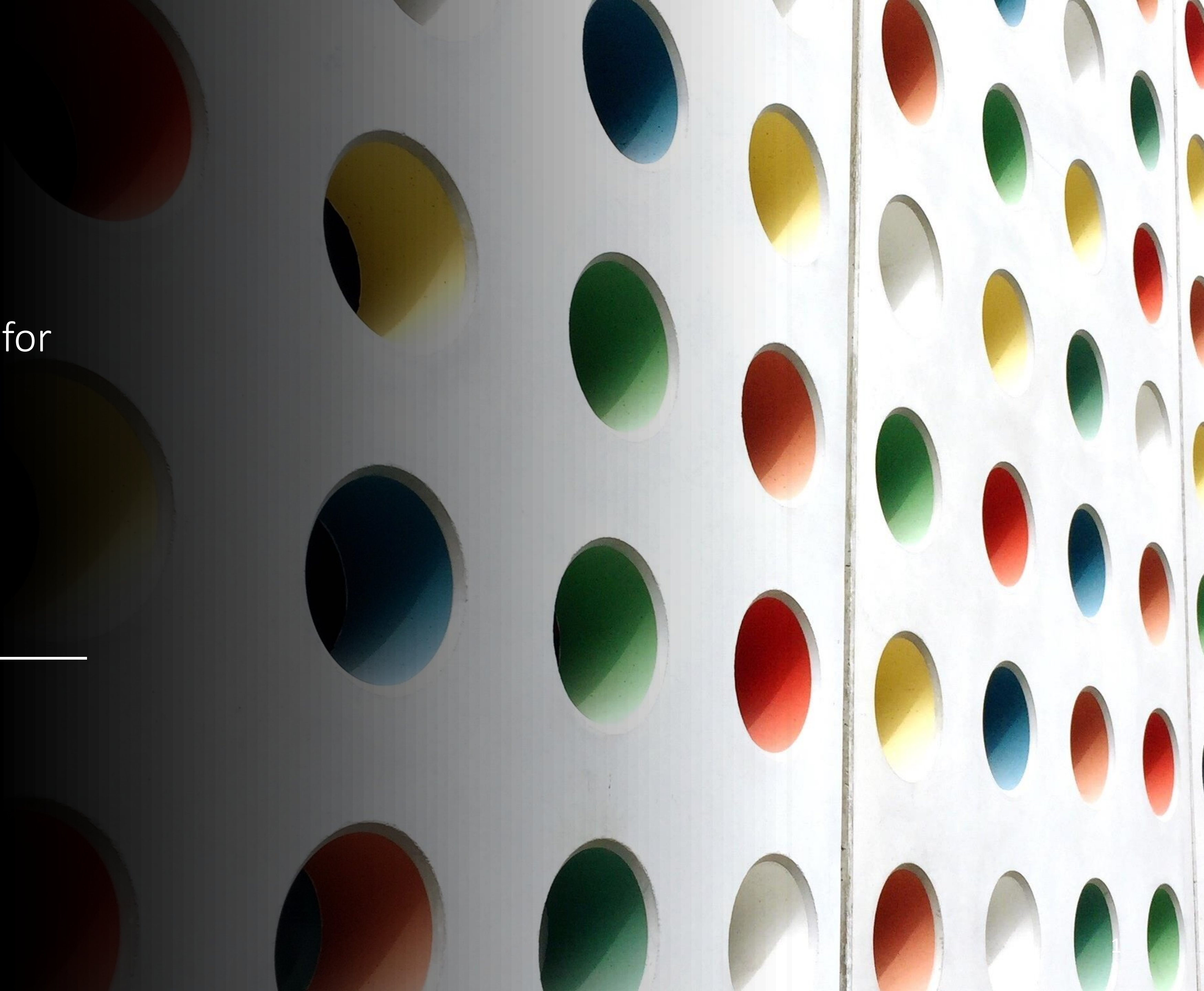
COMP8710 Advanced Java for Programmers

Lecture 17
More JavaFX (2)

Yang He



Topics

- JavaFX Introduction
- JavaFX in IntelliJ IDEA
- Introduction to Apache Maven
- Interacting with users
- Managing events
- Properties & bindings
- Model-View-Controller (MVC)
- Concurrency in JavaFX
- Set A2

Event programming

- Event programming: code is executed when an event is received
- ... vs procedural programming: code execution follows statement/method order
- E.g. OS monitors all sorts of events (keystrokes, mouse clicks etc.) and dispatches them to appropriate applications
- For each control (button etc.), define an event handler and tell the control to invoke handler whenever event is received

Events in JavaFX

- Events are notifications that something has happened
  - As a user clicks a button, presses a key, moves a mouse, or performs other actions, events are dispatched
  - Registered event filters and event handlers within the application receive the event and provide a response

More information at https://docs.oracle.com/javafx/2/events/jfxpub-events.htm

Event listeners (1)

- An EventHandler<T> is an object that handles events of type T:

```
@FunctionalInterface
interface EventHandler<T> {
   void handle(T event);
}
```

- They can be registered on nodes that generate events, e.g.

```
var btn = new Button("Click me");
btn.setOnAction(event -> { ... });

var s = new Scrollbar();
s.setOnScroll(myScrollEventHandler);
```

Event listeners (2)

- Other examples

```
image.setOnMouseEntered(...);

image.setOnMouseExited(...);

textBox.setOnKeyPressed(...);

textBox.setOnKeyReleased(...);
```

Event handlers

- Different ways of creating event handlers:

a)
```
btn.setOnAction(new EventHandler<ActionEvent>() {

  @Override
  public void handle(ActionEvent event) {
        System.out.println("Hello");
    }
});
```
Anonymous function

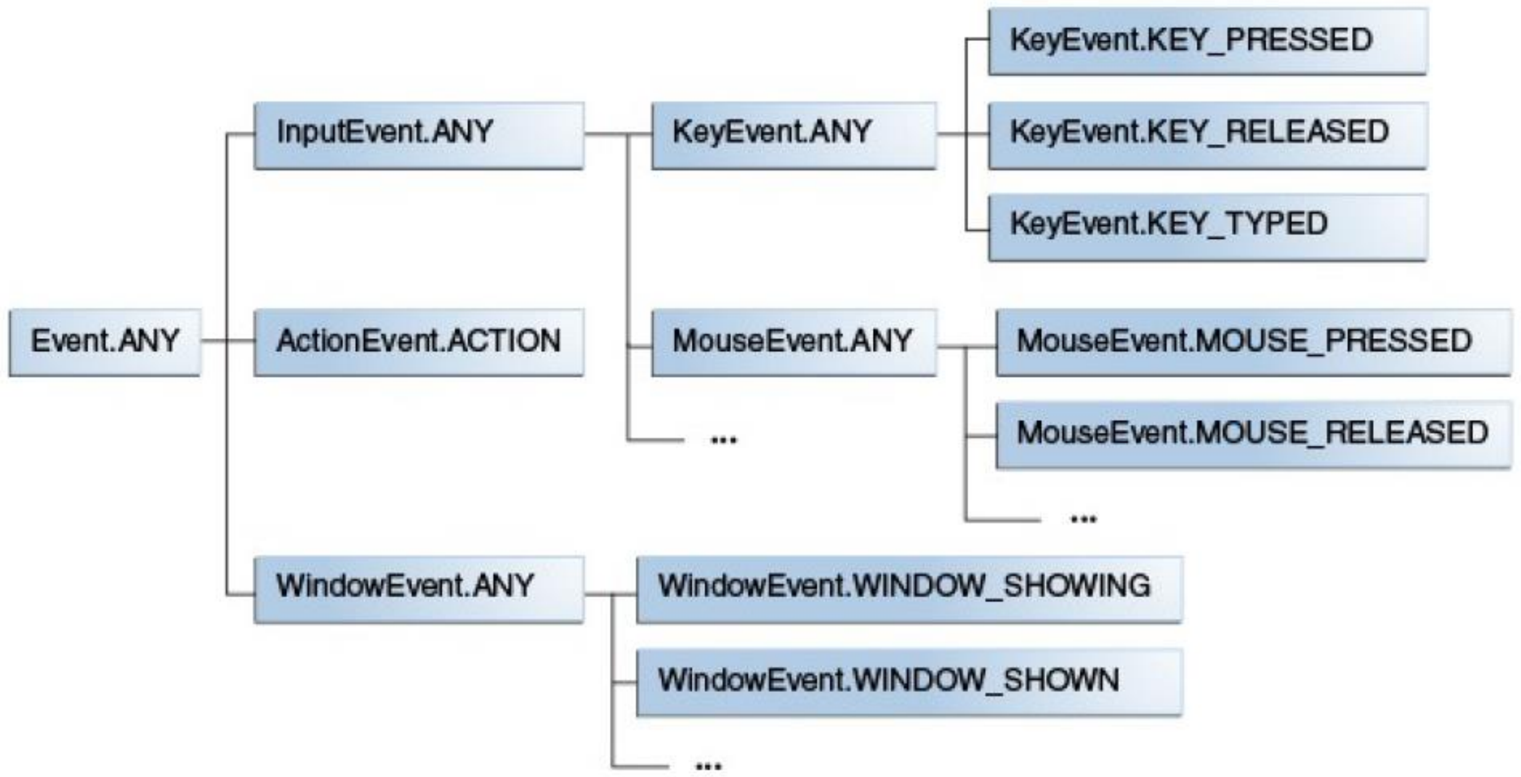
b)
```
btn.setOnAction(e -> System.out.println("Hello"));
```
Lambda expression

c)
```
btn.setOnAction(this::handleButton);

void handleButton(ActionEvent event) {
    System.out.println("Hello");
}
```
Method reference

Events: hierarchy

- Events notify your application of actions taken by the user
  - They enable the application to respond to the event
  - Represented by event
    - DragEvent
    - KeyEvent
    - MouseEvent
    - ScrollEvent
    - …



Events: properties (1)

- Every event must extend `javafx.event.Event` and has the following attributes:
  - Event type
    - e.g. `MouseEvent, KeyEvent`
  - Source
    - The object, i.e. a Node, on which a handler has been registered and which sent the event to it
  - Target
    - Node on which the action occurred and the end node in the event dispatch chain

Events: properties (2)

- E.g.

```
btn.setOnMouseClicked(e -> {
    System.out.println("Event Type:"+ e.getEventType());
    System.out.println("Source: "+ e.getSource());
});
```

Properties & Bindings (1)

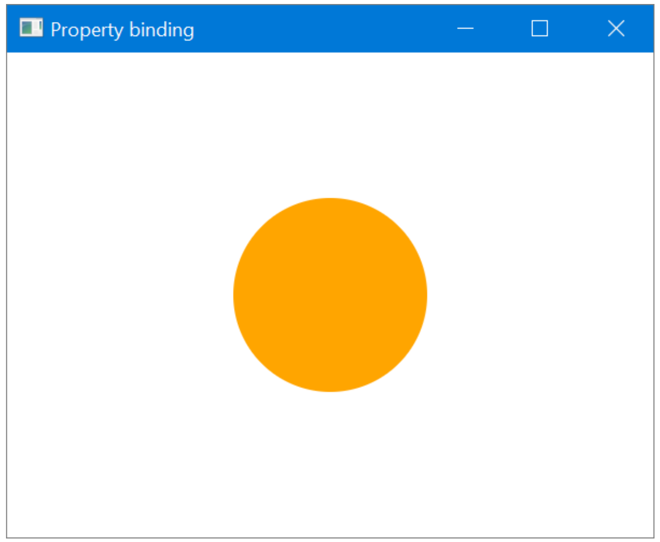
- Most UI nodes have Properties
- Properties are value wrappers that
  - Can be listened on, or
  - Be bound to other properties
- They implement the `Observable` interface

Properties & Bindings (2)

- E.g. Binding the circle’s centre and radius to its container:

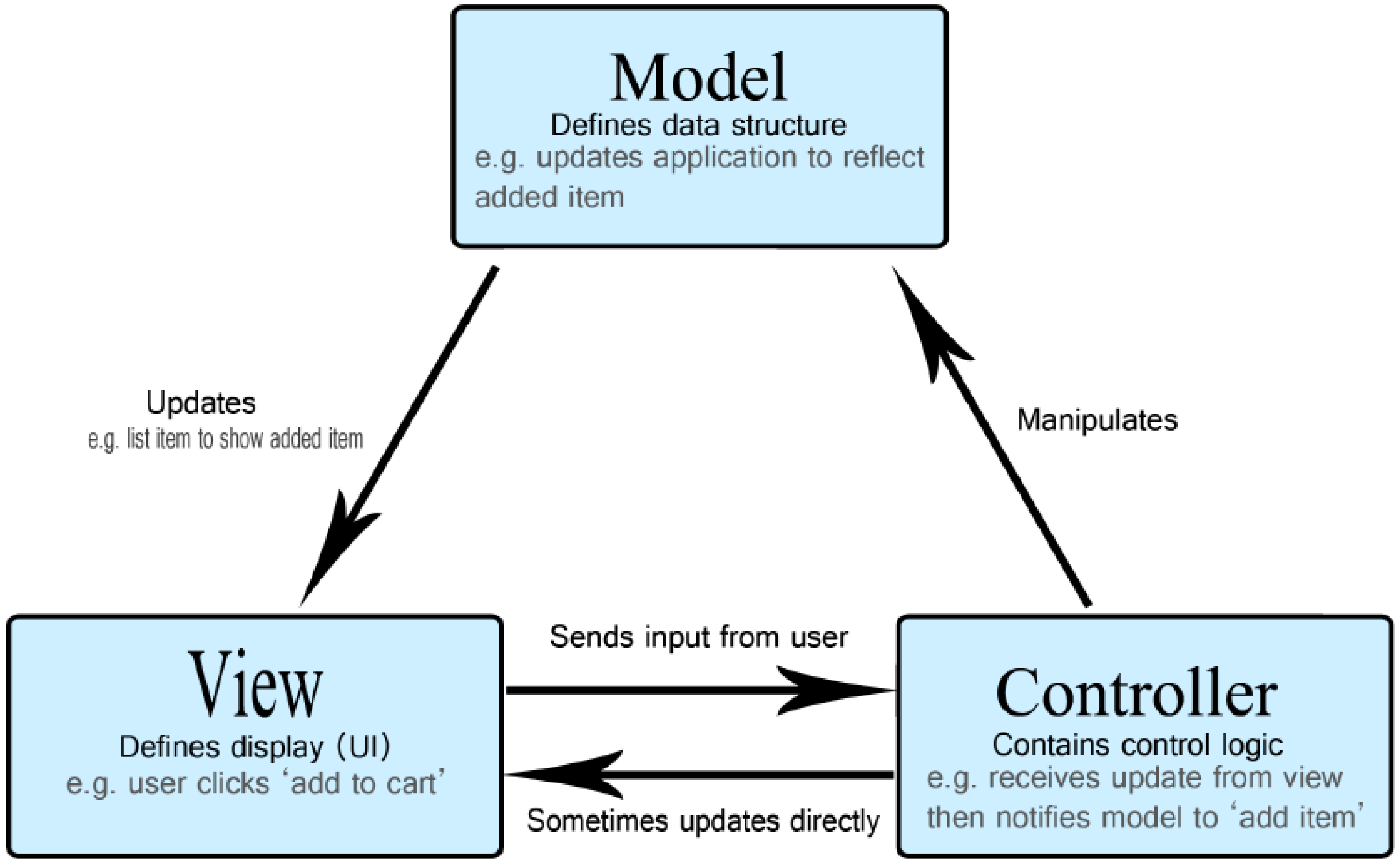
```
var widthProp = container.widthProperty();
var heightProp = container.heightProperty();
var minProp = widthProp.get() < heightProp.get() ? widthProp : heightProp;

circle.radiusProperty().bind(minProp.divide(5));
circle.centerXProperty().bind(widthProp.divide(2));
circle.centerYProperty().bind(heightProp.divide(2));
```



Model-View-Controller

- Source: Mozilla



Model
Defines data structure
e.g. updates application to reflect added item
Updates
e.g. list item to show added item
Manipulates
View
Defines display (UI)
e.g. user clicks 'add to cart'
Sends input from user
Sometimes updates directly
Controller
Contains control logic
e.g. receives update from view then notifies model to 'add item'

Model-View-Controller (3)

- JavaFX enables you to design with MVC using FXML and Java
  - Model consists of application-specific domain objects
  - View consists of FXML
    - We can use the graphical Scene Builder to generate our XML
  - Controller is Java code that defines the GUI's behaviour for interacting with the user

Source: https://docs.oracle.com/javafx/2/best_practices/jfxpub-best_practices.htm

Demo:
Simple JavaFX MVC example

`SimpleBankMVC`

A simple bank (1)

- The Account class

```
public class Account {
   private static int count = 0;
   private String id;
   private final String name;
   private final int balance;

   public Account(String name, int balance){
      count++;
      id = "Account" + count;
      this.name = name;
      this.balance = balance;
   }

   public String getID() { return id; }

   public String getName() { return name; }

   public int getBalance() { return balance; }

   @Override
   public String toString() {
      return getID() + ": " + getName() + " (" +
            getBalance() + ")";
   }
}
```

A simple bank (2)

- The `Main` class:

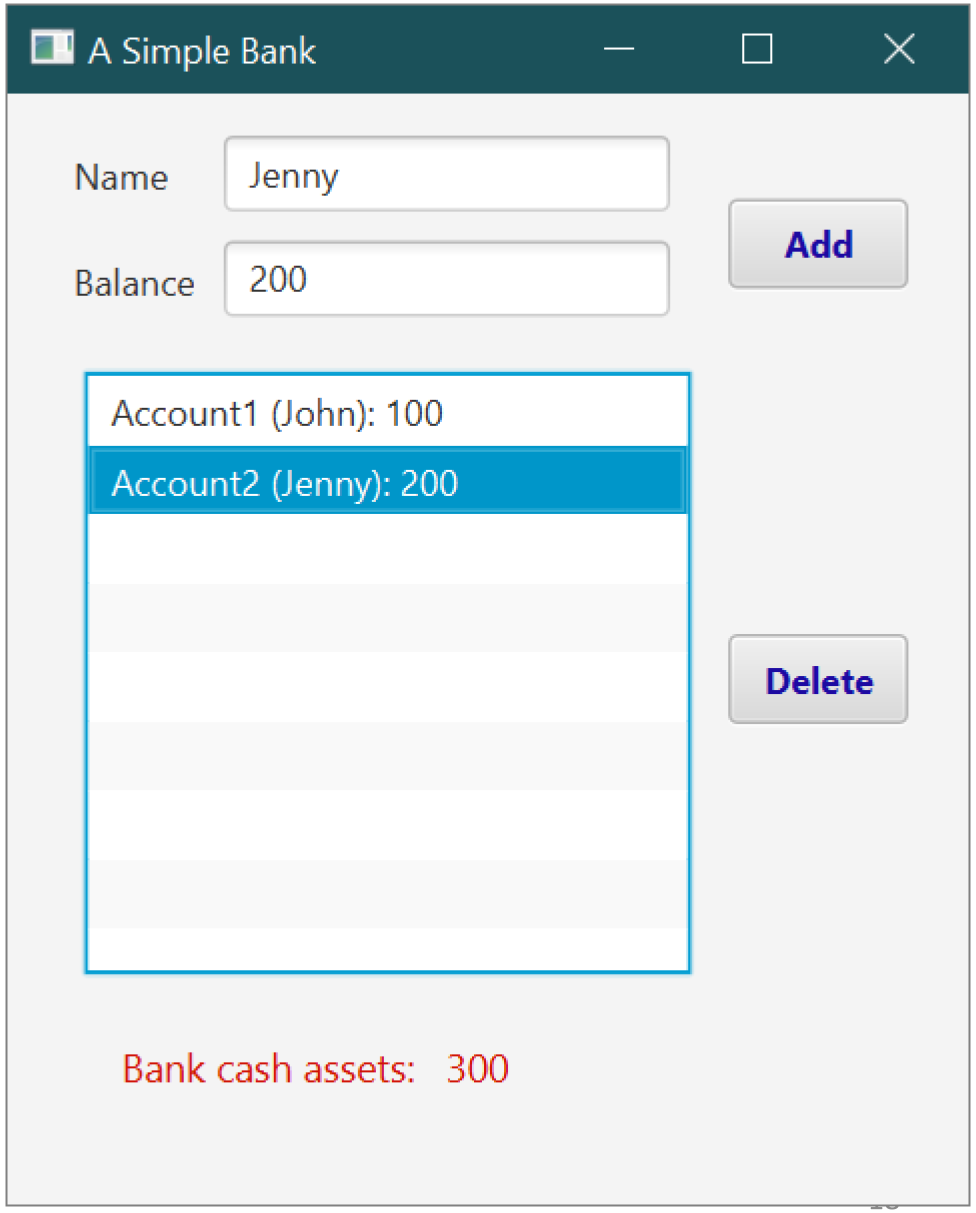
```
public class Main extends Application {
   @Override
   public void start(Stage primaryStage) throws IOException {
      Controller controller = new Controller();
      controller.show(primaryStage);
   }

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

- The “Model” is defined as `Model.java`
- The “View” is defined as GUI.fxml
- The “Controller” is defined as `Controller.java`

A simple bank (3)

- `GUI.fxml`:
  - Two `TextFields` for user inputs: name and balance
  - A `Button` to add a new account
  - A ListView for all accounts
  - A `Button` to delete a selected account (enabled when selecting an account for deletion)
  - A `Label` to display bank assets



Name: Jenny
Balance: 200

Account1 (John): 100
Account2 (Jenny): 200

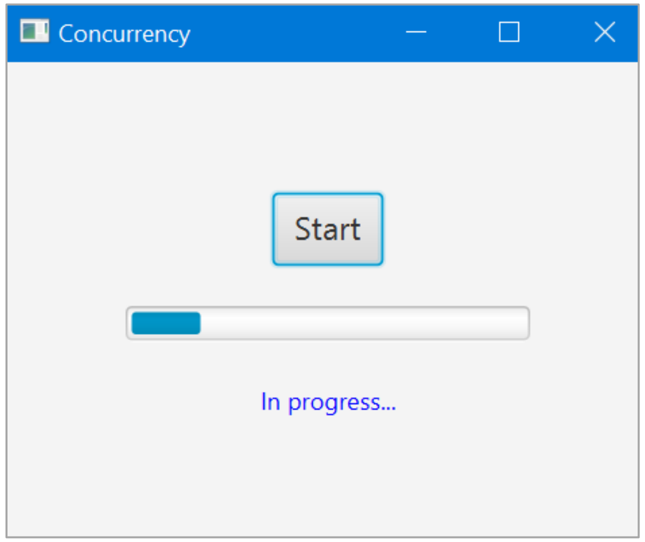
Bank cash assets: 300

Concurrency in JavaFX (1)

- Only a single thread, i.e. the JavaFX application thread, can render anything on the screen
- A long running task in JavaFX may leave the GUI unresponsive
  - It may be run in a separate thread instead
  - To modify the GUI directly, we can use the runLater method of the Platform class

Concurrency in JavaFX (2)

- E.g. updating JavaFX progressBar



```
void startRunning(ActionEvent actionEvent) {
   var taskThread = new Thread(() -> {
      var progress = new AtomicInteger(0);
      for (int i = 0; i < 10; i++) {
         try {
            Thread.sleep(500);
            progress.set(progress.get() + 10);
            Platform.runLater(() -> {
               progressBar.setProgress(progress.get() / 100.0);
               var msg = progress.get() == 100 ?
                         "Done." : "In progress...";
               message.setText(msg);
            });
         } catch (InterruptedException e) {
            e.printStackTrace();
         }
      }
   });
   taskThread.start();
}
```

Set Assessment 2