Day 2 : How Powerful

JavaFX in 4 Days

Zaw Min Lwin
CTO
SOLT Engineering,.Ltd.

2.3 Property & Bindings

JavaFX in 4 Days

Contents

- Property
  - Single Value Property
  - Properties Collections
    - Custom Property Object
- Binding
  - High Level Binding
  - Low Level Binding

Property

- Binding is one of useful feature of JavaFX
- You can bind one variable to another by using Binding API
- Property objects are used to bind by Binding API
- You can use predefined property types or you can use properties attributes as Java Bean

How to create

```
public class Sample1 {

    public static void main(String[] args) {
        IntegerProperty val1 = new SimpleIntegerProperty(10);
        IntegerProperty val2 = new SimpleIntegerProperty(2);

        System.out.println(val1.add(val2).intValue());
    }
}
```

Creating Object

The result is 12. What is Interesting?

What happen?

```
public static void main(String[] args) {
    DoubleProperty var1 = new SimpleDoubleProperty(10);
    DoubleProperty var2 = new SimpleDoubleProperty(2);

    NumberBinding result = var1.add(var2);

    System.out.println(result.intValue());

    var1.set(15);

    System.out.println(result.intValue());

    var2.set(15);

    System.out.println(result.intValue());
}
```

Bind var1 and var2

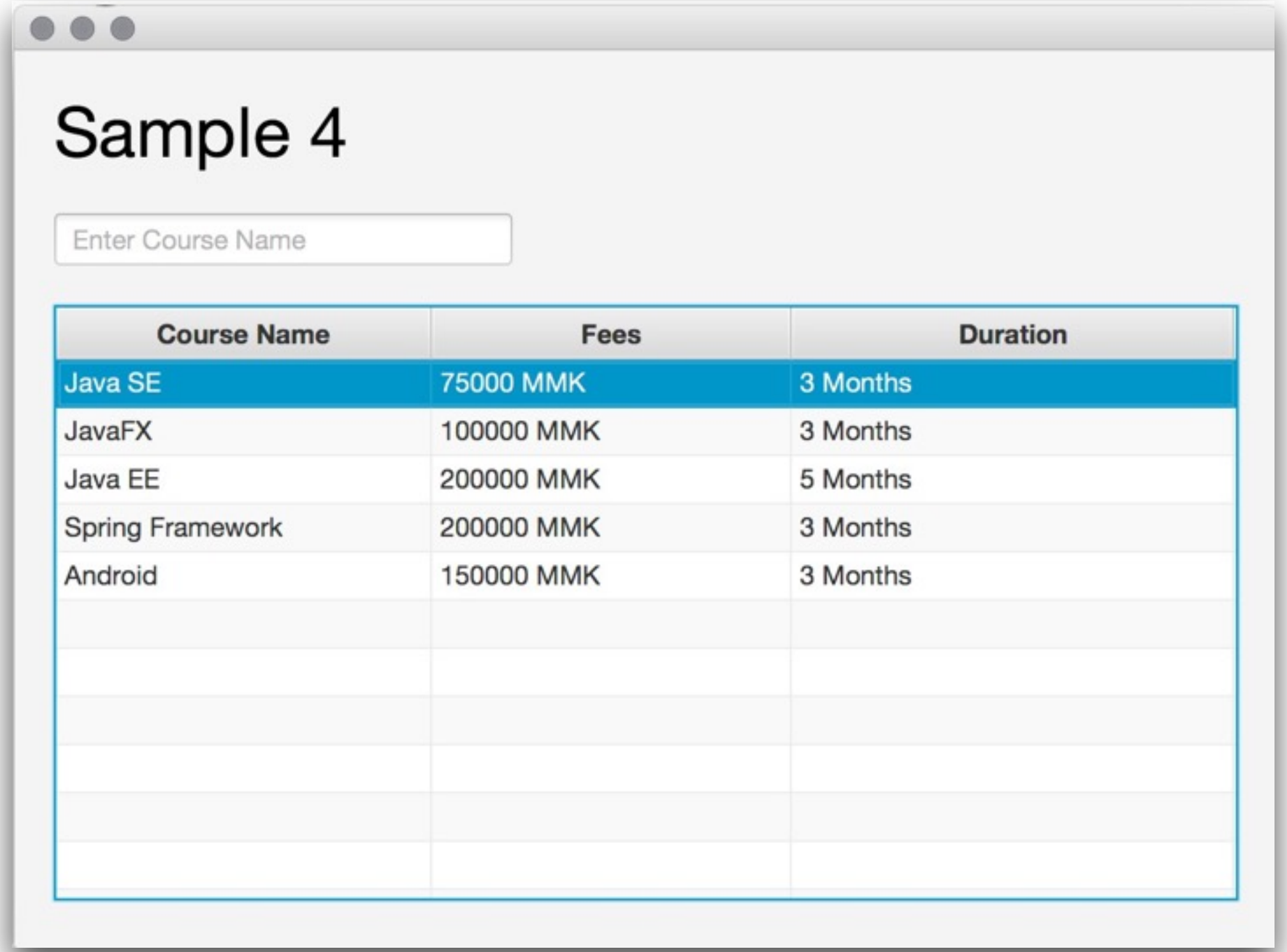
When the original variable is change, the result will change

Using Listener

```
public static void main(String[] args) {
   DoubleProperty var1 = new SimpleDoubleProperty(10);
   DoubleProperty var2 = new SimpleDoubleProperty(2);

   NumberBinding result = var1.add(var2);
   result.addListener((a, b, c) -> {
      System.out.println("Old Value -> " + b);
      System.out.println("New Value -> " + c);
   });

   var1.set(14);
   var2.set(15);
}
```

You can observe value changes and execute actions by adding listener

In SceneGraph



Using Property in SceneGraph

```
@FXML
private TextField name;
@FXML
private TableView<Course> table;
// other members

@Override
public void initialize(URL location, ResourceBundle resources) {

    table.getItems().addAll(courses);
    // other codes

    // add listener
    name.textProperty().addListener((a, b, c) -> {
        table.getItems().clear();
        table.getItems().addAll(courses.stream()
                .filter(course -> course.getName().startsWith(c))
                .collect(Collectors.toList()));
    });
}
```

Adding Listener to TextProperty

Properties Collections

- Collections that used in JavaFX are also observe the changes.

- Major collections are ObservableSet, ObservableList and ObservableMap

- Most of the methods are the same as Collections

- But Objects are generated by static method FXCollections utility class

FX Collections

```
ObservableList<String> list = FXCollections
        .observableArrayList("Language Specifications",
                "OOP", "Essential API");

list.addListener(new ListChangeListener<String>(){

    @Override
    public void onChanged(Change<? extends String> c) {
        while(c.next()) {
            if(c.wasAdded()) {
                System.out.println("Add : " + c.getAddedSubList());
            } else if(c.wasRemoved()) {
                System.out.println("Rmv : " + c.getRemoved());
            }
        }
    }});

list.add("JavaFX");
list.addAll("Lambda Expression", "Stream API", "Date & Time API");

list.remove(2);

System.out.println(list);
```

Add Listener

Add element to collection

Remove from collection

Custom Object Property

- You can use Property Objects in member of Custom Object
- You need to obey Java Beans Naming Rules
- If you want to get Property Values the name must be valueProperty

Bill

```
public class Bill {

    private DoubleProperty price = new SimpleDoubleProperty(0);
    private IntegerProperty count = new SimpleIntegerProperty(0);
    private NumberBinding total = price.multiply(count);

    public double getPrice() {
        return price.doubleValue();
    }

    public void setPrice(double price) {
        this.price.set(price);
    }

    public DoubleProperty priceProperty() {
        return this.price;
    }

    // other methods
}
```

Properties

getter

setter

property getter

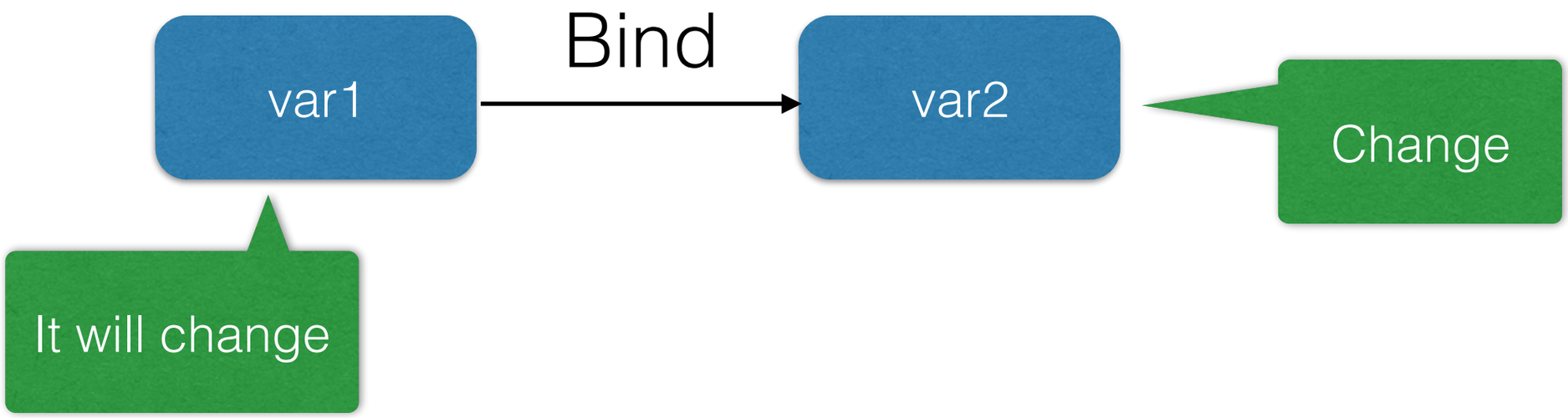
Using Bill

```
public static void main(String[] args) {
   Bill javaSE = new Bill();

   javaSE.totalProperty().addListener((a, b, c) -> {
      System.out.println("Total is : " + c);
   });

   javaSE.setCount(10);
   javaSE.setPrice(75000);
}
```

Adding Listener

Bindings

- By using Binding API you can synchronise one property variable to another.



Binding Sample

```
public static void main(String[] args) {
    StringProperty var1 = new SimpleStringProperty("Java SE");
    StringProperty var2 = new SimpleStringProperty("Java EE");

    System.out.println(var1.get());

    var1.bind(var2);

    System.out.println(var1.get());

    var2.set("Java FX");

    System.out.println(var1.get());
}
```

var1 bind to var2

value of var1 is “Java EE”

value of var1 is “JavaFX”

Lazy Binding

```
public static void main(String[] args) {
   StringProperty var1 = new SimpleStringProperty("Java SE");
   StringProperty var2 = new SimpleStringProperty("Java EE");

   var1.addListener(a -> {
      System.out.println("Change Value : " + a);
   });

   var1.bind(var2);

   var2.set("JDBC");
   var2.set("Java FX");

   System.out.println(var1.get());
}
```

Invalidate Listener

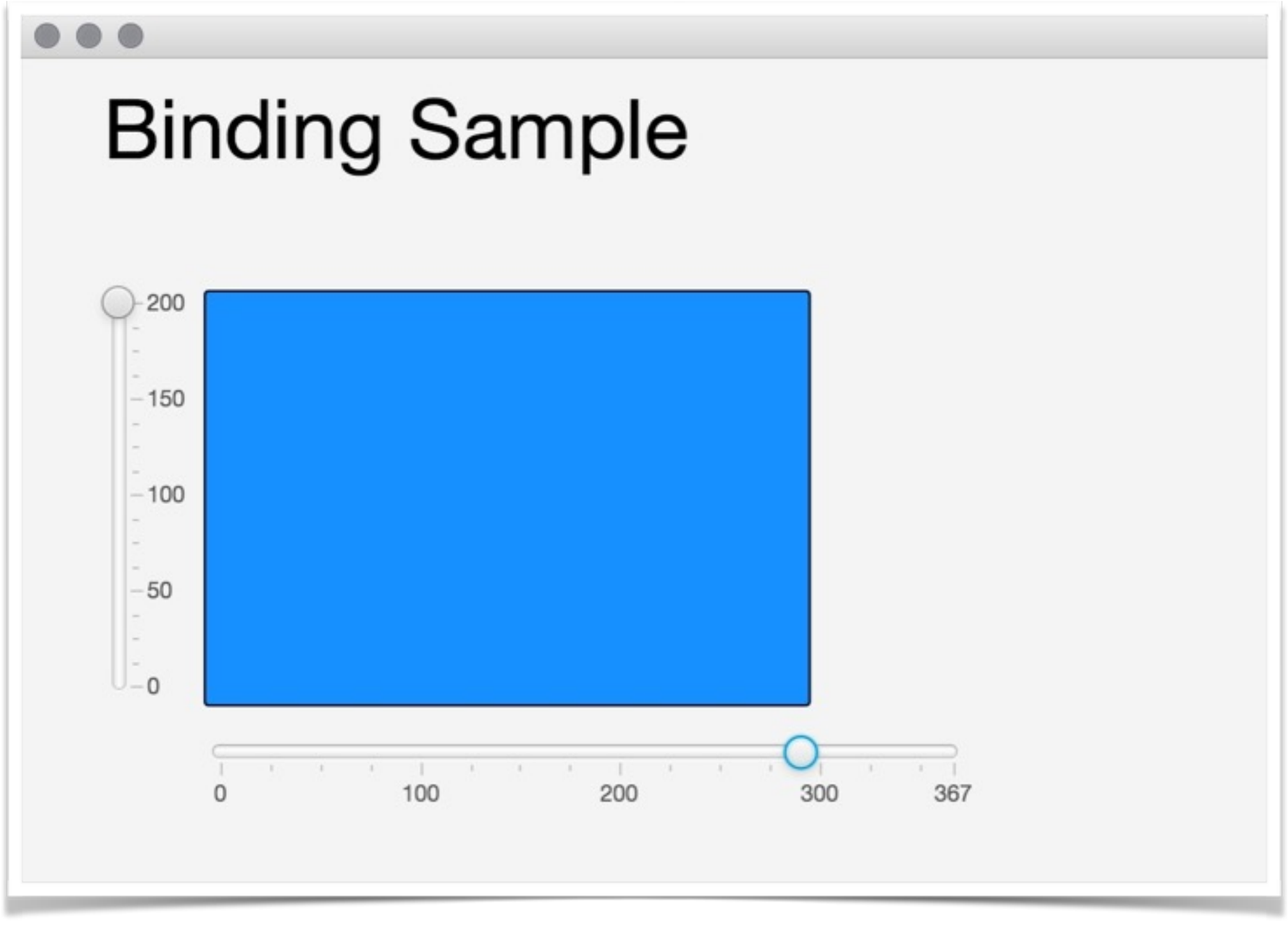
Value will be update when get method has been called

Binding With UI Components

```
@FXML
private Rectangle rec;
@FXML
private Slider sliderY;
@FXML
private Slider sliderX;

@Override
public void initialize(URL location, ResourceBundle resources) {
    sliderY.valueProperty().bindBidirectional(rec.heightProperty());
    sliderX.valueProperty().bindBidirectional(rec.widthProperty());
}
```

Binding each other

Binding With UI



High Level Binding API

- High Level Binding API Contains two methods
  - Fluent API
    - Fluent API exposes methods on various dependency objects
  - Bindings Class
    - Bindings Class provide static factory methods

Using Fluent API

```
public static void main(String[] args) {
    DoubleProperty var1 = new SimpleDoubleProperty(0);
    DoubleProperty var2 = new SimpleDoubleProperty(0);

    NumberBinding multiply = var1.multiply(var2);
    NumberBinding adding = var1.add(var2);
    NumberBinding dividing = var1.divide(var2);
    NumberBinding substract = var1.subtract(var2);

    var1.set(10);
    var2.set(3.3);

    System.out.println(multiply.doubleValue());
    System.out.println(adding.doubleValue());
    System.out.println(dividing.doubleValue());
    System.out.println(substract.doubleValue());
}
```

Fluent API methods

Bindings Class

```
public static void main(String[] args) {
    DoubleProperty var1 = new SimpleDoubleProperty(0);
    DoubleProperty var2 = new SimpleDoubleProperty(0);

    NumberBinding multiply = Bindings.multiply(var1, var2);
    NumberBinding adding = Bindings.add(var1, var2);
    NumberBinding dividing = Bindings.divide(var1, var2);
    NumberBinding substract = Bindings.subtract(var1, var2);

    var1.set(10);
    var2.set(3.3);

    System.out.println(multiply.doubleValue());
    System.out.println(adding.doubleValue());
    System.out.println(dividing.doubleValue());
    System.out.println(substract.doubleValue());
}
```

Factory Methods of Bindings Class

Using Both

```
public static void main(String[] args) {
   Bill bill1 = new Bill(10, 1050);
   Bill bill2 = new Bill(3, 1800);
   Bill bill3 = new Bill(5, 2030);

   NumberBinding total = Bindings.add(
        bill1.totalProperty().add(bill2.totalProperty()),
        bill3.totalProperty());

   System.out.println(total.doubleValue());
}
```

Fluent and Bindings

Low Level Bindings

- If high level API is not enough your requirements, you can use low level API

- Low level API is for developers who want more flexible (or better performance) that provided by High Level API

Using Low Level API

```
private NumberBinding total = new IntegerBinding() {
   {
      super.bind(burmese,
            english, maths,
            physics, chemistry,
            biology);
   }
   @Override
   protected int computeValue() {
      return Bindings.add(
            burmese.add(biology).add(english),
            maths.add(physics).add(chemistry))
               .intValue();
   }
};
```

Bind IntegerProperty to TextProperty

```
@Override
public void initialize(URL location, ResourceBundle resources) {
    exam = new Exam();

    exam.burmeseProperty().bind(getBinding(burmese));
    // codes
}

private IntegerBinding getBinding(TextField field) {
    return new IntegerBinding() {

        {
            bind(field.textProperty());
        }

        @Override
        protected int computeValue() {
            try {
                return Integer.parseInt(burmese.textProperty().get());
            } catch (Exception e) {}
            return 0;
        }
    };
}
```

IntegerProperty

TextProperty