An Attempt to Give Swing a Breath...

...

Diploma Thesis

Student: ***Zougianos Georgios*** (2442)
Supervisor Professor: ***Zarras Apostolos***

Motivation

Java on desktop - not dead yet (see Talk by Gerrit Grunwald)

Swing:

- The successor of AWT
- Stable
- Part of the JDK (JavaFX removed after JDK 11 in 2018)

Objective = a framework that:

- Introduces an MVC variation - Multiple subviews
- Takes advantage of dependency injection design pattern - Decoupling
- Clean & concise code - reduces boilerplate

Other Swing libraries & tools

- Standard Widget Toolkit (SWT) - brother of Swing
  - Based on AWT like Swing
  - Platform dependent
  - Eclipse IDE is written with SWT
- SwingLabs - `@Deprecated`
  - Widgets such as autocomplete textfields, date pickers, etc...
  - APIs (`SystemTray`, `Desktop`) that were integrated to Swing
- JGoodies - Products & Freeware
  - Data binding
  - Data validation
  - Custom APIs to manipulate widgets
- Window Builders
  - Help the developer to build the UI with drag n' drop functionalities
  - Most Java IDEs have one for Swing UIs

Background

- Model - View - Controller
  - No strict rules on how to implement
  - Separation of Concern is the core idea
  - The idea is used in web apps as well
  - A lot of variations on top of it (MVA, MVP, MVVM, Presentation Model)
- Three elements/layers
  - Model - Business logic & data
  - View - What user sees on screen (text, icons, borders, painting logic, etc)
  - Controller - Handles user actions, orchestrates the other two

Trygve Reenskaug

“ Different people reading about MVC in different places take different ideas from it and describe these as 'MVC' ”

- Martin Fowler on GUI Architectures

Background

Swing uses an alternate MVC

- Each widget is a View & Controller with the View being a delegate (e.g `JButton`, `ButtonUI`)
- Two kind of models:
  - GUI state model (e.g ButtonModel - `isPressed()`)
  - Data model (e.g TableModel - `getValueAt(int row, int col)`)
- Listeners to widgets as an additional controllers for each type of interaction

```
jButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            System.out.println("User pressed the button");
        }
 });
```

Background

Inversion of Control (IoC) - Hollywood principle

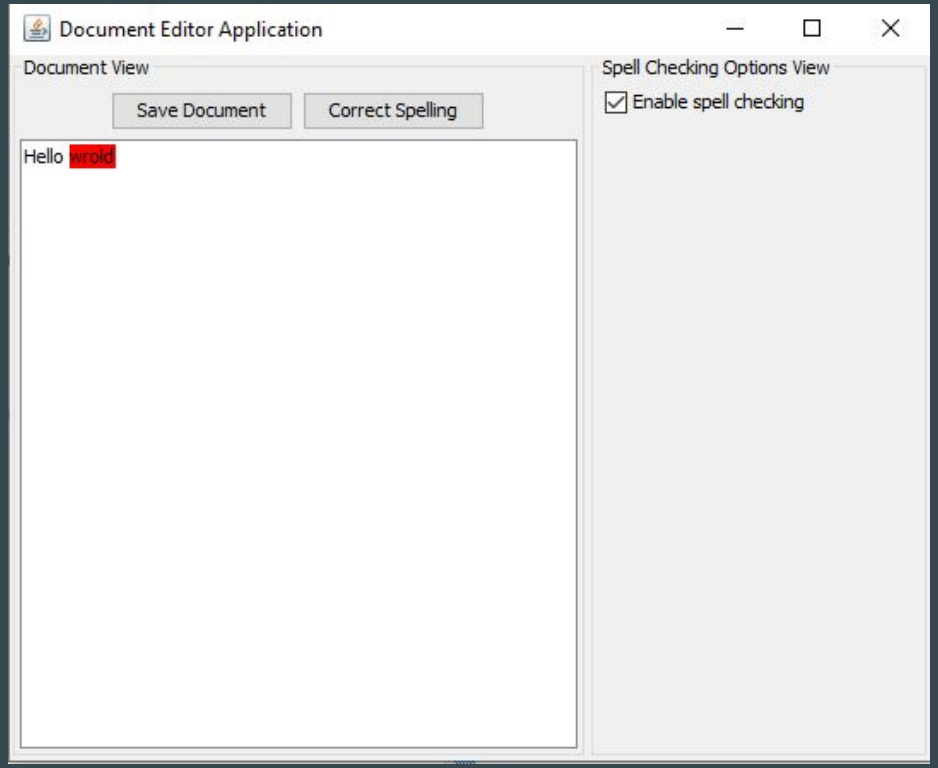
- *“Don't call us, we'll call you”*
- From our daily life: JUnit, Template method pattern

Dependency Injection

- Separates creation from usage
- Decoupling
- Terminology:
  - Dependency = Parameter
  - Injection = Passing of the parameter
  - Injector = Creates the client
- In this thesis:
  - Guice (as IoC container) - JSR 330

```
@Inject
public Thesis(Supervisor supervisor) {
      this.supervisor = supervisor;
}
```

Let's build...


Document Editor Application
Document View
Save Document
Correct Spelling
Hello wrold
Spell Checking Options View
Enable spell checking

Usually in Swing: Architecture

Smart UI

- Simple to understand
- Fast development when there are few use cases
- Absence of Separation of Concern - God classes
- Single Responsibility Principle (SRP) violation
- Unlikely testable
- Often a result from the usage of a WindowBuilder

Usually in Swing: Architecture

Container-level MVC

- Separation of Concern:
  - A Controller class
  - A View class
  - (At least one) Model class
- Risk of God objects when application grows
- Multiple ways to hook into user actions
  - View object registers the listeners (anonymous classes) - Delegate callbacks to the Controller
  - Controller is a listener itself - Implements Interfaces of the toolkit
  - ...

Usually in Swing: Architecture

Beyond container-level MVC: Hierarchical MVC

- A hierarchy of MVC triads
- A Controller class, A view class, 0...N Model classes for each MVC triad
- Communication between triads through the controllers
  - Parent - child(ren) relationship
- Code of a use case might exist in multiple classes
- A parent controller might end up a God object if there is a lot of communication between two children of the same level

Usually in Swing: Concurrency

- Swing is not thread-safe
- Swing code must run in its own thread: Event Dispatch Thread (EDT)
- A heavy task in the EDT “freezes” the GUI
- Debugging a “frozen” GUI can be tough
- `SwingWorker` API = boilerplate even for simple cases

Swing Boot

...

Swing Boot: Architecture

A different kind of MVC:

- Model layer: Same as MVC
- Control(ler) layer: A controller for **each use case**
  - Must implement `swingboot.Control` interface - `perform(T par)`
  - Essentially **commands**
  - Translate GUI state to record state & vice versa
- View layer: (Almost) Same as MVC
  - A view object can be splitted to multiple subviews
  - Declare which and when controls are performed

Swing Boot: Annotations

- `@InstallControls`
  - Indicates that a view object has control declarations
  - Used upon the view class
- @OnEventHappend
  - Used upon widget fields
  - “@On” + swingListener.getMethodName()
  - Accepts a `Class<Control>` as `value()`
  - `Basic Swing event filtering`

```
@InstallControls
class DocumentView extends JPanel {
    @OnActionPerformed(SaveDocumentControl.class)
    private JButton saveDocumentButton = new JButton("Save Document");

    @Inject
    public DocumentView() {...}     }
```

Swing Boot: Annotations

- `@InitializedBy`
  - Accepts a `Class<Control>` as `value()`
  - Performed **once** before all other controls are installed
  - Used upon the view class
- `@WithoutControls`
  - Used upon a method of a view class
  - Deactivates all controls during the execution of the method (PassiveView)

Swing Boot: Annotations - Control's Parameter

- The generic parameter of `perform(T genericParamter)` method
  - Rarely needed
  - Use it to alternate the flow of a use case instead of having two controls
  - Use it to pass information from Swing events (injected to the method as argument)
  - `@ParameterSource` upon a **non-void method in the view class that declares the control**
  - Identifier to `@OnEventHappend` and `@ParameterSource`

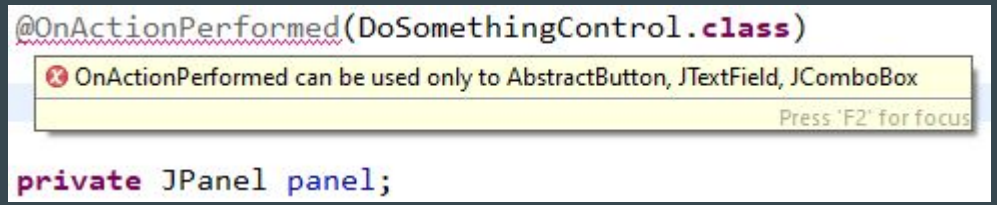
```
@OnActionPerformed(value = CorrectMisspelledWordsControl.class, parameterSource = "ignore_uppercase")
private JButton correctMisspelledWordsButton;

@OnActionPerformed(value = CorrectMisspelledWordsControl.class, parameterSource = "ignore_uppercase")
private JButton correctMisspelledWordsIgnoreUppercaseButton;

@ParameterSource("ignore_uppercase")
Boolean shouldIgnoreUpperCase(ActionEvent event) {
    return event.getSource() == correctMisspelledWordsIgnoreUpperaseButton;
}
```

Swing Boot: Annotation Processor

- Annotations are misplace prone
  - Errors spotted in runtime
  - Reduces developer's productivity
- Solution: An annotation processor
  - Finds misplaced annotations at compile time



Swing Boot: Concurrency

- Assert that the method runs in the “correct” thread
  - @AssertBackground, @AssertUi **upon a method**
  - Log or throw an exception if the method is not executed in the expected thread
- Decide in which thread the method runs
  - @InUi, @InBackground **upon a method**
  - @InBackground cannot be cancelled
  - @InBackground makes no sense to return a value

“ Is anyone in the room Doug Lea? No. Is anyone in the room Brian Goetz? No. Then don’t use threads! “

- Dan North on Decisions, Decisions

Swing Boot: Pros & Cons

- Advantages
  - Separation of Concern: MVC
  - Loose coupling between the VC layers: Dependency Injection
    - Besides view composition, there are no hierarchies
  - Reduced code noise: Annotations
    - No anonymous classes
- Disadvantages
  - You “marry” the framework
  - Controls: A lot of classes to coordinate
  - New knowledge for an old toolkit

Swing Boot: How it works

- `ControlModule`
- `ConcurrencyModule`
- Listen injections from the IoC container
- Scan and analyze the class for reflection elements (fields, methods, annotations, etc)
- @WithoutControls, @In[Thread], @Assert[Thread] = **AOP**

Swing Boot: In future releases...

- Support other IoC containers
  - Spring? Dagger2?
- Replace reflection with code generation
  - Annotation processors
- Cancelable/Restartable @InBackground
- Documentation/Wiki page
- Support Kotlin?

Is it worth it?

…

“It depends”

Swing Boot: Facts

- A maven project
- Tests
  - JUnit 5
  - Mockito
  - jOOR (annotation processor)
- Repository
  - https://github.com/gzougianos/swing-boot

Thank you!

...

Questions?