Labwork 5 Builder and Decorator– Object Orientation with Design Patterns 2015

This lab is worth 6% or 60 points from the total 500 points for labwork this semester (includes 10 marks UML exam preps)

UML DIAGRAM DRAWING EXAM PREPS (10 Points)

Draw a UML diagram for **Labwork 2 Part 4** the AddressFactory. Include ALL class level details and relationships between participants in the diagram.

All pattern participants included\shown	(4 marks)
All relationships shown in correct UML syntax	(4 marks)
Diagram class level details (attributes\behaviours)	(2 marks)

Part 1: Basic Builder example extended (15 points)

Create a new project called **Lab5Part1**. Modify the Builder example from the lecture to cater for **Japanese** stocks, where Japanese stocks will have 5 or more investments listed and the GUI will **build a Radio Button view** and return for display in the main GUI [i.e., the pattern will work as follows: 3 or less will build Check Box, less than 5 will build List, 5 or more will build the Radio Button].

- Modify **Director** to cater for radio button panels (3 points)
- Add Japanese investment lists to main chooser (>=5) (3 points)
- Add new subclass and implementation for radio button choice (5 points)
- Test the program for 5 or more stocks (Japanese) (4 points)

Part 2: Decorator with GUI exercise **(20 points)**

Create a new project called **Lab5Part2**. Implement a **Decorator** design pattern so that a JLabel border can be changed from one colour to another using the mouse (use mouse enter event for example). This would mean that you would create an abstract Decorator class which inherits from JComponent and subclasses for each type of decoration required (RedDecorator, BlueDecorator, NoBorderDecorator); these subclasses will paint the border once the mouse enters.

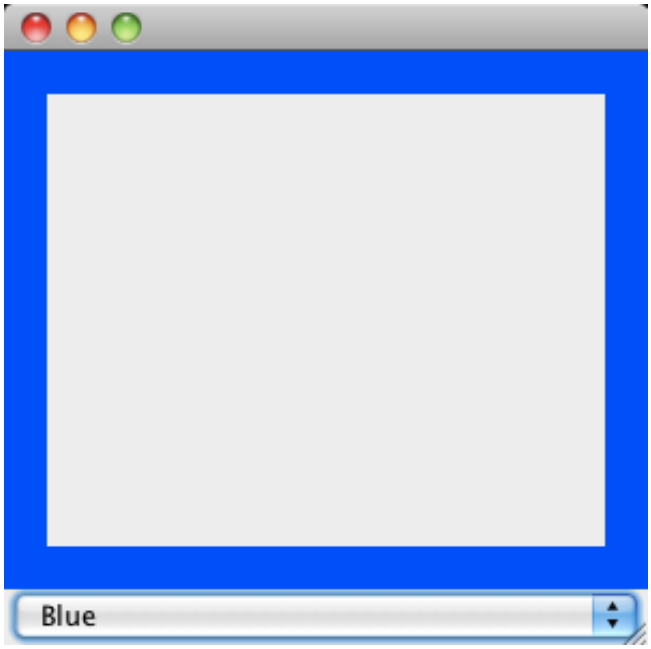


Figure 1: Blue border added when mouse enters using BlueDecorator

Proposed marking scheme:

- Abstract decorator (3 points)
- Concrete BlueDecorator (3 points)
- Concrete RedDecorator (3 points)
- Concrete NoBorderDecorator (removes the borders) (3 points)
- Mouse Listeners implemented (3 points)
- JCombo implemented (2 points)
- Set decorators works in the GUI for all three settings (3 points)

Part 3: Full Builder example with GUI (20 points)

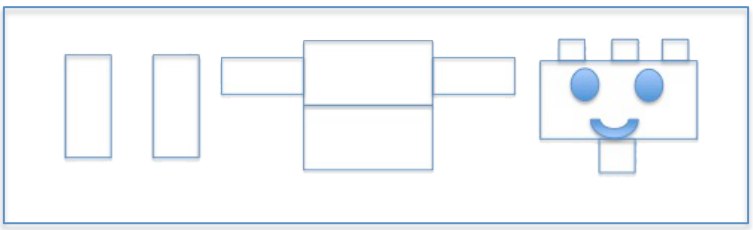
Problem: To deliver a program to build robots and display the finished robot to the screen as a JPanel within the main JFrame program: the building process must be generalizable to all future robot designs

Possible solution: Use the builder pattern to return the completed JPanel for inclusion in the GUI, encapsulate the building process within the builder pattern, e.g., buildHead(), buildBody(), buildBase()

Create a new project called **Lab5Part3**. Apply the Builder pattern to build a robot builder GUI system in a JFrame. The Builder pattern will encapsulate the building process for all robots, to build the top section (head), the middle section (the body), and the bottom section (the legs\base) and return the completed robot as a JPanel of three images. For example you could build the following robot types: **LegoRobot** and **StickmanRobot** and the robots will be built and returned for display (it would help to sketch a UML diagram before implementation, use the generic Builder pattern as the guide):

StickmanRobot components

LegoRobot components



- Director to direct building process (3 points)
- Abstract builder (3 points)
- Concrete builder for one type of robot, e.g., StickmanRobot (5 points)
- Concrete builder for another type of robot, e.g., LegoRobot (5 points)
- JFrame application to choose and display robots using Builder (4 points)