Labwork 9 Summary Lab – 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 6 Part 1** the **Adapter** pattern for the zoo animals. Include ALL class level details and relationships between participants in the diagram (Note: YOU DO NOT NEED TO HAVE IMPLEMENTED THE SOLUTION FULLY TO DRAW A UML DIAGRAM OF THE SYSTEM)

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: UML Task – Abstract Factory (15 points)

Create a new project called **Lab9Part1**. Create a detailed UML diagram of the **abstract factory** design pattern. Include all the relationships and participants in the diagram. Outline the function of EACH of the participants shown in the diagram (IN YOUR OWN WORDS).

Proposed marking scheme:

- Participants shown (Abstract Fact, Concrete, Products, Client) (8 points)
- Correct UML relationships shown (3 points)
- Outline each of the participant functions (4 points)

Part 2: UML Task – Builder (15 points)

Create a new project called **Lab9Part2**. Create a detailed UML diagram of the **abstract factory** design pattern. Include all the relationships and participants in the diagram. Outline the function of EACH of the participants shown in the diagram (IN YOUR OWN WORDS).

Proposed marking scheme:

- Participants shown (Director, Conc. Builders, Products, Client) (8 points)
- Correct UML relationships shown (3 points)
- Outline each of the participant functions (4 points)

Part 3: Singleton example **(20 points)**

Create a new project called **Lab9Part3**. Create a Java class called **President** so that only one instance of this class can be created, i.e., apply the **Singleton Pattern** to this class (include at least two attributes, e.g., name and age (>35 of course!)). Provide a method within the class called **createPresident()** that returns a reference to the only possible instance of the class. Finally develop a test application that attempts to create more than one instance of the President object (prove that EACH instance returned is a shared instance within the test program).

Proposed marking scheme:

- President class created (at least two attributes) (4 points)
- Implement the President **private** constructor (4 points)
- Implement the **createPresident()** method to return instance (5 points)
- Create test program to attempt to create more than 1 instance (4 points)
- Write code to prove only ONE instance is created in program (3 points)