Final exam

10 pt

[15 minutes]

We learned 3 different patterns that all 3 act as a sort of wrapper in front of a target class. Give the names of these 3 patterns and for all 3 patterns explain their characteristics. Explain how they are different from each other.

5 pt

[10 minutes]

Describe how we can relate **dependency injection** to one or more principles of SCI. Your answer should be about 2 to 3 paragraphs. The number of points you get for this questions depends how well you explain the relationship between **dependency injection** and one or more principles of SCI.

30 pt

[25 minutes]

In the lab we wrote our own test framework. Suppose all classes we want to test are annotated with **@Service** annotation:

```
public interface Calculator {
   public void reset() ;
   public int add(int newValue);
   public int subtract(int newValue);
}

@Service
public class CalculatorImpl implements Calculator {
   private int calcValue=0;

   public void reset() {
      calcValue=0;
   }

   public int add(int newValue) {
      calcValue=calcValue+newValue;
      return calcValue;
   }
```

```
    public int subtract(int newValue) {
        calcValue=calcValue-newValue;
        return calcValue;
    }
}
```

Our framework should now support the following tests classes:

```
@TestClass
public class MyTest {
    @Inject  //calculator is injected by the testframework
    Calculator calculator;

    @Before
    public void init() {
        calculator = new CalculatorImpl();
    }
    @Test
    public void testMethod1() {
        assertEquals(calculator.add(3),3);
        assertEquals(calculator.add(6),9);
    }
    @Test
    public void testMethod2() {
        assertEquals(calculator.add(3),3);
        assertEquals(calculator.subtract(6),-1);
    }

}
```

Our framework needs a framework context, and we can start this framework with the following code:

```
public class Application {

    public static void main(String[] args) {
        FWContext fWContext = new FWContext();
        fWContext.start();
    }
}
```

Now all test methods in the test classes should be executed.
You can assume that the framework contains the following Asserts class:

```
public class Asserts {

    public static void assertEquals(int x, int y) {
        if (x != y)
            System.out.println("Fail: result = "+x+" but expected "+y);
```

```
    }
}
```

You can also assume that the framework contains all the definitions for the annotations (@Inject, @Before, etc)
The only thing we still need to do for this test framework is writing the **FWContext** class.

Given is the partial pseudo code of this FWContext class. Only the pseudo code of the method getServiceBeanOftype() is given. Write the missing pseudo code so that the framework does the necessary tasks. Write the pseudo code in a similar way as the pseudo code given in the method getServiceBeanOftype().

```
public class FWContext {

    public FWContext() {

    }

    private void performDI() {

    }

    public Object getServiceBeanOftype(Class interfaceClass) {
        loop over serviceObjects list
          for every serviceObject, get its interfaces
          loop over the interfaces
            if (interfaces is equal to interfaceClass.getName()){
              return current serviceObject
            }
    }

    public void start() {

    }
}
```

55 pt

[70 minutes]

The question is given in the attachment. Open the PDF file in the browser.

Draw the solution of this question in StarUML and **upload** your solution as a **JPG picture** (only JPG pictures are accepted) as solution to this question