National University of Computer & Emerging Sciences, Karachi
Spring-2020 CS-Department
Final Exam
24TH June 2020, 9:00 AM – 12:00 PM

Course Code: CS441	**Course Name: Design Defects and Restructuring**
Instructor Name: Engr. Abdul-Rahman	
Student Roll No:	**Section No:**

Instructions:

- Read each question completely before answering it. There are 4 question and 4 pages.
- Handwriting / diagrams should be neat and clean. Cutting and over-handwriting is not allowed.
- Explain answers in your own words. Use your own scenarios / code snippets, don't use the already given scenarios from text-book, lecture slides or internet.
- You are required to finish paper in 180 minutes. Use additional 30 minutes to upload answer.
- Use Cam Scanner to create **PDF** and rename your solution as **YourName_Section_RollNo.PDF**.
- PDF should be in sequence as per question ordering. The orientation should be portrait for each page. Each page of PDF should be clearly visible. Write Name /Roll No and Page No on each page.
- In 1st hour and last 30 mins of exam I will be online at: **https://meet.google.com/emh-vamv-kde**
- Submit Paper in your section only (as registered in flex), otherwise you will get zero.
- The paper submission on Google Forms is restricted to your **FAST-NU email ID only**.
- Ideal PDF size is **5-10 MB**. You can't upload more than 100MB due to limit on each file in drive.
- Paper will be distributed via Slate/Facebook/Email with 3 URLs of Google Forms for each section.
- There is **only one submission allowed** for uploading the solution of final paper. So be very careful and double check everything before uploading. Don't submit paper in multiple places, your preference of submission should be (1) Slate, and (2) Google form.
- Those of you who are using slate, they may experience network connectivity issues. Alternatively, you may access via IP address : **http://203.124.42.218:8080/portal**
- Final Exam Google form submission URL for Section 8A: **https://forms.gle/xq4R7deA9QjdKTJu9**
- Final Exam Google form submission URL for Section 8C: **https://forms.gle/ED7hQUHcqnjKzv5f6**
- Final Exam Google form submission URL for Section 8E: **https://forms.gle/wQKicary1PPTo7AX8**
- If you are having issues in upload, you may also email the answer (within deadline) to **abdulrahman@nu.edu.pk** with subject **DDR-Exam_YourName_Section_RollNo**. Don't send to any other email address and only use your **FAST-NU email ID**, otherwise submission won't be considered.
- Solution will not be acceptable in any case after **3.5 hours**. Make sure to submit it within deadline.
- Electricity failure during upload will not be accepted as an excuse for not being able to submit the paper within time, you are supposed to upload it via mobile data + cellphone.

Time: 180 minutes. **Max Marks**: **60**

Q1: **[Marks=10]**

a) A Drawing editor application can add, remove and draw any Graphic Object on canvas as well as can get the child of any Graphic object in the hierarchy. The graphic system let users build complex diagrams out of simple components. The user can group components to form larger components, which in turn can be grouped to form still larger components. A simple implementation could define classes for graphical primitives such as Text, Line, Rectangle plus other classes that act as containers for these primitives such as Pictures.
   i. What design pattern can be used to represent this problem in programming.
   ii. Draw UML class diagram of the given scenario using your identified design pattern.
   iii. Draw UML class diagram of the general structure of the pattern you have identified.

b) What is Content Coupling? Define it and explain it using a detailed code example. Explain in your own words. Use your own scenario, don't use scenarios/code given in text book or in lecture slides.

c) Explain Open Closed Principle. How can we confirm to OCP principle? Explain in your own words. Use your own scenario, don't use scenarios/code given in text book or in lecture slides.

d) Explain the concept of "Speculative Generality" in detail with examples. What could be the possible Reasons and Signs and Symptoms for this problem? What are the possible treatments to this problem?

Explain the treatments in detail. Explain in your own words. Use your own scenario, don't use scenarios/code given in text book or in lecture slides.

e) Parallel Inheritance Hierarchies can become a code smell. What are the Signs and Symptoms of this code smell? What could be the possible Reasons for this Problem in code? What are the appropriate treatments to this problem? Explain the treatments in detail. Explain in your own words. Use your own scenario, don't use scenarios/code given in text book or in lecture slides.

Q2: Refactoring [Marks=20]

For each of the following questions, write complete code including any definition of functions that you may introduce after refactoring. Also explain the reason behind using a particular refactoring technique, what benefit we get by using it and what bad smell have been removed after refactoring.

a) Refactor the following code using consolidate conditional expression.

```
public calculateSoftwareCost(methodology, programming_language, deadline) {
    if (methodology == 'test driven development') {
        return false;
    } if (programming_language == 'AngularJS') {
        return false;
    } if (deadline == '1.3 months') {
        return false;
    }
    // continue cost calculation
}
```

b) Refactor the following code using Decompose Conditional.

```
public function calculateProjectRate(Project project, urgency, type) {
    if(type == 'complex' || urgency == 'rush') {
        rate = (project->getBaseRate() * project->getSize()) * 1.5;
    } else {
        rate = project->getBaseRate() * project->getSize();
    }
    return rate;
}
```

c) In the following code example, Refactor the code by Extract class technique.

```
abstract class Shape {
  public void Draw() {
    try {
        // draw
    }
    catch (Exception e) {
        LogError(e);
    }
  }
  public static void LogError(Exception e) {
    File.WriteAllText(@"c:\Ex.txt", e.ToString());
  }
}
```

d) Multiple clients need to use the same part of "Inventory" interface. Refactor the following code to utilize the extract interface technique.

```
public class Inventory{
   private String ItemName;
   private double amount;
   private String Product_Code;
   private String Department;
   public boolean hasSpecialUse;
   public getRate();
   public double demand_distro = 99.4;
}
```

e) Refactor the following code using Extract Variable technique.

```
public static boolean isWithinRange(double initialValue, double finalValue) {
    return getSpeed()>=initialValue && getSpeed()<=finalValue;
}
```

f) Refactor the following code using Extract Superclass technique.

```
public class Triangle implements IDrawable {
  public Point p1, p2, p3;
  public Integer Color;
  public void Draw() {
    // draw...
  }
}
public interface IDrawable {
  void Draw();
}
```

g) Refactor the following code using Extract Method technique.

```
public void main() {
    int x=1;
    int y=2;
    int z=x+y;
    int k=x+z;
}
```

h) Refactor the following code using inline Temp technique.

```
double CalculatePrice(double product_price) {
    int dPer = GetDiscountPercentage();
    double tPer = GetTaxPercentage();
    double dRatio = (double)dPer / 100;
    double tRatio = (double)tPer / 100;
    return product_price - dRatio * tRatio * product_price;
}
```

i) Find out the problem/bad smell in the following code and identify the refactoring technique to fix it.

```
private int calculateNetSalary(double salary, double premium) {
   bonus = 0.01 * salary;
   salary = salary + bonus;
   salary = salary + premium;
    return salary;
}
```

j) Find out the problem/bad smell in the following code and identify the refactoring technique to fix it.

```
float SetCurrentPrice() {
     float discountRatio;
     if (netPrice() > 5400) discountRatio = 0.75;
     else discountRatio = 0.78;
     return netPrice() * discountRatio;
 }
```

Q3: **Design Patterns** **[Marks=15]**

Assume that we have developed a functionality for digital signatures in a class called DSASig having an interface called Signature and a method called sign(String sigType, String fileName). Where sigType is the signature type being used and filename is the name of file being signed. Now we are successfully using this DSASig object to digitally sign any file from our client code called ClientApp which has the main driver code of our application. Now in the opensource market we discovered a new component containing two more types of signatures: ElGamalSig and ECDSASig, having interface called AdvancedSig having methods signECDSA() and signElGamal(). Now we wish to “extend” our DSASig component’s sign() method to also include the functionality of signECDSA() and signElGamal(). Note that we want to achieve the extension in our existing class, that is our existing DSASig.sign() method being accessed from the client code must be “extended” to be able to also sign via ECDSA and ElGamal present in the 3rd part opensource component.

a) Identify the design pattern name that best suites in above case study. You need to be very careful in choosing the pattern that will enable us to extend the existing functionality, and in order to integrate functionality of a 3rd part component with your own component, you also need to develop a bridge between two incompatible interfaces.
b) What additional class(es) for your design pattern will you have to create in order to plug-in the functionality exposed by 3rd party AdvancedSig interface?
c) Draw the class diagram for your identified design pattern based on the given scenario.
d) Write code for your selected design pattern and also write the main() / client code to test the classes of pattern. Test the client code with:
    a. sigType = dsa, fileName= topSecret.docx
    b. sigType = ecdsa, fileName= confidential.pdf
    c. sigType = elgamal, fileName= secret.txt
    d. sigType = rsa, fileName= news.rtf
e) Also show sample output.

Note: If you identify the wrong pattern then all the subsequent parts of the question will be invalid. You don't actually need to implement any digital signatures instead just print that file is signed using particular type of signature (dsa, ecdsa or elgamal etc.)

Q4: **Design Patterns** **[Marks=15]**

Assume that you need to create an application to prepare Roti. But there are several different kinds of Roti such as SheerMaal, RoghniNaan, ChilliCheeseNaan, EggNaan and Chapati. The general process to prepare the Roti contains following tasks: chooseFlour(), addIngredients(), addMilk(), heating() and addToppings(). For each kind of Roti, the tasks or steps chooseFloor() and addIngredients() has to be redefined/customized based on the type of Roti being prepared, whereas addMilk(), heating() and addToppings() will remain the same for all kinds of Roti. For each type of Roti you need to always add 1 cup of milk, standard heating duration is 5 minutes and always add sesame seeds as topping. Now, in the following table different Flour types and ingredients are given for each kind of Roti that you need to prepare:

Type of Roti	Flour type	Ingredients
SheerMaal	Whole Wheat Red Flour!	Sugar, and Ghee
RoghniNaan	Whole Wheat White Flour	poppy seeds, onion seeds and a dollop of butter
ChilliCheeseNaan	Whole Wheat White Flour	chopped green chillies, processed cheese, salt and some chopped coriander
EggNaan	Fine Wheat White Flour	half-fried egg, baking powder, salt, and sugar
Chapati	Fine Wheat White Flour	Edible Oil and Salt.

Solve the above given scenario using appropriate design pattern.

a) Identify the design pattern that best suites in above case study. You need to be very careful in choosing the pattern which places the invariant steps in the base class and lets subclasses redefine certain steps of a procedure without changing the procedure's structure. It is important that we don't need to add / remove functionality of any existing behavior at runtime. Our requirement is to solve it via compile time.
b) Draw the generalized pattern diagram for the pattern that you have identified.
c) Draw the class diagram for your identified design pattern for this particular scenario.
d) Write code for your selected design pattern and also write the main() / client code to test the classes of pattern. Test the client code with all the different kinds of Roti preparation and print "**********" as a separator between each type of roti being prepared.
e) Also show sample output.

Note: If you identify the wrong pattern then all the subsequent parts of the question will be invalid.

BEST OF LUCK!