United International **University**

Department of Computer Science and Engineering

CSE 1115: Object Oriented Programming

Midterm Exam Trimester: Spring 2025 Time: 1 hour 30 minutes Marks: 30

Any examinee found adopting unfair means will be expelled from the trimester / program as per UIU disciplinary rules.

Answer all questions.

---

1. a) Consider the following codes and write the output [5 Marks] [CO1]

MindGame.java

```
class MindGame {
    int a = 0;
    int b = 2;
    static int c = 0 ;
    static int d = 5;

    MindGame()
    {
        a--;
        b += 5;
        c++;
        d--;
    }
    static {
        c--;
        d += c;
    }
    {
        a = c++;
        b = b+d;
    }
    public void displayAB()
    {
        System.out.println("a = "+a);
        System.out.println("b = "+b);
    }
    public static void displayCD()
    {
        System.out.println("c = "+c);
        System.out.println("d = "+d);
    }
}
```

Main.java

```
public class Main {
    public static void main(String[]
        args)

    {
        MindGame.displayCD();
        MindGame m1 = new MindGame();
        m1.displayAB();
        m1.displayCD();
        MindGame.displayCD();
    }
}
```

1. without creating object just call a static method of the class it runs the whole class.

2. value also change at the time of assign like c = 0 ther
= c++;
so a = 0 but c = 1

(b) **Main.java** program provided below which contains a main method that creates multiple instances of the **Shape** class using different constructors. Your task is to complete **Shape.java** by adding necessary constructors so that **Main.java** runs correctly. [5 Marks] [CO1]

1. Learn about this.
2. pass an object then assign the valu

Main.java

```
    public class Main {
    public static void main(String[] args) {
    // set default value 10 to width, height and length
        Shape shape1 = new Shape();

    //set width = 5, height = 10, length = 0
        Shape shape2 = new Shape(5, 10);
        //set width = 3, height = 6, length = 9
        Shape shape3 = new Shape(3, 6, 9);

       // set width, height and length to shape3's width, height and length respectively.
        Shape shape4 = new Shape(shape3);

    }
}
```

Shape.java

```
   class Shape {
   int length, width, height;
   // add necessary constructors
    }
```

2. Consider the following class named **Burger** representing a generic burger. Each burger should have a **name** and a **size** as attributes. The **Burger** class has a *prepare()* method that always prints a base message with the burger's name and size.

   Now, you have two subclasses of burgers: **BeefBurger** and **VeggieBurger**. Each type has two additional attributes (field) named **flavor** (data type: **String**) and **price** (data type: **int**). Both burgers have a *prepare()* method which prints the base message at first and then prints the additional details (**flavor** and **price**).

   Based on the given code and expected output, write the necessary code so that the expected output is produced. Note that you **cannot change** any given code. If you **complete the code and run**, then the following output should be found: [3+3=6 Marks] [CO1]

```
Output:
Preparing Double Decker Large burger
Flavor: Spicy
Price: 400

Preparing Mushroom Delight Small burger
Flavor: Sweet
Price: 300
```

1. extends
2. @Override
3. super in constructor.
3. super in method for base massag

MainBurger.java

```java
class Burger {
  private String name;
  private String size;

    public Burger(String name, String size) {
        this.name = name;
        this.size = size;
    }
    // always print the base message
    public void prepare() {
        System.out.println("Preparing " + name + " " + size + " burger");
    }
}

class MainBurger {
    public static void main(String[] args) {

        Burger beefBurger = new BeefBurger ("Double Decker", "Large","Spicy", 400);
        beefBurger.prepare();

        Burger veggieBurger = new VeggieBurger ("Mushroom Delight", "Small", "Sweet",
            300);
       veggieBurger.prepare();
    }
}
```

3. The given Java code defines a **Vehicle** class with two methods: *start()* and *move()*. The **Bus** and **Cycle** classes inherit from **Vehicle**. The **VehicleTest** class implements the *main()* method. Follow the instructions. [2+2+2=6 Marks] [CO1]

   - **Identify and correct errors** in the given Java code.
   - You **cannot modify the Vehicle class**.
   - You **cannot add new methods** to any class.
   - Add necessary correction to the *main()* method and the sub classes (Bus and Cycle).
   - **Provide the output** of the corrected code.

1. Final Methods Cannot Be Overridden
2. Subclass-Specific Methods Require Cas
3. Proper Syntax for Casting
   ((SubClass) object).method();
   ❌ (SubClass)object.method()

```
public class Vehicle {

   void start(){
      System.out.println("Vehicle is
         starting");
   }

   final void move(){
      System.out.println("Vehicle is
         moving");
   }
}
```

```
class Bus extends Vehicle{

   void move(){
      System.out.println("Bus is moving");
   }

   void needFuel(){
      System.out.println("Bus needs fuel");
   }

   void start(){
      System.out.println("Bus is
         starting");
   }
}
```

```
class Cycle extends Vehicle {
   void pedal(){
      System.out.println("Pedal the
         cycle");
   }
}
```

```
class VehicleTest{

   public static void main(String[] args) {
      Vehicle bus = new Bus();
      bus.start();
      bus.needFuel();
      Vehicle cycle = new Cycle();
      cycle.move();
      cycle.pedal();
   }
}
```

4. (a) Consider the following code **FlightBooking.java**.

FlightBooking.java

```
public class FlightBooking {
    private String passengerName;
    private int seatNumber;
    private boolean confirm = false;
    public FlightBooking(String passengerName, int seatNumber) {
        this.passengerName = passengerName;
        this.seatNumber = seatNumber;
    }
}
```

Based on the class, a **main** class **FTest.java** is written as follows:

FTest.java

```
class FTest{
  public static void main(String[] args) {
     FlightBooking f1 = new FlightBooking("John", 12);
     FlightBooking f2 = new FlightBooking("Maria", 3);
     if( f1.confirm == false ){
        f1.confirm = true;
        f1.seatNumber = 32;
     }
     System.out.println("seat number of Maria: " + f2.seatNumber );
     System.out.println("seat number of John: " + f1.seatNumber);
  }
}
```

Is there any error in *main()* method? If yes, modify the codes in **"FlightBooking.java"** and **"FTest.java"**. Note that you **cannot** change access modifiers of any instance variables in **FlightBooking.java**.

[1+4 = 5 Marks] [CO1]

(b) Consider the following **Box.java** and **Main.java**. What is the output of the program? Justify your answer. [3 Marks] [CO1]

```
public class Box {
    double width;
    double height;
    double depth;

    public void print(){
        System.out.println(width);
        System.out.println(height);
        System.out.println(depth);
    }
}
```

```
public class Main {
    public static void main(String[]
        args) {
        Box b1= new Box();
        b1.width=1;
        b1.height=2;
        b1.depth=3;
        Box b2 = b1;
        b1=null;
        b2.print();
    }
}
```

1. b1 set everything.
2. b2 assigned b1 reference address.
3. b1 set null
4. so the object's reference went on b2 variable and b1 set null.