United International University
CSI 211: Object Oriented Programming, Fall 2017
Mid Term Exam
Total Marks: 30, Time: 1 hour 45 minutes

Answer any 5 out of the following 6 questions $(5 \times 6 = 30)$**.**

1. Find out if the following JAVA classes have any error. **List the errors** if any. **Fix the code and rewrite** after the errors list. You cannot delete any line of code. However, you are allowed to edit or add any code as per requirement. After the code is fixed, write down the **output** of the correct code. [2+2+2]

```java
package Test.packA;
public class Animal {
    int legs;
    protected int eyes;
    public Animal() {
        System.out.println("Default constructor of Animal");
    }
    private Animal(int legs, int eyes) {
        System.out.println("Constructor of Animal");
        this.legs = legs;
        this.eyes = eyes;
    }
    public String toString() {
        return "legs=" + legs + ", eyes=" + eyes;
    }
    public int getLegs() {
        return legs;
    }
    public void setLegs(int legs) {
        this.legs = legs;
    }
}
```

```java
package Test.packB;
import Test.packA.*;
class Dinosaur extends Animal{
    boolean isExtinct;
    public Dinosaur() {
        System.out.println("Default Constructor of Dinosaur");
    }
    public Dinosaur(int legs, int eyes,boolean isExtinct) {
        super();
        super.legs = legs;
        super.eyes = eyes;
        this.isExtinct = isExtinct;
        System.out.println("Constructor of Dinosaur");
    }
    public void accident(){
        if(eyes>0) eyes -= 1;
    }
    public String toString() {
        return "legs= " + getLegs() + ", eyes= " +
            eyes + ", isExtinct=" + isExtinct;
    }
}
```

```java
package Test.packA;
public class Panda extends Animal{
    boolean hasLice;
    public Panda() {
        System.out.println("Default Constructor of Panda");
    }
    public Panda(int legs, int eyes,boolean hasLice) {
        super(legs, eyes);
        this.hasLice = hasLice;
        System.out.println("Constructor of Panda");
    }
    public void accident(){
        if(legs>0) legs -= 1;
        if(eyes>0) eyes -= 1;
    }
    public String toString() {
        return "legs= " + getLegs() + ", eyes= " +
            eyes + ", hasLice=" + hasLice;
    }
}
```

```java
package Test.packB;
import Test.packA.*;
class Main{
    public static void main(String[] args) {
        Panda po = new Panda(4, 2, true);
        po.accident();
        System.out.println(po);

        Dinosaur Dino = new Dinosaur(4, 2, true);
        System.out.println(Dino);
    }
}
```

Table 1: Q. 1

2. Create a class called **Time** in package **timepack** that has 3 member variables hour, minute and second. Complete the class by adding necessary methods to it such that the main method in **Table 2** provides the expected output. You may add more classes and/or methods if deemed necessary. However, you are not allowed to modify the main method itself. [6]

```java
class TimeTest {
    public static void main(String [] args) {
        Time t1 = new Time();
        Time t2 = new Time(12, 20);
        Time t3 = new Time(23, 11, 12);
        System.out.println(t1);
        System.out.println(t2);
        System.out.println(t3);
        timeSwap(t1, t3);
        System.out.println("After swap");
        System.out.println(t1);
        System.out.println(t2);
        System.out.println(t3);
        System.out.println("After add 1");
        t3.add(t1);
        System.out.println(t3);
        System.out.println("After add 2");
        t2.add(45, 0);
        System.out.println(t2);
        System.out.println("After add 3");
        t1.add(1, 10, 20);
        System.out.println(t1);

    }
}
```

```
Expected Output:
12:00:00 AM
11:20:00 PM
11:11:12 PM
After swap
11:11:12 PM
11:20:00 PM
12:00:00 AM
After add 1
11:11:12 AM
After add 2
12:05:00 AM
After add 3
12:21:22 PM
```

Table 2: Q. 2

3. (a) Consider the three classes *Super*, *Sub* and *SubSub* and find the output for the main method. [4]

```java
class Super {
    void print(int a) {
        System.out.println("super: " + a);
    }
    void print() {
        System.out.println("inside sub");
    }
}

class Sub extends Super {
    void print(int a) {
        System.out.println("sub: " + a);
    }
    void print() {
        System.out.println("inside subsub");
    }
}
```

```java
class SubSub extends Sub {
    void print(int a) {
        System.out.println("super: " + a);
    }
    void print() {
        System.out.println("inside sub");
    }
}

class Main {
    public static void main(String[] args) {
        Super x = new Super();
        Sub y = new Sub();
        SubSub z = new SubSub();

        x.print(2);
        x=y;
        x.print();
        y=z;
        y.print(3);
        z.print();
    }
}
```

Table 3: Q. 3(a)

(b) Suppose we have two classes: *InkPrinter* and *LaserPrinter*. Both the classes inherit an abstract class, *Printer*. This class has a concrete method, *getPaper()* and an abstract method *print()*. *InkPrinter* and *LaserPrinter* use different approaches to print but use same way to get papers. Now explain, why abstract class is necessary in this context. [2]

4. (a) What is the difference between following two declarations in Java? [1]

   i. int c [ ], x

   ii. int [ ] c, x

   (b) i. Find out if the following JAVA classes in **Table 4** have any error. **List the errors** if any. **Fix the code and rewrite** after the errors list. You cannot delete any line of code. However, you are allowed to edit or add any code as per requirement. [4]

```java
class ClassRules implements Rules{
   public void ruleOne() {
        System.out.println("Class rule 1");
   }
   void ruleTwo() {
        System.out.println("Class rule 2");
   }
   void show(){
       System.out.println("Class Rules");
   }
}
```

```java
interface Rules {
    int var = 10;
    void ruleOne();
    void ruleTwo();
}
interface Points{
    void addPoints();
}
```

```java
class GameRules implements Rules{
    public void ruleOne(){
        System.out.println("Game rule 1");
        var += 10;
    }
}
```

```java
class Main{
    public static void main(String[] args) {
        Rules cr = new ClassRules();
        System.out.println("Variable = " +
                        Rules.var);
        cr.show();
    }
}
```

Table 4: Q. 4(b)

   ii. Write a class called **Game** that has all the methods ruleOne(), ruleTwo() and addPoints() by inheriting appropriate interface(s) and/or class(s) such that the following code snippet can be executed correctly.

   Game g1 = new Game();
   g1.addPoints();
   System.out.print(g1.point);

   Initially, point should be 0 for any game. It increases by 5 every time addPoints() is called. [1]

5. Suppose you are developing a software for calculating earnings of different types of teaching assistants. There are 2 types of TeachingAssistant: Grader, LabAssistant. A grader assists teacher by checking assignment scripts, whereas a lab assistant helps the teacher by attending to students in the lab.

   Class **TeachingAssistant** has two attributes: *name* and *id*. The constructor of TeachingAssistant class initializes *name* and *id* with **this** reference keyword. There is one method named `void printEarnings()` which prints the *name* and *id* of the TeachingAssistant. The classes that extend TeachingAssistant are **Grader** and **LabAssistant**.

   Grader class overrides printEarnings() method. This method first invokes parent method and then prints the earning of grader by multiplying the number of graded assignments with per-assignment-pay. To do so, you should include two instance variables in Grader class definition: *count*, *payPerAssignment*.

LabAssistant class also overrides printEarnings() in similar way except that a lab assistant is paid on an hourly basis. The earning is calculated by multiplying total number of hour he spends in the lab by the hourly payment. In order to do so, include two instance variables in LabAssistant class: *hour* and *hourlyPay*.

Your task is to write the complete code of these three classes. Use appropriate Access Modifiers (private, protected or public) while declaring instance variables and methods. [6]

6. (a) Write a code fragment to create the following multidimensional integer array. [2]

0			
1	2		
3	4	5	
6	7	8	9

(b) Consider the classes *Plate* and *DinnerPlate* in **Table 5** and find the output for the provided main method. [1]

```java
class Plate {
    Plate(int i) {
        System.out.println("Plate " + i);
    }
    Plate(String s) {
        System.out.println("Plate " + s);
    }
    void info() {
        System.out.println("Info of Plate");
    }
}

class DinnerPlate extends Plate {
    DinnerPlate(int i) {
        super("Plate");
        System.out.println("DinnerPlate " + i);
    }
    void info() {
        System.out.println("Info of DinnerPlate");
    }
}
```

```java
public class Test {
    public static void main(String[] args) {
        DinnerPlate dp;
        Plate pl;
        pl = new DinnerPlate(5);
        pl.info();
        dp = (DinnerPlate) pl;
        dp.info();
    }
}
```

Table 5: Q. 6(b)

(c) Suppose you are building a software for a Grocery Shop. Now, write a Java class named **Product**. It has two attributes **name** and **price** with types respectively **String** and **Floating point number**. The constructor of Product class initializes **name** and **price** with **this** reference keyword. Create another class named **ProductMain** and implement the main method. In the main method create a **new Product** with name **F** and price **1000**. [3]