BSCCS2005: Practice Assignment with Solutions
Week 4

1. Consider the code given below. [MCQ:2 points]

```
public interface Shape{
    public double area();
    public default double volume() {
        return -1.0;
    }
}

public interface Printable{
    public default void print() {
        System.out.println("not implemented");
    }
}

public class Rectangle implements Shape, Printable{
    private double w, h;
    public Rectangle(double w_, double h_) {
        w = w_;
        h = h_;
    }
    public double area() {
        return w * h;
    }
    public void print() {
        System.out.print(area() + " ");
        System.out.print(volume());
    }
}

public class FClass{
    public static void main(String[] args) {
        Rectangle r = new Rectangle(20.0, 50.0);
        r.print();
    }
}
```

What will be the output?

- ◯ `1000.0` followed by a runtime error
- ◯ `not implemented`
- √ `1000.0 -1.0`
- ◯ It generates compiler error

Solution: In `Rectangle` class, the method `print()` from `Printable` and the method `area()` from `Shape` are overridden. Since, the object is of `Rectangle` type, the methods `print()` and `area()` invoke the implementations from `Rectangle` class. However, since `volume()` is not overridden, the default implementation will be inherited. Thus, the output is `1000.0 -1.0`.

2. Consider the code given below. [MCQ:2 points]

```
public interface Flyable{
    public void fly();
    public default void travel() {
        System.out.println("travel with wings");
    }
}

public interface Movable{
    public void move();
    public default void travel() {
        System.out.println("travel with legs");
    }
}

public class Bird implements Flyable, Movable{
    public void fly() {
        System.out.println("fly with wings");
    }
    public void move() {
        System.out.println("move with legs");
    }
}

public class FClass{
    public static void main(String[] args) {
        Bird b = new Bird();
        b.fly();
        b.move();
        b.travel();
    }
}
```

What will be the output?

- ◯ `fly with wings`
  `move with legs`
- ◯ `fly with wings`
  `move with legs`
  `travel with wings`
- ◯ `fly with wings`
  `move with legs`
  `travel with legs`

√ It generates a compiler error

Solution: Call to the `travel()` method in `class Bird` is ambiguous, since it inherits two different versions of default implementations from two different interfaces.

3. Consider the code given below. [MCQ:2 points]

```
public interface Comparable{
    public abstract int comp(Comparable x);
}
public class Name implements Comparable{
    private String fname;
    private String lname;

    public Name(String fname, String lname){
        this.fname = fname;
        this.lname = lname;
    }
    public int comp(Comparable x) {
        if(lname.compareTo(((Name)x).lname) == 0)
            return fname.compareTo(((Name)x).fname);
        return lname.compareTo(((Name)x).lname);
    }
    public void print() {
        System.out.println(fname + " " + lname);
    }
}
public class FClass{
    public static void sort(Comparable[] names) {
        for(int i = 0; i < names.length - 1; i++) {
            for(int j = 0; j < names.length - i - 1; j++) {
                if(names[j].comp(names[j + 1]) > 0) {
                    Comparable tname = names[j];
                    names[j] = names[j + 1];
                    names[j + 1] = tname;
                }
            }
        }
    }
    public static void main(String[] args) {
        Name[] names = new Name[] {new Name("Charlotte", "Brown"),
                                   new Name("Ava", "Smith"),
                                   new Name("Emma", "Williams"),
                                   new Name("Olivia", "Smith"),
                                   new Name("Emma", "Johnson")};
        sort(names);
        for(int i = 0; i < names.length; i++)
            names[i].print();
    }
```

```
}
```

What will be the output?

◯
```
Ava Smith
Charlotte Brown
Emma Johnson
Emma Williams
Olivia Smith
```

√
```
Charlotte Brown
Emma Johnson
Ava Smith
Olivia Smith
Emma Williams
```

◯
```
Ava Smith
Charlotte Brown
Emma Williams
Emma Johnson
Olivia Smith
```

◯ It generates an error

Solution: As per the `comp()` implementation in `Name` class, the names will be first sort by `lname`, and in case of equality of `lname`, sort by `fname`.

4. Consider the code given below. [MSQ:2 points]

```
1 public abstract class Polygon{
2   public abstract int perimeter();
3 }
4 public interface Shape{
5   public abstract int area();
6 }
7 public class Rectangle extends Shape implements Polygon{
8   public int area( ){
9     System.out.println("length*breadth");
10  }
11   public int perimeter( ){
12     System.out.println("length + breadth");
13   }
14 }
```

Identify the error in the code.

- ◯ Line 2: perimeter( ) is not defined
- ◯ Line 5: area( ) is not defined
- √ Line 7: incorrect usage of extends and implements
- ◯ Line 7: class Rectangle is not declared as abstract
- ◯ Line 8: area( ) has taken arguments
- √ Line 10: missing return statement
- √ Line 13: missing return statement

Solution:

- Line 7: public class Rectangle extends Polygon implements Shape
- area(), perimeter() should return int

5. Consider the program given below and choose the correct option.

[MCQ:2 points]

```
public abstract class NewYear {
    abstract String resolution();
    public NewYear(){
        System.out.println("Resolution: ");
    }
}
public class Year_2022 extends NewYear{
   public String resolution() {
      return "Walk up early, exercise and take shower everyday.";
   }
   public static void main(String args[]) {
      System.out.print(new Year_2022().resolution());
   }
}
```

◯ We cannot declare a constructor of the `abstract` class.

◯ Compile time error.

√ Output of this program is:
Resolution:
Walk up early, exercise and take shower everyday.

◯ Output of this program is:
Walk up early, exercise and take shower everyday.

Solution: The compiler adds a constructor to an `abstract` class if a programmer does not declare it explicitly. Constructor of an abstract class is necessary because the subclass constructors calls the base class constructor by default.

6. Consider the program given below and choose the correct option.

[MCQ:2 points]

```
public interface Rectangle{
   abstract int areaRectangle(int length,int breadth);
}
public interface Circle{
   abstract int areaCircle(int radius);
}
public interface Shape extends Rectangle,Circle{

}
public class TwoDimension implements Shape{
   public int areaRectangle(int length,int breadth){
      return length*breadth;
   }
   public int areaCircle(int radius){
      return (int)(Math.PI*Math.pow(radius,2));
   }
}
public class Example{
   public static void main(String[] args){
      TwoDimension ref=new TwoDimension();
      System.out.println("The area of the rectangle is: "+ref.areaRectangle(5,2)
      +"\n"+"The area of the Circle is: "+ref.areaCircle(5));
   }
}
```

- ◯ This program generates a compile time error because an interface cannot be extended.
- ◯ The area of the rectangle is: 10 followed by a runtime error.
- √ The area of the rectangle is: 10
  The area of the Circle is: 78
- ◯ The area of the rectangle is: 10
  The area of the Circle is: 75

Solution: Classes implement interfaces, and interfaces are allowed to inherit from other interfaces.

7. Consider the Java program given below and predict the output. [MCQ : 2 points]

```
public abstract class Base{
    public Base(){
        System.out.println("Base class constructor");
    }
}
public class Sub extends Base{
    public Sub(){
        System.out.println("Subclass constructor");
    }
}
public class Example {
    public static void main(String[] args){
        Base base=new Sub();
    }
}
```

- ◯ Compilation failed, you cannot define a constructor for abstract class.
- ◯ Compilation failed because of the statement below.
  Base base=new Sub();
- √ Base class constructor
  Subclass constructor
- ◯ Run time error

Solution: Option 3
You can define constructor in the abstract class. It executes when its subclass gets instantiated.

8. Consider the Java program given below and choose the correct option.

[MCQ : 2 points]

```
public interface City{
    public abstract void travel(String name);
}
public class Mumbai implements City{
    public void travel(String name) {
        System.out.println(name+" Travelling to Mumbai");
    }
}
public class Hyderabad implements City{
    public void travel(String name) {
        System.out.println(name+" Travelling to Hyderabad");
    }
}
public class Traveller{
    private City city;
    private String name;
    public Traveller(City city,String name) {
        this.city = city;
        this.name = name;
    }
    public void journey(){
        city.travel(name);
    }

}
public class Example{
    public static void main(String[] args) {
        Traveller traveller1=new Traveller(new Hyderabad(),"Johny");
        traveller1.journey();
        Traveller traveller2=new Traveller(new Mumbai(),"Virat");
        traveller2.journey();
    }
}
```

- ◯ Compilation failed because an interface cannot be a member of another class.
- ◯ Compilation failed because Traveller constructor will not accept Hyderabad and Mumbai class objects.
- √ Johny Travelling to Hyderabad
  Virat Travelling to Mumbai
- ◯ null Travelling to null
  null Travelling to null

Solution: interface can hold its implemented classes objects.
`traveller1.journey();` generates `Johny Travelling to Hyderabad` as output.
`traveller2.journey();` generates `Virat Travelling to Mumbai` as output.

9. Choose all the correct option(s) regarding the code given below.

```
public interface Academics{
    default void getName(){
        System.out.println("Academics : getName");
    }
    default void printName(){
        System.out.println("Academics : printName");
    }
}

public class University{
    private int univ_id;

    private class College implements Academics{
        private String college_name;

        public void getName(){
            System.out.println(college_name);
        }

        public void getID(){
            System.out.println(univ_id);
        }

        College(String name){
            college_name = name;
        }
    }
    public College getReference(){
        return new College("IITMadras");
    }
}
public class FClass{
        public static void main(String[] args) {
            University uni = new University();
            Academics acad = uni.getReference();
            acad.getName();
            acad.getID();
        }
}
```

[MSQ:2points]

◯ This code generates output:
IITMadras 0

◯ This code generates compilation error because default method `getName` is overridden.

◯ This code generates compilation error because method `getReference` does not return a valid `College` type object.

◯ This code generates compilation error because private instance variable `univ_id` of outer class can not be directly accessed inside method `getID` of inner class.

√ This code generates compilation error because `getID` method of class `College` is not accessible by `acad` object..

10. Consider the Java program given below.

[MCQ : 2 points]

```
import java.util.Date;
public class Timer{
    private Employee e;
    public Timer(Employee e){
        this.e=e;
    }
    public void start(Date date){
        if(date.getDay()==6 || date.getDay()==0){
            e.notification1();
        }
        else{
            e.notification2();
        }
    }
}
public class Employee {
    public void check(){
        Timer obj=new Timer(this);
        Date d=new Date();
        obj.start(d);
    }
    public void notification1(){
        System.out.println("Happy weekend...");
    }
    public void notification2(){
        System.out.println("Today is a working day, stay at work.");
    }
    public static void main(String[] args) {
        Employee e=new Employee();
        e.check();
    }
}
```

Note that the `getDay()` method returns an integer value from 0 to 6. Each integer represents a day of the week as follows:
0: Sunday, 1: Monday, 2: Tuesday, 3: Wednesday, 4: Thursday, 5: Friday, 6: Saturday.
If this program in run on a Sunday, what will be the output?

- ◯ This code generates output: `Today is a working day, stay at work.`
- ◯ This code generates a compile time error.
- ◯ This code generates no output.

√ This code generates the output: Happy weekend...

Solution: Assume if you execute this program on Sunday, getDay() method will return 0.
If getDay() is equal to 0 or 6 it will call notification1(), otherwise notification2().