CSE 201 ADVANCED PROGRAMMING

MidSem Examination [2021-22] [Total 20 marks, Duration: 1hr]

[Section A]

Answer any SEVEN of the following. Don’t spend more than 3 minutes on any of these questions. **[7x1=7 marks]**

Q1) Why and where do we need @Override annotation?

Q2) In what all ways can we overload a function?

Q3) Name the four types of access modifiers. Discuss how they modify accessibility.

Q4) What is the difference between composition and association class relationships?

Q5) What is the difference between dependency and association class relationships?

Q6) How does a setter function help in Encapsulation?

Q7) Differentiate between static and instance variables in a class.

Q8) Differentiate between static and non-static methods in a class.

Q9) In what all ways can we initialize instance variables of a class?

Q10) What do you mean by reference and object types of a variable? [NOTE: They are also known as declared and actual types, respectively]

[Section B]

Answer any TWO of the following. Don’t spend more than 6 minutes on any of these questions. **[2x3=6 marks]**

Q1) When does a compiler link a function to a function call? When it doesn’t, what links them and how?

Q2) Write a program to illustrate the importance of super keyword in avoiding null pointer exception.

Q3) In what all ways can we make a class immutable? Give an example of an immutable class.

[Section C]

Answer any ONE of the following. **[1x7=7 marks]**

Q1) Write a program that illustrates all three kinds of class relationships.

Q2) Identify errors in the following code and rectify them:

```
interface Animal{
    public void eat(Animal A);
    public void move(Animal A);
}

abstract class Bird implements Animal{
    abstract public void eat(Animal A);
    abstract public void move(Animal A);
}

class Parrot extends Bird{
    protected String name;
    Parrot(String name){
        name=name;
    }
    @Override
    public void eat(Bird A){
        System.out.println(this.name+ " eats with " + ((Parrot)A).name);
    }
    @Override
    static void move(Animal A) {
        System.out.println(this.name+ " flies with " + ((Parrot)A).name);
    }
}

class Crow extends Parrot{
    Crow(String name){
        super();
    }
    @Override
    public void eat(Animal A) {
        System.out.println(super.name+ " eats with " + ((Parrot)A).name);
    }
    @Override
    public void move(Animal A) {
        System.out.println(super.name+ " flies with " + ((Parrot)A).name);
    }
}

class Main2 {
    public static void main2(String[] args) {
        Animal Ap1=new Animal("parrot1");
        Bird Ac1 = new Crow("crow1");
        Parrot Ap2 = new Parrot("parrot2");
        Crow Ac2 = new Crow("crow2");
        Bird [] birds = {Ap1,Ac1,Ap2,Ac2};
        birds[0].eat(birds[1]);
        birds[0].move(birds[2]);
        birds[2].eat(birds[3]);
        birds[3].move(birds[1]);
    }
}
```

Expected Output:

```
parrot1 eats with crow1
parrot1 flies with parrot2
parrot2 eats with crow2
crow2 flies with crow1
```