ASSIGNMENT-5

COURSE CODE-CS261

COURSE NAME-OBJECT ORIENTED DESIGN AND PROGRAMMING

Q1) Write a program to make a class Animal abstract , and make an abstract method sound(). Makeanother 2 class Wolf and dog which would extend class Animal and replicate sound of Wolf and dog ,if wolf sound is woof and if dog sound is bowl.

ANS.

```
abstract class Animal
{
	abstract public void sound();
}
class dog extends Animal
{
	public void sound()
	{
		System.out.println("bowl");
	}
}
class wolf extends Animal
{
	public void sound()
	{
		System.out.println("woof");
	}
}
class animal
{
	public static void main(String args[])
	{
		Animal d=new dog();
```

```
        Animal w=new wolf();
        d.sound();
        w.sound();
    }
}
```

Q2) Write a program to make a class Shape abstract , and make an abstract method area(). Make another 2 class Square and Rectangle which would extend class shape() . Then calculate the shape of Square and Rectangle , length of sides to be given by the user

ANS.

```
import java.util.*;
abstract class Shape
{

    abstract public void Area(int l,int b);
}
class Square extends Shape
{
    public void Area(int a,int b)
    {
        int area;
        area=a*b;
        System.out.println("Area of the square is : "+area);
    }
}
class Rectangle extends Shape
{
    public void Area(int l,int b)
    {
        int area;
        area=l*b;
        System.out.println("Area of the rectangle : "+area);
```

```
	}
}
class shapes
{
	public static void main(String args[])
	{
		int s,l,b;
		Scanner sc=new Scanner(System.in);
		System.out.println("Enter the side of the square");
		s=sc.nextInt();
		System.out.println("Enter the length and breadth of the rectangle");
		l=sc.nextInt();
		b=sc.nextInt();
		Shape ob1=new Square();
		Shape ob2=new Rectangle();
		ob1.Area(s,s);
		ob2.Area(l,b);
	}
}
```

Q3) Predict the output and justify

```
//Interface
interface Multiply{
//abstract methods
public abstract int multiplyTwo(int n1, int n2);
/* We need not to mention public and abstract in interface
* as all the methods in interface are
* public and abstract by default so the compiler will
* treat this as
* public abstract multiplyThree(int n1, int n2, int n3);
*/
int multiplyThree(int n1, int n2, int n3);
/* Regular (or concrete) methods are not allowed in an interface
* so if I uncomment this method, you will get compilation error
* public void disp(){
*
```

```
System.out.println("I will give error if u uncomment me");
* }
*/
}
class Demo implements Multiply{
public int multiplyTwo(int num1, int num2){
return num1*num2;
}
public int multiplyThree(int num1, int num2, int num3){
return num1*num2*num3;
}
public static void main(String args[]){
Multiply obj = new Demo();
System.out.println(obj.multiplyTwo(3, 7));
System.out.println(obj.multiplyThree(1, 9, 0));
}
}
```

ANS.

The output of the given program will be “21 and 0”.In java an instance of interface cannot be created but it can be instantiated using instance of the other class or it can refer to the object of it’s implementing class and abstract method must be overridden to be instantiatiated.

Q4) Predict output and justify

```
abstract class AbstractDemo{
public void myMethod(){System.out.println("Hello");
}
abstract public void anotherMethod();
}
public class Demo extends AbstractDemo{
public void anotherMethod() {
System.out.print("Abstract method");
```

```
}
public static void main(String args[])
{
AbstractDemo obj = new AbstractDemo();
obj.anotherMethod();
}
}
```

ANS.

The output of the above program will be compile time error because we cannot create the object of the abstract class.

Q5) Can multiple inheritance occurs in case of interface , explain with example

ANS.

Yes,it is possible to do multiple inheritance in the case of interface.A class can implement two or more interfaces.In case both the implemented interfaces contain default method with same method signature,the implementing class should explicitly specify which default method is to be used or it should override the default method.

Example-

```
interface A
{
        void print();
}
interface B
{
       void message();
}
class C implements A,B
{
       public void print()
       {
```

```
        System.out.println("Hello");
    }
    public void message()
    {
        System.out.println("Hi");
    }
}
class Test
{
    public static void main(String args[])
    {
        C ob=new C();
        ob.print();
        ob.message();
    }
}
```

OUTPUT-

Hello

Hi

Q6) Predict the output

```
interface Printable{
void print();
}
interface Showable{
void print();
}
class TestInterface3 implements Printable, Showable{
public static void main(String args[]){
TestInterface3 obj = new TestInterface3();
obj.print();
}
}
```

ANS.

The output of the above code will be “compile time error” because the print function declared in the the interfaces(Printable,Showable) is not defined in any other class or main class.Or,TestInteface3 is not abstract and does not override abstract method print().

Q7) Predict the output

```
abstract class Bike{abstract void run();
}
class Honda4 extends Bike{
public static void main(String args[]){
Bike obj = new Honda4();
obj.run();
}
}
```

ANS.

The output of the above code will be a “compile time error” because abstract methods do not have body so,it must be overridden for proper functioning of program.

Q8) Predict the output

```
abstract class Bike{
Bike(){System.out.println("bike is created");}
abstract void run();
void changeGear(){System.out.println("gear changed");}
}
//Creating a Child class which inherits Abstract class
class Honda extends Bike{
void run(){System.out.println("running safely..");}
}
//Creating a Test class which calls abstract and non-abstract methods
class TestAbstraction2{
public static void main(String args[]){
```

Bike obj = new Honda();
obj.run();
obj.changeGear();
}
}
ANS.
The output of the above code will be-

bike is created
running safely..
gear changed

Q9)
interface Drawable{
void draw();
default void msg(){System.out.println("default method");}
}
class Rectangle implements Drawable{public void draw()
{System.out.println("drawing rectangle");}
}
public class TestInterfaceDefault{
public static void main(String args[]){
Drawable d=new Rectangle();
d.draw();
d.msg();
}}
ANS.

The output of the above code wil be-

drawing rectangle
default method

NAME-AMAN KUMAR KANOJIA
ROLL NO.-201851014
SECTION -1
GROUP-1A