Java - Sample Inheritance Problem Set

Md. Mohsin Uddin

East West University

mmuddin@ewubd.edu

June 25, 2019

Abstract class example I

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

When the above code is compiled and executed, it produces the following result:

```
running safely
```

Abstract class example II

```
abstract class Shape{
    abstract void draw();
}
class Rectangle extends Shape{
    void draw(){System.out.println("drawing rectangle");}
}
class Circle extends Shape{
    void draw(){System.out.println("drawing circle");}
}
class TestAbstraction1{
    public static void main(String args[]){
        Shape s=new Circle();
        s.draw();
    }
}
```

When the above code is compiled and executed, it produces the following result:

```
drawing circle
```

Abstract class example III : Part I

```
abstract class Bank{
    abstract int getRateOfInterest();
}
class SBI extends Bank{
    int getRateOfInterest(){return 7;}
}
class PNB extends Bank{
    int getRateOfInterest(){return 8;}
}
class TestBank{
    public static void main(String args[]){
        Bank b;
        b=new SBI();
        System.out.println("Rate of Interest is: "
                    +b.getRateOfInterest()+" %");
        b=new PNB();
        System.out.println("Rate of Interest is: "
                    +b.getRateOfInterest()+" %");
    }
```

Abstract class example III : Part II

```
}
```

When the above code is compiled and executed, it produces the following result:

```
Rate of Interest is: 7 %
Rate of Interest is: 8 %
```

Abstract class example IV : Part I

```
abstract class Bike{
    Bike(){System.out.println("bike is created");}
    abstract void run();
    void changeGear(){System.out.println("gear changed");}
}
class Honda extends Bike{
    void run(){System.out.println("running safely..");}
}
class TestAbstraction2{
    public static void main(String args[]){
        Bike obj = new Honda();
        obj.run();
        obj.changeGear();
    }
}
```

Abstract class example IV : Part II

When the above code is compiled and executed, it produces the following result:

```
bike is created
running safely..
gear changed
```

Abstract class example V : Part I

```
interface A{
    void a();
    void b();
    void c();
    void d();
}

abstract class B implements A{
    public void c(){System.out.println("I am c");}
}

class M extends B{
    public void a(){System.out.println("I am a");}
    public void b(){System.out.println("I am b");}
    public void d(){System.out.println("I am d");}
}

class Test5{
    public static void main(String args[]){
```

Abstract class example V : Part II

```
        A a=new M();
        a.a();
        a.b();
        a.c();
        a.d();
    }
}
```

When the above code is compiled and executed, it produces the following result:

```
I am a
I am b
I am c
I am d
```

Abstract class example VI : Part I

```
abstract class Person {
    private String name;
    private String gender;
    public Person(String nm, String gen){
        this.name=nm;
        this.gender=gen;
    }

    //abstract method
    public abstract void work();
    @Override
    public String toString(){
        return "Name="+this.name+"::Gender="+this.gender;
    }

    public void changeName(String newName) {
        this.name = newName;
    }
}
```

Abstract class example VI : Part II

```
class Employee extends Person {
    private int empId;
    public Employee(String nm, String gen, int id) {
        super(nm, gen);
        this.empId=id;
    }
    @Override
    public void work() {
        if(empId == 0){
                System.out.println("Not working");
        }else{
                System.out.println("Working as employee!!");
        }
    }
    public static void main(String args[]){
        //coding in terms of abstract classes
        Person student = new Employee("Rahim","Female",0);
```

Abstract class example VI : Part III

```
        Person employee = new Employee("Amlan","Male",123);
        student.work();
        employee.work();
        employee.changeName("Amlan Talukder");
        System.out.println(employee.toString());
    }

}
```

When the above code is compiled and executed, it produces the following result:

```
Not working
Working as employee!!
Name=Amlan Talukder::Gender=Male
```

Interface example I: Part I

```
interface Printable{
    void print();
}
interface Showable{
    void show();
}
class A7 implements Printable,Showable{
    public void print(){
        System.out.println("Hello");
    }
    public void show(){
        System.out.println("Welcome");
    }
    public static void main(String args[]){
        A7 obj = new A7();
        obj.print();
        obj.show();
    }
}
```

Interface example I: Part II

When the above code is compiled and executed, it produces the following result:

```
Hello
Welcome
```

Interface example II: Part I

```
interface Printable{
    void print();
}
interface Showable extends Printable{
    void show();
}
class TestInterface4 implements Showable{
    public void print(){System.out.println("Hello");}
    public void show(){System.out.println("Welcome");}
    public static void main(String args[]){
        TestInterface4 obj = new TestInterface4();
        obj.print();
        obj.show();
    }
}
```

Interface example II: Part II

When the above code is compiled and executed, it produces the following result:

```
Hello
Welcome
```

Java 8 Interface Feature: default method : example III: Part I

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

Java 8 Interface Feature: default method : example III: Part II

```
}
```

When the above code is compiled and executed, it produces the following result:

```
drawing rectangle
default method
```

Java 8 Interface Feature: static method : example IV: Part I

```
interface Drawable{
    void draw();
    static int cube(int x){return x*x*x;}
}
class Rectangle implements Drawable{
    public void draw(){
        System.out.println("drawing rectangle");
    }
}
class TestInterfaceStatic{
    public static void main(String args[]){
        Drawable d=new Rectangle();
        d.draw();
        System.out.println(Drawable.cube(3));
    }
}
```

Java 8 Interface Feature: static method : example IV: Part II

When the above code is compiled and executed, it produces the following result:

```
drawing rectangle
27
```

References

DEITEL, Java How to Program, 11/e

Java: the complete reference, Herbert Schildt, McGraw-Hill Education Group