Swapnanil Dutta Roll:12

Question: Create a class Employee is having instance variables name and id. Create its subclass named Scientist which has an instance variables no_of_publication and experience. Now create its subclass, say DScientist which has instance variable award. Put a method like:
`public String toString(){ }`
in every class where you describe the class and from the main() method create an object of each class and print each object.

Code:

```
class Employee {
    int id;
    String name;
    Employee(int id, String name) {
        this.id = id;
        this.name = name;
    }
    public String toString() {
        return "Id = " + this.id + "\nName = " + this.name + "\n";
    }
}

class Scientist extends Employee {
    int no_of_publication;
    String experience;
    Scientist (int id, String name, int no_of_publication, String
experience) {
        super(id, name);
        this.no_of_publication = no_of_publication;
        this.experience = experience;
    }
    public String toString() {
        return super.toString() + "Publications = " +
this.no_of_publication + "\nExperience = " + this.experience + "\n";
    }
}

public class DScientist extends Scientist {
    String award;
    DScientist(int id, String name, int no_of_publication, String
experience, String award) {
```

```
        super(id, name, no_of_publication, experience);
        this.award = award;
    }
    public String toString() {
        return super.toString() + "Awards = " + this.award;
    }
    public static void main(String[] args) {
        Employee em = new Employee(712541, "Swagato Patra");
        Scientist st = new Scientist(712454, "Abhishek Pal", 2, "Junior
Scientist");
        DScientist ds = new DScientist(712136, "Swapnanil Dutta", 5,
"Senior Scientist", "Employee of the month");


System.out.println("----------------------Details-------------------------
-");
        System.out.println(em+"\n");
        System.out.println(st+"\n");
        System.out.println(ds+"\n");
    }
}
```

Output:

```
PS D:\OOPS-PCC-CS593\Day-11-(07.10.2020)> javac DScientist.java
PS D:\OOPS-PCC-CS593\Day-11-(07.10.2020)> java DScientist
----------------------Details-------------------------
Id = 712541
Name = Swagato Patra


Id = 712454
Name = Abhishek Pal
Publications = 2
Experience = Junior Scientist


Id = 712136
Name = Swapnanil Dutta
Publications = 5
Experience = Senior Scientist
Awards = Employee of the month
```

Question: Create a class with a method `void show()` and make three subclasses of it and all subclasses have this show() method overridden and call those methods using their corresponding object references.

Code:

```
class Grandparent {
    void show() {
        System.out.println("Class GrandParent");
    }
}

class Parent extends Grandparent {
    @Override
    void show() {
        System.out.println("Class Parent");
    }
}

class Child extends Parent {
    @Override
    void show() {
        System.out.println("Class Child");
    }
}

public class Qstn4{
    public static void main(String[] args) {
        Grandparent a = new Grandparent();
        a.show();
        Parent b = new Parent();
        b.show();
        Child c = new Child();
        c.show();
    }
}
```

Output:

```
PS D:\OOPS-PCC-CS593\Day-11-(07.10.2020)> javac Qstn4.java
PS D:\OOPS-PCC-CS593\Day-11-(07.10.2020)> java Qstn4
Class GrandParent
Class Parent
Class Child
```

Question: Do the problem 4 using dynamic method dispatching.

Code:

```
class Grandparent {
    void show() {
        System.out.println("Class GrandParent");
    }
}

class Parent extends Grandparent {
    @Override
    void show() {
        System.out.println("Class Parent");
    }
}

class Child extends Parent {
    @Override
    void show() {
        System.out.println("Class Child");
    }
}

public class DynamicMethodDispatching {
    public static void main(String[] args) {
        Grandparent obj;
        obj = new Grandparent();
        obj.show();
        obj = new Parent();
        obj.show();
        obj = new Child();
        obj.show();
    }
}
```

Output:

```
PS D:\OOPS-PCC-CS593\Day-11-(07.10.2020)> javac DynamicMethodDispatching.java
PS D:\OOPS-PCC-CS593\Day-11-(07.10.2020)> java DynamicMethodDispatching
Class GrandParent
Class Parent
Class Child
```

Question: Check without having any abstract method/s whether a class can be abstract; if so, then use that concrete method(s) from another class having the main method.

Code:

```
abstract class Base {
    public static void show() {
        System.out.println("Base abstract class");
    }
}

public class ConcreteAbstract extends Base {
    public static void main(String[] args) {
        show();
    }
}
```

Output:

```
PS D:\OOPS-PCC-CS593\Day-11-(07.10.2020)> javac ConcreteAbstract.java
PS D:\OOPS-PCC-CS593\Day-11-(07.10.2020)> java ConcreteAbstract
Base abstract class
```

Question: Create an abstract class with three abstract methods check whether you can we override only a few methods (not all methods) in subclass or not.

Code:

```
abstract class Base {
    abstract String method();
}

public class Abstract extends Base {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

Output:

```
PS D:\OOPS-PCC-CS593\Day-11-(07.10.2020)> javac Abstract.java
Abstract.java:5: error: Abstract is not abstract and does not override abstract method method() in Base
public class Abstract extends Base {
       ^
1 error
```