1. **Java Inheritance**

Inheritance in java is a mechanism in which one object acquires all the properties and behaviors of parent object. It is an important part of OPPS (Object Oriented programming system).

The idea behind inheritance in java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of parent class, and you can add new methods and fields also.

Why use inheritance in java:

* For Method Overriding (so runtime polymorphism can be achieved).
* For Code Reusability.

**Terms used in Inheritance**

Class: A class is a group of objects which have common properties. It is a template or blueprint from which objects are created.

Sub Class/Child Class: Subclass is a class which inherits the other class. It is also called a derived class, extended class, or child class.

Super Class/Parent Class: Superclass is the class from where a subclass inherits the features. It is also called a base class or a parent class.

Reusability: As the name specifies, reusability is a mechanism which facilitates you to reuse the fields and methods of the existing class when you create a new class. You can use the same fields and methods already defined in previous class.

Syntax of Java Inheritance

**class** Subclass-name **extends** Superclass-name{

//methods and fields

}

The ‘extends’ keyword indicates that you are making a new class that derives from an existing class. The meaning of "extends" is to increase the functionality.

In the terminology of Java, a class which is inherited is called parent or super class and the new class is called child or subclass.

**File: Employee.java**

**class** Employee {

**float** salary = 40000;

}

**class** Programmer **extends** Employee {

**int** bonus = 10000;

**public** **static** **void** main(String args[]) {

Programmer p = **new** Programmer();

System.***out***.println("Programmer salary is:" + p.salary);

System.***out***.println("Bonus of Programmer is:" + p.bonus);

}

}

OutPut:

Programmer salary is:40000.0

Bonus of programmer is:10000

**Types of inheritance in java:**



Multiple and Hydrid inheritance is not supported in java through class.



**Single Inheritance Example**

**File: TestInheritance.java**

**class** Animal {

**void** eat() {

System.***out***.println("eating...");

}

}

**class** Dog **extends** Animal {

**void** bark() {

System.***out***.println("barking...");

}

}

**class** TestInheritance {

**public** **static** **void** main(String args[]) {

Dog d = **new** Dog();

d.bark();

d.eat();

}

}

Output:

barking...

eating...

**Multilevel Inheritance Example**

**File: TestInheritance2.java**

**class** Animal {

**void** eat() {

System.***out***.println("eating...");

}

}

**class** Dog **extends** Animal {

**void** bark() {

System.***out***.println("barking...");

}

}

**class** BabyDog **extends** Dog {

**void** weep() {

System.***out***.println("weeping...");

}

}

**class** TestInheritance2 {

**public** **static** **void** main(String args[]) {

BabyDog d = **new** BabyDog();

d.weep();

d.bark();

d.eat();

}

}

**Output:**

weeping...

barking...

eating...

**Hierarchical Inheritance Example**

**File: TestInheritance3.java**

**class** Animal {

**void** eat() {

System.***out***.println("eating...");

}

}

**class** Dog **extends** Animal {

**void** bark() {

System.***out***.println("barking...");

}

}

**class** Cat **extends** Animal {

**void** meow() {

System.***out***.println("meowing...");

}

}

**class** TestInheritance3 {

**public** **static** **void** main(String args[]) {

Cat c = **new** Cat();

c.meow();

c.eat();

// c.bark();//C.T.Error

}

}

Output:

meowing...

eating...

**Why multiple inheritances are not supported in java?**

To reduce the complexity and simplify the language, multiple inheritances are not supported in java.

Consider a scenario where A, B and C are three classes. The C class inherits A and B classes. If classes A and B have same method and you call it from child class object, there will be ambiguity to call method of A or B class.

Since compile time errors are better than runtime errors, java renders compile time error if you inherit 2 classes. So whether you have same method or different, there will be compile time error now.

**Homework:**

Problem1:

Create two classes:

BaseClass

The Rectangle class should have two data fields-width and height of int types. The class should have display()method, to print the width and height of the rectangle separated by space.

DerivedClass

The RectangleArea class is derived from Rectangle class, i.e., it is the sub-class of Rectangle class. The class should have read\_input() method, to read the values of width and height of the rectangle. The RectangleArea class should also overload the display() method to print the area (width\*height) of the rectangle.

Input Format

The first and only line of input contains two space separated integers denoting the width and height of the rectangle.

Constraints

1 <= width,height <= 10^3

Output Format

The output should consist of exactly two lines:

In the first line, print the width and height of the rectangle separated by space.

In the second line, print the area of the rectangle.

Sample TestCase 1

Input

12 9

Output

12 9

108

Reference:

<https://beginnersbook.com/2013/04/oops-concepts/>

Class Examples:

**import** pack1.Animal;

**import** pack1.Mammals;

**public** **class** Simpleinheritance {

**public** **static** **void** main(String[] args) {

Animal a1 = **new** Animal(3, 5);

a1.show();

Mammals m1 = **new** Mammals(2,4,"Red");

m1.show();

}

}

**package** pack1;

**public** **class** Animal {

**public** **int** no\_of\_eyes;

**public** **int** no\_of\_legs;

Animal() { // Constructor Overloading

no\_of\_eyes = -1;

no\_of\_legs = -1;

}

**public** Animal(**int** no\_of\_eyes, **int** no\_of\_legs) {

**this**.no\_of\_eyes = no\_of\_eyes;

**this**.no\_of\_legs = no\_of\_legs;

}

**public** **void** show() {

System.***out***.println("I am in class Animal");

System.***out***.println("No of eyes = " + no\_of\_eyes);

System.***out***.println("No of Legs = " + no\_of\_legs);

}

}

**package** pack1;

**public** **class** Mammals **extends** Animal{

**public** String colour;

**public** Mammals(**int** no\_of\_eyes, **int** no\_of\_legs, String colour) {

//this.no\_of\_eyes = no\_of\_eyes;

//this.no\_of\_legs = no\_of\_legs;

**super**(no\_of\_eyes,no\_of\_legs);

**this**.colour = colour;

}

**public** **void** show() { //Method overriding -- inheritance -- for methods having same name and same parameters

// System.out.println("No of eyes = " + no\_of\_eyes);

// System.out.println("No of Legs = " + no\_of\_legs);

System.***out***.println("I am in class Mammal");

**super**.show();

System.***out***.println("Animal colour = " + colour);

}

}

**Homework:**

**Write a Java program showing usage of Super to call the superclass method.**