1. Create a class Publication with data members title(String) and price(int). From this class derive two classes Book and CD. Class Book adds pages(int) and CD adds Size(int). Each of these classes should have constructors and display(). Write a java program to implement this using super, this and method overriding concepts.

**package** com.rsc.babystepprogram;

**import** java.util.\*;

**class** Publication {

String title;

**int** price;

**public** Publication(String title, **int** price) {

**this**.title = title;

**this**.price = price;

}

**void** display() {

System.***out***.println("Publication title: "+**this**.title);

System.***out***.println("Publication price: "+**this**.price);

System.***out***.println("Display method in publication class");

}

}

**class** Book **extends** Publication {

**int** pages;

**public** Book(String title, **int** price, **int** pages) {

**super**(title, price);

**this**.pages = pages;

}

**void** display() {

System.***out***.println("Book title: "+**this**.title);

System.***out***.println("Book price: "+**this**.price);

System.***out***.println("No of pages: "+**this**.pages);

System.***out***.println("Display method in book class");

}

}

**class** CD **extends** Publication {

**int** size;

**public** CD(String title, **int** price, **int** size) {

**super**(title, price);

**this**.size = size;

}

**void** display() {

System.***out***.println("CD title: "+**this**.title);

System.***out***.println("CD price: "+**this**.price);

System.***out***.println("Size of CD: "+**this**.size);

System.***out***.println("Display method in CD class");

}

}

**public** **class** InheritanceExample {

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

Scanner scan = **new** Scanner(System.***in***);

Publication pub1 = **new** Publication("Deepthi", 1200);

pub1.display();

Publication pub2 = **new** Book("Mathemaics", 234, 980);

pub2.display();

Publication pub3 = **new** CD("Physics"+ "", 12, 34);

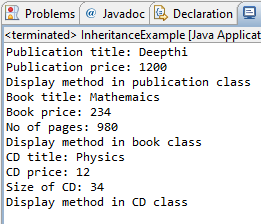
pub3.display();

scan.close();

}

}

Output:



1. A simple java program to demonstrate method overriding.

**package** com.rsc.babystepprogram;

**import** java.util.\*;

**class** Vehicle {

**public** **void** run() {

System.***out***.println("Vehicle is running...");

}

}

**class** Car **extends** Vehicle {

**public** **void** run() {

System.***out***.println("Car is running...");

}

}

**public** **class** MethodOverridingExample {

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

Scanner scan = **new** Scanner(System.***in***);

Vehicle v1 = **new** Vehicle();

Vehicle v2 = **new** Car();

v1.run();

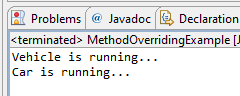
v2.run();

scan.close();

}

}

Output:



1. A java program to create an interface called Shape with CalculateArea(). Create three classes namely Square,Circle,Triangle which implements Shape.

**package** com.rsc.babystepprogram;

**import** java.util.\*;

**interface** Shape {

**public** **void** calculateArea();

}

**class** Square **implements** Shape {

**int** side;

Square(**int** side) {

**this**.side = side;

}

**public** **void** calculateArea() {

System.***out***.println("Area of square: "+ **this**.side\***this**.side);

}

}

**class** Circle **implements** Shape {

**float** radius;

Circle(**float** radius) {

**this**.radius = radius;

}

**public** **void** calculateArea() {

System.***out***.println("Radius of the circle: "+ Math.***PI***\***this**.radius\***this**.radius);

}

}

**class** Triangle **implements** Shape{

**int** base, height;

Triangle(**int** base, **int** height) {

**this**.base = base;

**this**.height = height;

}

**public** **void** calculateArea() {

System.***out***.println("Area of the traingle: "+ 0.5\***this**.base\***this**.height);

}

}

**public** **class** InterfaceExample {

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

Scanner scan = **new** Scanner(System.***in***);

System.***out***.println("Enter the side of the square: ");

**int** side = scan.nextInt();

Square s1 = **new** Square(side);

s1.calculateArea();

System.***out***.println("Enter the radius of the circle: ");

**float** radius = scan.nextFloat();

scan.nextLine();

Circle c1 = **new** Circle(radius);

c1.calculateArea();

System.***out***.println("Enter the base and height of traingle: ");

String s[] = scan.nextLine().split(" ");

**int** base = Integer.*parseInt*(s[0]);

**int** height = Integer.*parseInt*(s[1]);

Triangle t1 = **new** Triangle(base, height);

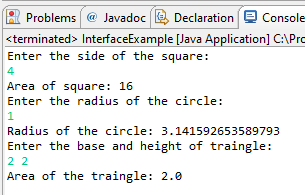
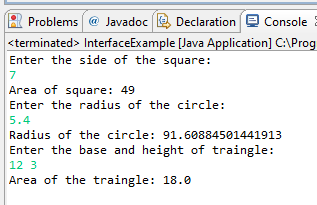
t1.calculateArea();

scan.close();

}

}

Output:

1. Create two packages p1 and p2. The package p1 contains class A which contains one display(). Create class B in package p2. The main method of class B invoke A’s display(). Write a java program to do this.

Package1:

**package** package1;

**public** **class** A {

**public** **void** display() {

System.***out***.println("This method is in package1 and class A");

}

}

Package2:

**package** package2;

**import** package1.A;

**public** **class** B {

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

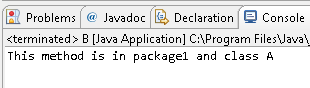
A objA = **new** A();

objA.display();

}

}

Output:



1. A java program to count numbers, characters in the command line arguments using Exception handling mechanism.

**package** com.rsc.babystepprogram;

**public** **class** CountNumberAndCharacters {

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

**int** numberCount = 0, characterCount = 0;

System.***out***.print("Command line arguments: ");

**for**(String i: args)

System.***out***.print(i + " ");

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

**for**(**int** i=0; i<args.length; i++) {

**try** {

**int** number = Integer.*parseInt*(args[i]);

numberCount += 1;

} **catch**(Exception e) {

characterCount += 1;

}

}

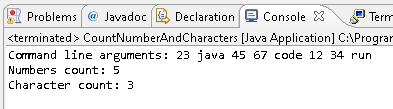
System.***out***.println("Numbers count: " + numberCount);

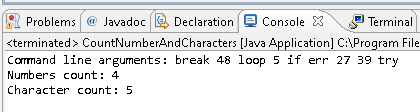
System.***out***.println("Character count: " + characterCount);

}

}

Output:





1. What is inheritance?
2. Inheritance is one of the Object Oriented Programming techniques in which one class acquires or inherits the attributes and behaviours of another methods partially or completely.

The class which is inherited is called parent class or base class or super class.

The class which inherits is called child class or derived class or sub class.

1. What is multiple inheritance?
2. Multiple inheritance is a type of inheritance in which a child class inherits from more than one parent class. Luckily, Java does not support multiple inheritance since it leads to ambiguity to the compiler.
3. What is the use of super keyword?
4. a. Super keyword is used to call the immediate parent class methods.

b. it is also used to call the base class constructor.

c. used to differentiate between super class and sub class methods with same name

1. what is abstract method?
2. A method which is declared as abstract is called abstract method. Abstract methods are declared but not implemented.
3. What is abstract class?
4. A class which is declared as abstract is called abstract class. it may or may not contains abstract methods. If a class contains at least one abstract method, then the class must be declared as abstract. These classes cannot be instantiated.
5. What is the use of final modifier?
6. a. Final is a modifier in java which when used with variable, makes it constant and its value cannot be changed.

b. methods which are declared as final cannot be overridden.

c. classes which are declared as final cannot be extended or inherited.

1. What is interface? Write the syntax of interface?
2. Interface is a mechanism through which we can achieve abstraction and multiple inheritance in java. Interface contains abstract methods with no methods body. There can be only abstract methods in interface.
3. What is package?
4. A package is a collection of related classes and interfaces in java. All the classes that are related to a particular project can be collected in a package so that it can be imported in another project easily. And also different packages can also contain classes with same name.
5. What is exception?
6. An exception is an abnormal condition which obstructs the normal flow of execution of the program. The program will crash if those exceptions are not handled properly. Java is very much helpful in dealing with those exceptions easily with the help of exception handling
7. What is the use of finally block?
8. Finally block is followed by the try catch block in exception handling. Whether or not the exception is handled, the code in finally block is executed at last.