**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.**

**A.**

class Publication

{

String title;

int price;

public Publication(String title,int price)

{

this.title=title;

this.price=price;

}

void display()

{

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

System.out.println("Title:" + title + " Price:" + price);

}

}

class Book extends Publication

{

int pages;

Book(String title,int price,int pages)

{

super(title,price);

this.pages=pages;

}

void display()

{

super.display();

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

System.out.println("Title:" +title+ " Price: " + price + " Pages:" + pages);

}

}

class CD extends Publication

{

int size,pages;

CD(String title,int price,int pages,int size)

{

super(title,price);

this.size=size;

this.pages=pages;

}

void display()

{

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

System.out.println("Title:" +title+ " Price:" + price + " Pages:" + pages + " Size:" +size);

}

}

public class InheritResult

{

public static void main(String args[])

{

String title="Java";

int price=600;

int pages=500;

int size=40;

Book book = new Book(title,price,pages);

book.display();

CD cd=new CD(title,price,pages,size);

cd.display();

}

}

**Output :**

Display method in publication:

Title:Java Price:600

Display method in book:

Title:Java Price: 600 Pages:500

Display method in CD:

Title:Java Price:600 Pages:500 Size:40

**2.Write a simple java program to demonstrate method overriding.**

**A.** If subclass (child class) has the same method as declared in the parent class, it is known as **method overriding in Java**.

**Program :**

class Human{

public void eat()

{

System.out.println("Human is eating");

}

}

class Boy extends Human{

public void eat(){

System.out.println("Boy is eating");

}

public static void main( String args[]) {

Boy obj = new Boy();

obj.eat();

}

}

**Output :**

Boy is eating

**3.Write a java program to create an interface called Shape with CalculateArea(). Create three classes namely Square,Circle,Triangle which implements Shape.**

**A.** import java.lang.Math;

interface Shape

{

void CalculateArea();

}

class Square implements Shape

{

public void CalculateArea()

{

int side=8;

int area=side\*side;

System.out.println("The area of square is: " + area);

}

}

class Circle implements Shape

{

int radius=5;

public void CalculateArea()

{

double area=Math.PI\*radius\*radius;

System.out.println("The area of circle is: " + area);

}

}

class Triangle implements Shape

{

public void CalculateArea()

{

int base=6,height=7;

double area=0.5\*base\*height;

System.out.println("The area of triangle is: " + area);

}

}

public class InterfaceExample

{

public static void main(String args[])

{

Shape s1=new Square();

Shape s2=new Circle();

Shape s3=new Triangle();

s1.CalculateArea();

s2.CalculateArea();

s3.CalculateArea();

}

}

**Output :**

The area of square is: 64

The area of circle is: 78.53981633974483

The area of triangle is: 21.0

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.**

**A.**

package p1;

public class A

{

public void display()

{

System.out.println("I am in class A in package p1");

}

}

package p2;

import p1.A;

class B

{

public static void main(String args[])

{

A obj=new A();

obj.display();

}

}

**Output :**

./Playground/A.java:9: error: class, interface, or enum expected

package p2;

^

./Playground/A.java:10: error: class, interface, or enum expected

import p1.A;

^

2 errors

**5.Write a java program to count numbers, characters in the command line arguments using Exception handling mechanism.**

**A.**

import java.util.\*;

class Count

{

public static void main(String a[])

{

String s=a[0];

try

{

int digit=0;

int chr=0;

for(int i=0;i<s.length();i++)

{

if(Character.isLetter(s.charAt(i)))

chr++;

else

digit++;

}

System.out.println("No.of characters are: " +chr);

System.out.println("No.of digits are: " +digit);

}

catch(Exception e)

{

System.out.println(e);

}

}

}

**Output :**

No.of characters are: 0

No.of digits are: 1

**1. What is Inheritance?**

1. Inheritance can be defined as the process where one class acquires all the properties and behaviors of another class.

The class which inherits the properties of other is known as subclass or base class/ and the class whose properties are inherited is known as superclass (base class, parent class).

**Syntax:**

class Super {

.....

.....

}

class Sub **extends** Super {

.....

.....

}

**2. What is Multiple Inheritance?**

**A.** Multiple Inheritance is a feature of object oriented concept, where a class can inherit properties of more than one parent class.

To reduce the complexity and simplify the language, multiple inheritance is not supported in java.

**3. What is the use of Super keyword?**

**A.** The **super** keyword in Java is a reference variable which is used to refer immediate parent class object.Whenever you create the instance of subclass, an instance of parent class is created implicitly which is referred by super reference variable.

**Uses :**

1. super can be used to refer immediate parent class instance variable.
2. super can be used to invoke immediate parent class method.
3. super() can be used to invoke immediate parent class constructor.

**4. What is abstract method?**

**A.** If we want a class to contain a particular method but you want the actual implementation of that method to be determined by child classes, you can declare the method in the parent class as an **abstract**.

So, the method which is declared as abstract does not have implementation is known as an abstract method.

**5. What is abstract class?**

**A.** A class which is declared as abstract is known as an **abstract class**. It can have abstract and non-abstract methods. It needs to be extended and its method implemented. It cannot be instantiated.

**Example :**

abstract class Bike{ //abstract class abstract void run(); //abstract method }

class Honda4 extends Bike{ void run(){ //abstract method implementation System.out.println("running safely.."); }

public static void main(String args[]){

Bike obj = new Honda4();

obj.run();

} }

**6. What is the use of final modifier?**

**A.** **final** is a [non-access modifier](https://www.geeksforgeeks.org/access-and-non-access-modifiers-in-java/) applicable **only to a variable, a method or a class**. Following are different contexts where final is used.

1) **Final variable** is used to create constant variables.

2) **Final method** is used to prevent method overriding.

3) **Final class** is used to prevent inheritance.

**7. What is interface? Write the syntax of interface.**

**A.** An **interface in Java** is a blueprint of a class. It has static constants and abstract methods. The interface in Java is a mechanism to achieve [abstraction](https://www.javatpoint.com/abstract-class-in-java). There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction and multiple [inheritance in Java](https://www.javatpoint.com/inheritance-in-java).

### **Syntax:**

interface <interface\_name>{

//declare constant fields

//declare methods that abstract

//by default.

}

**8. What is package?**

**A.** A **package** is a group of similar types of classes, interfaces and sub-packages.

Package in java can be categorized in two form, built-in package and user-defined package.

There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc.

**9. What is exception?**

**A.** An exception (or exceptional event) is a problem that arises during the execution of a program. When an **Exception** occurs the normal flow of the program is disrupted and the program/Application terminates abnormally, which is not recommended, therefore, these exceptions are to be handled.

**10. What is the use of finally block?**

**A.** Finally block in java can be used to put "cleanup" code such as closing a file, closing connection etc. It is always executed whether exception is handled or not. It follows try or catch block.