Assignment-3

**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.**

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

**class** Publications

{

String title;

**int** price;

Publications(String title,**int** price)

{

**this**.title=title;

**this**.price=price;

}

**void** display()

{

System.***out***.println("The title is "+title+" price"+price);

}

}

**class** Book **extends** Publications

{

**int** page;

Book(String title,intprice,**int** page)

{

**super**(title,price);

**this**.page=page;

}

**void** display()

{

System.***out***.println("The book title is "+title+" price "+price+" pages "+page);

}

}

**class** CD **extends** Publications

{

**int** size;

CD(String title,intprice,**int** size)

{

**super**(title,price);

**this**.size=size;

}

**void** display()

{

System.***out***.println("The book title is "+title+" price "+price+" size "+size);

}

}

**class** Demo

{

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

{

Publications p=**new** Publications("JAVA",2000);

p.display();

Book b=**new** Book("Introduction to java",2000,200);

b.display();

CD c=**new** CD("Java CD",4000,30);

c.display();

}

}

**Output:**

The title is JAVA price2000

The book title is Introduction to java price 2000 pages 200

The book title is Java CD price 4000 size 30

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

**class** Parent {

**void** show()

{

System.***out***.println("Parent method");

}

}

**class** Child **extends** Parent {

**void** show()

{

System.***out***.println("Child method");

}

}

**class** Inherit {

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

Parent obj1 = **new** Parent();

obj1.show();

Parent obj2 = **new** Child();

obj2.show();

}

}

Output:

Parent method

Child method

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

**interface** Shape

{

**void** calculateArea();

}

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

{

**int** r = 5;

**double** pi = 3.14, area = 0;

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

{

area = pi \* r \* r;

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

}

}

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

{

**int** a=6;

**double** area;

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

{

area = a\*a;

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

}

}

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

{

**int** h = 6, b = 4;

**double** area;

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

{

area = 0.5\*b\*h;

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

}

}

**public** **class** Demo

{

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

{

Shape s;

s=**new** Circle();

s.calculateArea();

s=**new** Square();

s.calculateArea();

s=**new** Triangle();

s.calculateArea();

}

}

Output:

Area of circle is:78.5

Area of square is:36.0

Area of triangle is:12.0

**4.Create two packages p1 and p2.The package p1 contains class A which contains one display().Create class B in package 2.The main method of class B invoke A’s display .Write a java program to do this?**

**package p1**;

**public** **class** A

{

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

{

System.***out***.println("Package A called");

}

}

**import** p1.\*;

**class** B{

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

{

A a=**new** A();

a.display();

}

}

Output:

C:\programs\packages>javac B.java

C:\programs\packages>java B

Package A called

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

**class** Demo

{

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

{

**try**

{

String a=args[0];

**int** c=0,n=0;

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

{

**if**(!(Character.*isSpaceChar*(a.charAt(i))))

{

**if**(Character.*isDigit*(a.charAt(i)))

n+=1;

**else** **if**(Character.*isLetter*(a.charAt(i)))

c+=1;

}

}

System.***out***.println("The character count is "+cc+" digit count "+cn);

}

**catch**(Exception e)

{

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

}

}

}

Output:

C:\programs>java Demo

java.lang.ArrayIndexOutOfBoundsException: 0

C:\programs>java Demo srilu999

The character count is 5 digit count 3

**1.What is Inheritance?**

**Inheritance** in Java is a mechanism in which one object acquires all the properties and behaviors of a parent object. It is an important part of OOPs (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 the parent class. Moreover, you can add new methods and fields in your current class also.

Inheritance represents the IS-A relationship which is also known as a *parent-child* relationship.

**2 .What is Multiple Inheritance?**

“Multiple Inheritance” refers to the concept of one class extending (Or inherits) more than one base class. The problem with “multiple inheritance” is that the derived class will have to manage the dependency on two base classes.In java ,mulple inheritance can be achieved using interfaces.

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

**Usage of Java super Keyword:**

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 method without body (no implementation) is known as abstract method. A method must always be declared in an abstract class, or in other words you can say that if a class has an abstract method, it should be declared abstract as well.

public abstract int myMethod(int n1, int n2);

**5.What is abstract class?**

A class that is declared using “**abstract**” keyword is known as abstract class. It can have abstract methods(methods without body) as well as concrete methods (regular methods with body). A normal class(non-abstract class) cannot have abstract methods.

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

* When a final modifier is used with a class then the class cannot be extended further.
* This is one way to protect your class from being subclassed and often sensitive classes are made final due to security reason.
* When the final keyword is used with a method that it cannot be overridden in Java, which means you cannot override the logic of the method in the subclass.
* When the final keyword is used with a variable then its value cannot be changed once assigned.

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

 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*. There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction and multiple inheritance in Java

**Syntax:**

**interface** <interface\_name>{

    // declare constant fields

    // declare methods that abstract

    // by default.

}

**8. What is package?**

PACKAGE in Java is a collection of classes, sub-packages, and interfaces. It helps organize your classes into a folder structure and make it easy to locate and use them. More importantly, it helps improve code reusability.

Each package in Java has its unique name and organizes its classes and interfaces into a separate namespace, or name group.

Although interfaces and classes with the same name cannot appear in the same package, they can appear in different packages. This is possible by assigning a separate namespace to each Java package.

**9. What is exception?**

 An *exception* is an event, which occurs during the execution of a program, that disrupts the normal flow of the program's instructions.

When an error occurs within a method, the method creates an object and hands it off to the runtime system. The object, called an *exception object*, contains information about the error, including its type and the state of the program when the error occurred. Creating an exception object and handing it to the runtime system is called *throwing an exception*.

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

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

.