



OBJECT ORIENTED PROGRAMMING WITH JAVA

Lab manual

ALI HAIDER
syedalihaider.ciit@gmail.com

Course Title: Object Oriented Programming**Course Code:****Course Outcomes:**

At the end of the course the student should be able to:

1. Apply Object Oriented Programming concepts to solve a given problem.
2. Apply design patterns to design a solution for a given problem.
3. Apply inheritance, polymorphism and exception handling mechanism to implement reusable, robust java programs.
4. Implement user interface java programs for a given scenario.

List of Practical

Sr.#	LAB	Topics
1	One	Introduction to IDE-NetBeans, Getting Started with Java
2	Two	Java Basics (Input/output, variable declaration and initialization, strings and arrays)
3	Three	Java Basics (Selection Structure and Iterative Structure, functions)
4	Four	OOP (creating classes, objects, constructors)
5	Five	access modifiers, inheritance
6	Six	multiple/multilevel inheritance
7	Seven	Function overriding
8	Eight	Polymorphism
9	Nine	abstract classes and interfaces
10	Ten	exception handling and Java file handling

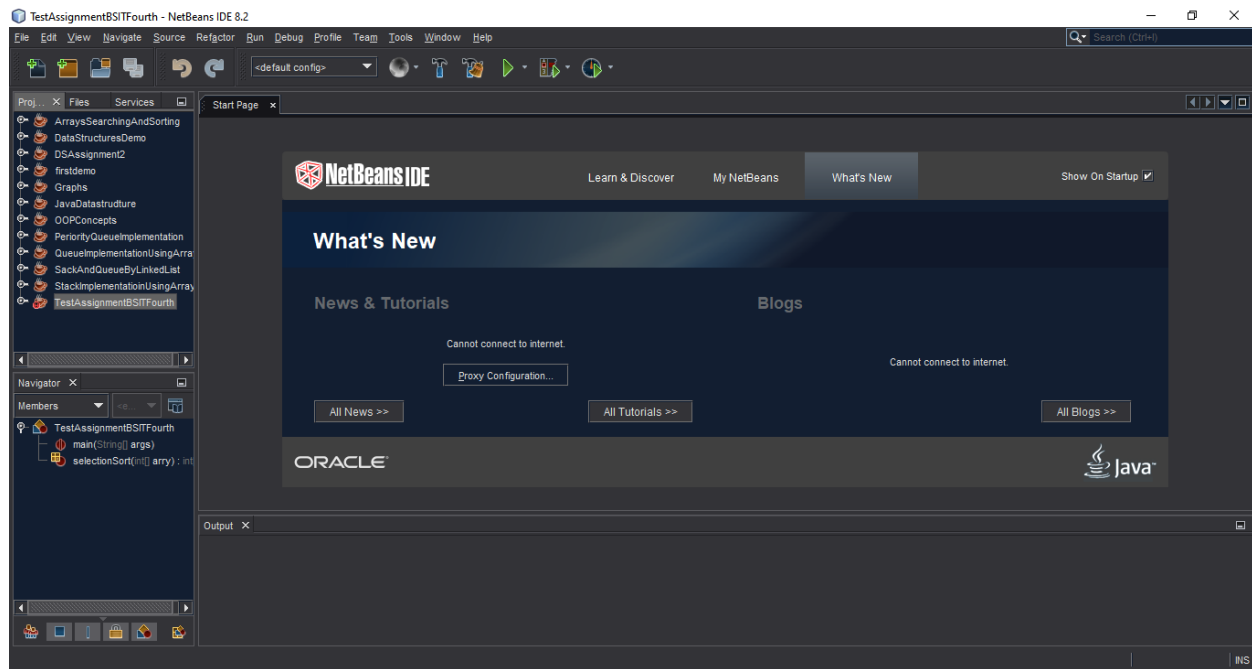
Introduction to NetBeans and Java

LAB-1

Download NetBeans

You can download NetBeans from here <https://netbeans.org/>

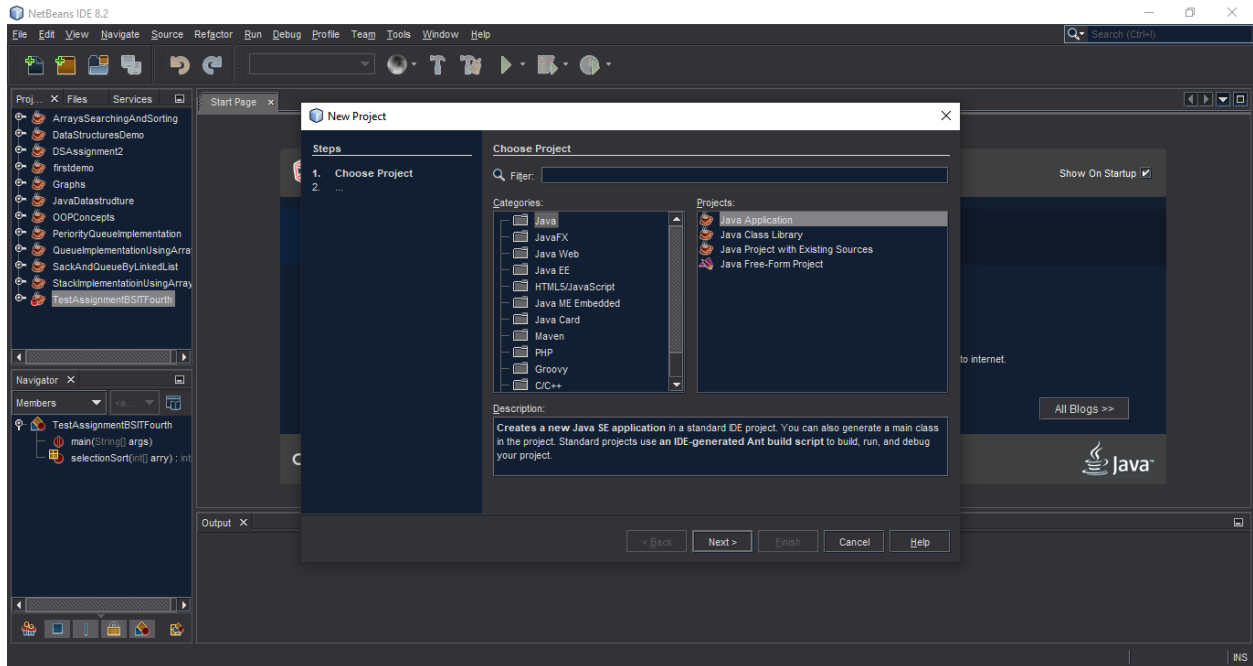
NetBeans User interface



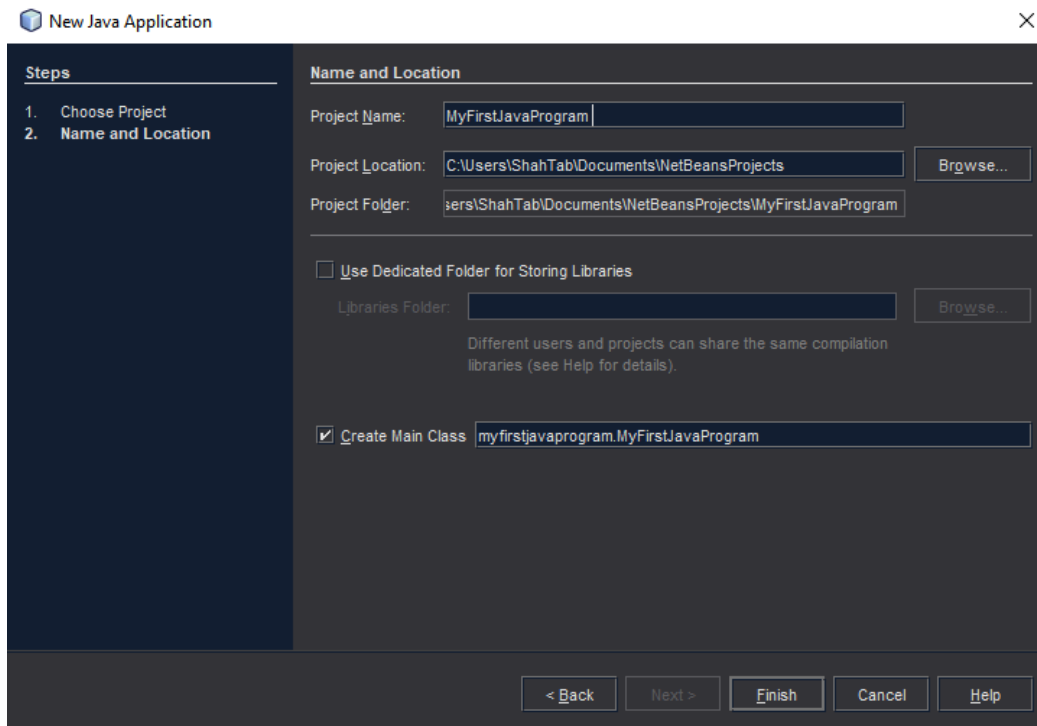
Creation and Compilation of new Project

For creating a new project press **Ctrl+Shift+N** or click on new project

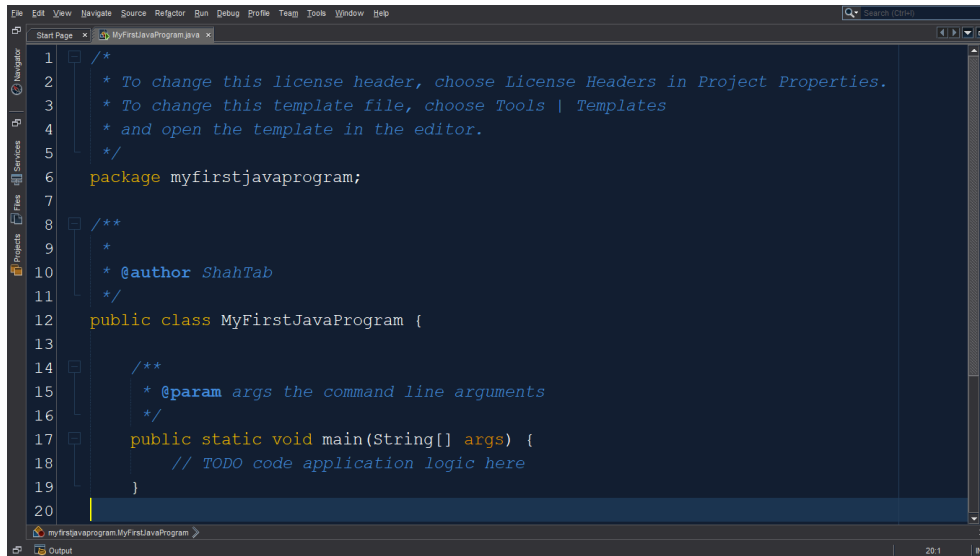




Select Project category Java and project type Java Application then click on next.

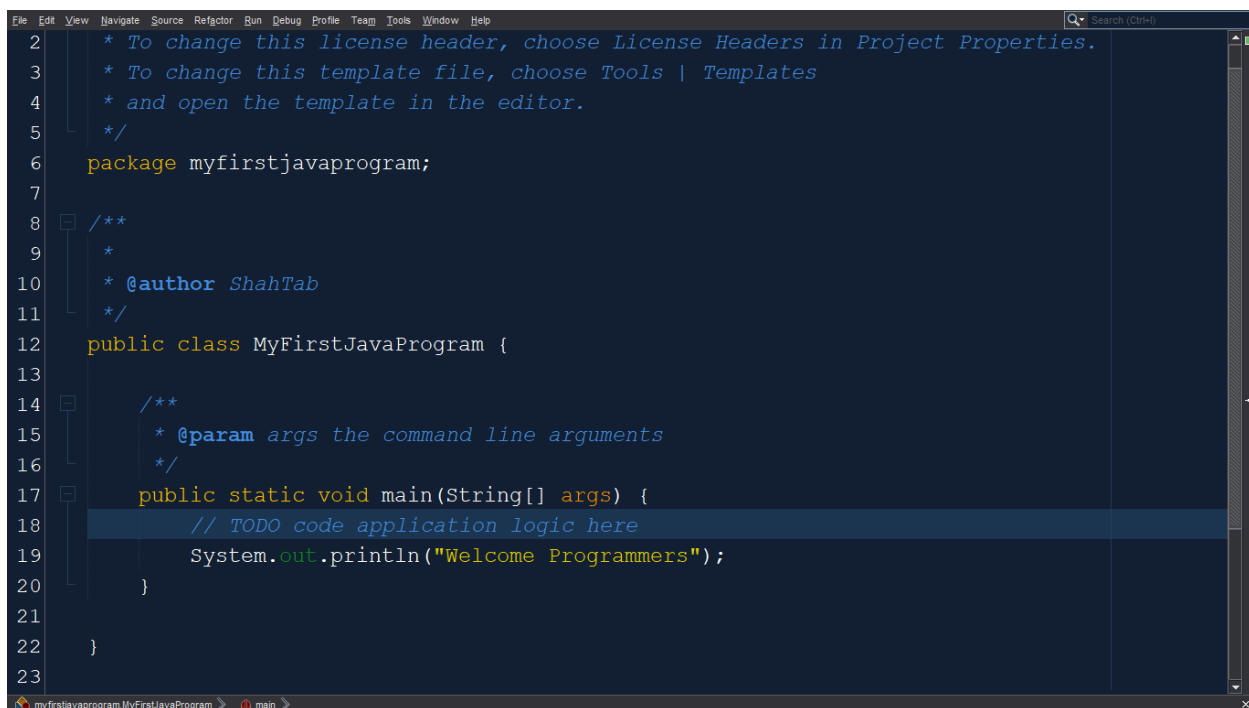


Type project Name “MyFirstJavaProgram” and click on finish.



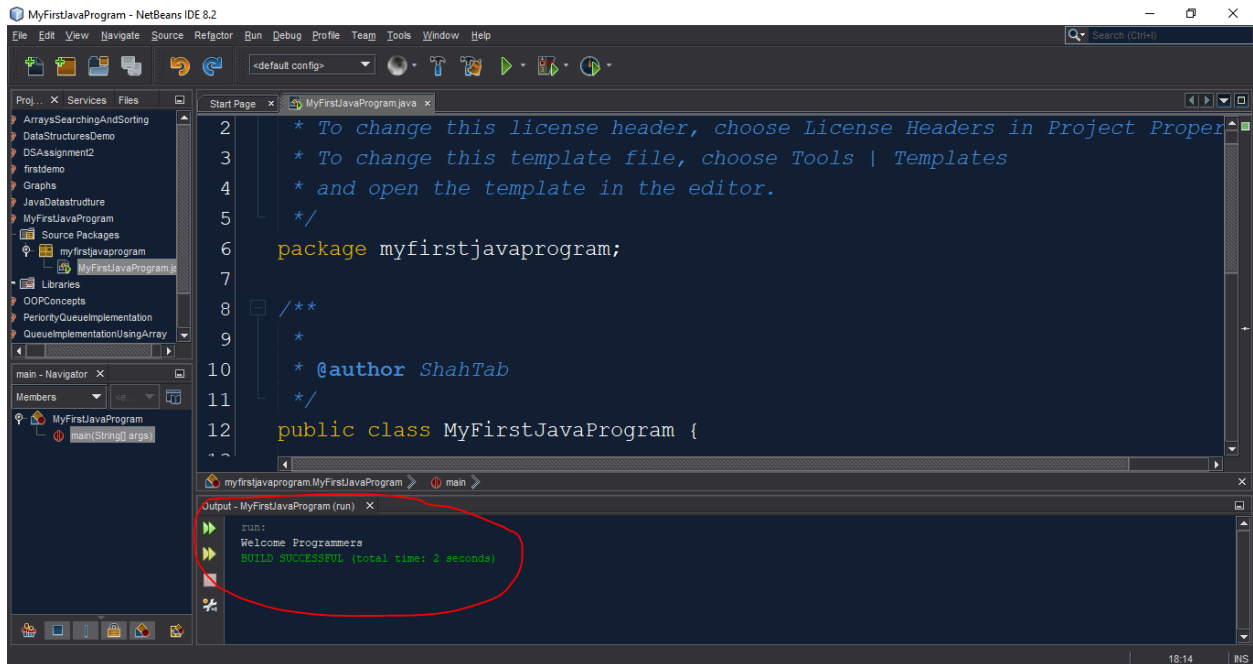
```
1  /*
2   * To change this license header, choose License Headers in Project Properties.
3   * To change this template file, choose Tools | Templates
4   * and open the template in the editor.
5   */
6  package myfirstjavaprogram;
7
8  /**
9   *
10   * @author ShahTab
11   */
12  public class MyFirstJavaProgram {
13
14      /**
15       * @param args the command line arguments
16       */
17      public static void main(String[] args) {
18          // TODO code application logic here
19      }
20  }
```

Now start writing Java code in main function so type `System.out.println("Welcome Programmers");`;



```
2  /*
3   * To change this license header, choose License Headers in Project Properties.
4   * To change this template file, choose Tools | Templates
5   * and open the template in the editor.
6   */
7  package myfirstjavaprogram;
8
9  /**
10   *
11   * @author ShahTab
12   */
13  public class MyFirstJavaProgram {
14
15      /**
16       * @param args the command line arguments
17       */
18      public static void main(String[] args) {
19          // TODO code application logic here
20          System.out.println("Welcome Programmers");
21      }
22  }
```

To run project press F6 and see result in output window.



Task-1

Download and Install NetBeans.

Task-2

Write a program using JAVA for displaying a message “Welcome to DCS&IT” on output window.

Hint: print using `System.out.println()` function.

Java Basics-1

LAB-2

First Java Program

Let us look at a simple code that will print the words “Welcome Programmers”.

Example

```
import java.io.*;

public class MyFirstJavaProgram {

    /* This is my first java program.

     * This will print ' Welcome Programmers ' as the output
     */

    public static void main(String []args) {

        System.out.println("Welcome Programmers"); // prints Welcome Programmers

    }

}
```

Description: In above example first we import java.io package that have different classes.

Java I/O

Java I/O (Input and Output) is used to process the input and produce the output. Java uses the concept of a stream to make I/O operation fast. The java.io package contains all the classes required for input and output operations.

For output you can write following statement:

```
System.out.println("Message Type Here");
```

For input you can use Scanner class like as follows:

```
Scanner sc=new Scanner(System.in);

String s=sc.nextLine();
```

Variable Declaration

Variable declaration is process of assigning datatype to variable.

```
datatype variableName;
```

For Example:

```
int a;  
  
float b;  
  
String s;
```

Variable Initialization

The process of assigning value to variable is known as variable initialization.

e.g

```
int a;  
  
float b;  
  
String s;  
  
//Variable initialization  
  
a=9;  
  
b=3.4;  
  
s= "Welcome";
```

Arrays

Arrays are objects that store multiple variables of the same type. However, an array itself is an object on the heap.

For array declaration and initialize

```
Datatype[ ] arrayName={value1,value2,value3,.....};  
  
Int[] marks={60,80,7,46,66};
```

Strings

Strings are used for storing text. A String variable contains a collection of characters surrounded by double quotes. Create a variable of type String and assign it a value:

```
String greeting = "Hello";
```

A String in Java is actually an object, which contain methods that can perform certain operations on strings. For example, the length of a string can be found with the `length()` method:

```
String txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";  
System.out.println("The length of the txt string is: " + txt.length());
```

Task-1

Write a program using JAVA for a function that take five numbers from the user and store in array of size 5 after that display the sum and average of numbers.

Hint: Use scanner class for taking input from user. You can use `sc.nextInt()` for taking integer from user and save that integer in array.

Task-2

Write a program using JAVA that take input Name, Age and Address from user and print it.

Hint: Use scanner class for taking input from user. You can use `sc.nextLine()` for taking string from user. Use String datatype for Name, Address and int for age.

Java Basics-2

LAB-3

Java Conditions and If Statements

Java supports the usual logical conditions from mathematics:

Less than: $a < b$

Less than or equal to: $a \leq b$

Greater than: $a > b$

Greater than or equal to: $a \geq b$

Equal to $a == b$

Not Equal to: $a != b$

You can use these conditions to perform different actions for different decisions. Java has the following conditional statements:

Use if to specify a block of code to be executed, if a specified condition is true

Use else to specify a block of code to be executed, if the same condition is false

Use else if to specify a new condition to test, if the first condition is false

Use switch to specify many alternative blocks of code to be executed

The if Statement

Use the if statement to specify a block of Java code to be executed if a condition is true.

Syntax

```
if (condition) {  
    // block of code to be executed if the condition is true  
}
```

We test two values to find out if 20 is greater than 18. If the condition is `true`, print some text:

```
if (20 > 18) {  
    System.out.println("20 is greater than 18");  
}
```

The if else Statement

```
int time = 20;  
if (time < 18) {  
    System.out.println("Good day.");  
} else {  
    System.out.println("Good evening.");  
}
```

Java For Loop

When you know exactly how many times you want to loop through a block of code, use the `for` loop.

Syntax

```
for (statement 1; statement 2; statement 3) {  
    // code block to be executed  
}
```

Statement 1 is executed (one time) before the execution of the code block.

Statement 2 defines the condition for executing the code block.

Statement 3 is executed (every time) after the code block has been executed.

Example

```
for (int i = 0; i < 5; i++) {  
    System.out.println(i);  
}
```

Example

```
for (int i = 0; i <= 10; i = i + 2) {  
    System.out.println(i);  
}
```

For-Each Loop

There is also a "for-each" loop, which is used exclusively to loop through elements in an **array**:

Syntax

```
for (type variableName : arrayName) {  
    // code block to be executed  
}
```

The following example outputs all elements in the **cars** array, using a "for-each" loop:

Example

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};  
for (String i : cars) {  
    System.out.println(i);  
}
```

Task-1

Write a program using JAVA for a function that take five numbers from the user and store in array of size 5 after that display the sum and average of numbers.

Hint: Use scanner class and loop for taking input from user. You can use `sc.nextInt()` for taking integer from user and save that integer in array.

Task-2

Write a program that prompts the user to input a positive integer. It should then print the multiplication table of that number.

Assignment Question

Write a program that prompts the user to input an integer and then outputs the number with the digits reversed. For example, if the input is 12345, the output should be 54321.

OOP (creating classes, objects, constructors)

LAB-4

Java is an object-oriented programming language. Everything in Java is associated with classes and objects, along with its attributes and methods. For example: in real life, a car is an object. The car has attributes, such as weight and color, and methods, such as drive and brake. A Class is like an object constructor, or a "blueprint" for creating objects.

Create a Class

To create a class, use the keyword `class`:

MyClass.java

Create a class named "MyClass" with a variable x:

```
public class MyClass {  
    int x = 5;  
}
```

Create an Object

In Java, an object is created from a class. We have already created the class named `MyClass`, so now we can use this to create objects.

To create an object of `MyClass`, specify the class name, followed by the object name, and use the keyword `new`:

Example

Create an object called "myObj" and print the value of x:

```
public class MyClass {  
    int x = 5;  
  
    public static void main(String[] args) {
```

```
        MyClass myObj = new MyClass();  
        System.out.println(myObj.x);  
    }  
}
```

Create an object called "myObj" and print the value of x:

```
public class MyClass {  
    int x = 5;  
  
    public static void main(String[] args) {  
        MyClass myObj = new MyClass();  
        System.out.println(myObj.x);  
    }  
}
```

Using Multiple Classes

You can also create an object of a class and access it in another class. This is often used for better organization of classes (one class has all the attributes and methods, while the other class holds the main() method (code to be executed)). Remember that the name of the java file should match the class name. In this example, we have created two files in the same directory/folder:

- MyClass.java
- OtherClass.java

MyClass.java

```
public class MyClass {  
    int x = 5;  
}
```

OtherClass.java

```
class OtherClass {  
    public static void main(String[] args) {  
        MyClass myObj = new MyClass();  
        System.out.println(myObj.x);  
    }  
}
```

```
}
```

Constructor

A constructor in Java is a **special method** that is used to initialize objects. The constructor is called when an object of a class is created. It can be used to set initial values for object attributes:

Example

Create a constructor:

```
// Create a MyClass class
public class MyClass {
    int x; // Create a class attribute

    // Create a class constructor for the MyClass class
    public MyClass() {
        x = 5; // Set the initial value for the class attribute x
    }

    public static void main(String[] args) {
        MyClass myObj = new MyClass(); // Create an object of class MyClass (This
will call the constructor)
        System.out.println(myObj.x); // Print the value of x
    }
}
```

Constructor Parameters

Constructors can also take parameters, which is used to initialize attributes. The following example adds an int y parameter to the constructor. Inside the constructor we set x to y (x=y). When we call the constructor, we pass a parameter to the constructor (5), which will set the value of x to 5:

Example

```
public class MyClass {
    int x;
```

```
public MyClass(int y) {  
    x = y;  
}  
  
public static void main(String[] args) {  
    MyClass myObj = new MyClass(5);  
    System.out.println(myObj.x);  
}  
}
```

You can have as many parameters as you want:

Example

```
public class Car {  
    int modelYear;  
    String modelName;  
  
    public Car(int year, String name) {  
        modelYear = year;  
        modelName = name;  
    }  
  
    public static void main(String[] args) {  
        Car myCar = new Car(1969, "Mustang");  
        System.out.println(myCar.modelYear + " " + myCar.modelName);  
    }  
}
```

Task-1

Write a program that have a class (PRODUCT) that contain three data members and three member function first is constructor that set the default suitable values and second is in() that can be set and receive values from the user and multiply three values in one variable and third function that show the product of three values. Test this program in main using minimum five object of PRODUCT.

Task-2

Create a class called Invoice that a hardware store might use to represent an invoice for an item sold at the store. An Invoice should include four pieces of information as data members a part (type string), a part description (type string), a quantity of the item being purchased (type int) and a price per item (type int). Your class should have a constructor that initializes the four data members. Provide a set and a get function for each data member. In addition, provide a member function named (getInvoiceAmount) that calculates the invoice amount (i.e., multiplies the quantity by the price per item), then returns the amount as an int value. If the quantity is not positive, it should be set to 0. If the price per item is not positive, it should be set to 0. Write a test program that demonstrates class Invoice's capabilities.

Assignment Question

Design and implement a class DATE that implements the day of the week in a program. The class DATE should store the day, such as Sun for Sunday. The program should be able to perform the following operations on an object of type DATE:

- a. Set the day.
- b. Print the day.
- c. Return the day.
- d. Return the next day.
- e. Return the previous day.
- f. Calculate and return the day by adding certain days to the current day.

For example, if the current day is Monday and we add 4 days, the day to be returned is Friday. Similarly, if today is Tuesday and we add 13 days, the day to be returned is Monday.

- g. Add the appropriate constructors.

OOP (access modifiers, inheritance)

LAB-5

Access Modifiers

Specifiers determine whether a field or method in a class, can be used or invoked by another method in another class or sub-class. Access Specifiers can be used to restrict access.

Access Modifiers	Default	private	protected	public
Accessible inside the class	yes	yes	yes	yes
Accessible within the subclass inside the same package	yes	no	yes	yes
Accessible outside the package	no	no	no	yes
Accessible within the subclass outside the package	no	no	yes	yes

Example:

```
public class Person {    // public class

    private int age;    // private (encapsulated) instance variables

    public set(int ag) {    // setting values of private fields

        this.age = x;

    }

    public get() {    // setting values of private fields

        return Point(age);

    }

}
```

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.

```
class Subclass-name extends Superclass-name  
{  
    //methods and fields  
}
```

e.g.

Create a class SHAP that have two data members height and width and two member functions input () and output () input take height and width from the user and output display the height and width. And make another class RECTANGLE that inherit from the SHAP that use the data members and function of the SHAP class. In main make two RECTANGLE objects and use using input and output function.

ANS:

```
private class shape{  
  
    Protected float height,width;  
  
    public shap(){height=0.0;width=0.0;}  
  
    public get(){  
  
        cout<<"Enter Height\n";  
  
        cin>>height;
```

```
cout<<"Enter Width\n";

cin>>width;}

public void show(){cout<<"Height : "<<height<<"\nWidth : "<<width;

}

}

public class rectangle extends shape{

};

int main(){

rectangle r;

r.get();

r.show();

getch();

}
```

Task-1

Using above example take another class COST. RECTANGLE that inherit publically also COST. In COST one member function that calculate the cost according to the area. In main create two object of RECTANGLE class in one object initialize height and width and in second take from the user and calculate the area. And area is a member function of RECTANGLE class after calculating the area the area is send to the COST class that calculate the area and display it.

Task-2

A point in the x-y plane is represented by its x-coordinate and y-coordinate. Design a class, point Type, that can store and process a point in the x-y plane. You should then perform operations on the point, such as setting the coordinates of the point, printing the coordinates of the point, returning the x-coordinate, and returning the y-coordinate. Also, write a program to test various operations on the point.

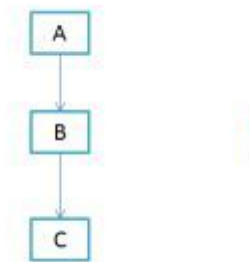
Assignment Question

Every circle has a center and a radius. Given the radius, we can determine the circle's area and circumference. Given the center, we can determine its position in the x-y plane. The center of the circle is a point in the x-y plane. Design a class, circleType, that can store the radius and center of the circle. Because the center is a point in the x-y plane and you designed the class to capture the properties of a point in Programming task 2, you must derive the class circleType from the class pointType. You should be able to perform the usual operations on the circle, such as setting the radius, printing the radius, calculating and printing the area and circumference, and carrying out the usual operations on the center. Also, write a program to test various operations on a circle.

OOP (multilevel inheritance)

LAB-6

Multilevel inheritance refers to a mechanism in OO technology where one can inherit from a derived class, thereby making this derived class the base class for the new class. As you can see in below flow diagram C is subclass or child class of B and B is a child class of A.



Example 1:

```
Class X
{
    public void methodX()
    {
        System.out.println("Class X method");
    }
}

Class Y extends X
{
    public void methodY()
    {
        System.out.println("class Y method");
    }
}
```

```
}

Class Z extends Y

{

    public void methodZ()

    {

        System.out.println("class Z method");

    }

    public static void main(String args[])

    {

        Z obj = new Z();

        obj.methodX(); //calling grand parent class method

        obj.methodY(); //calling parent class method

        obj.methodZ(); //calling local method

    }

}
```

Example 2:

```
class Shape {

    public void display() {

        System.out.println("Inside display");

    }

}

class Rectangle extends Shape {

    public void area() {

        System.out.println("Inside area");

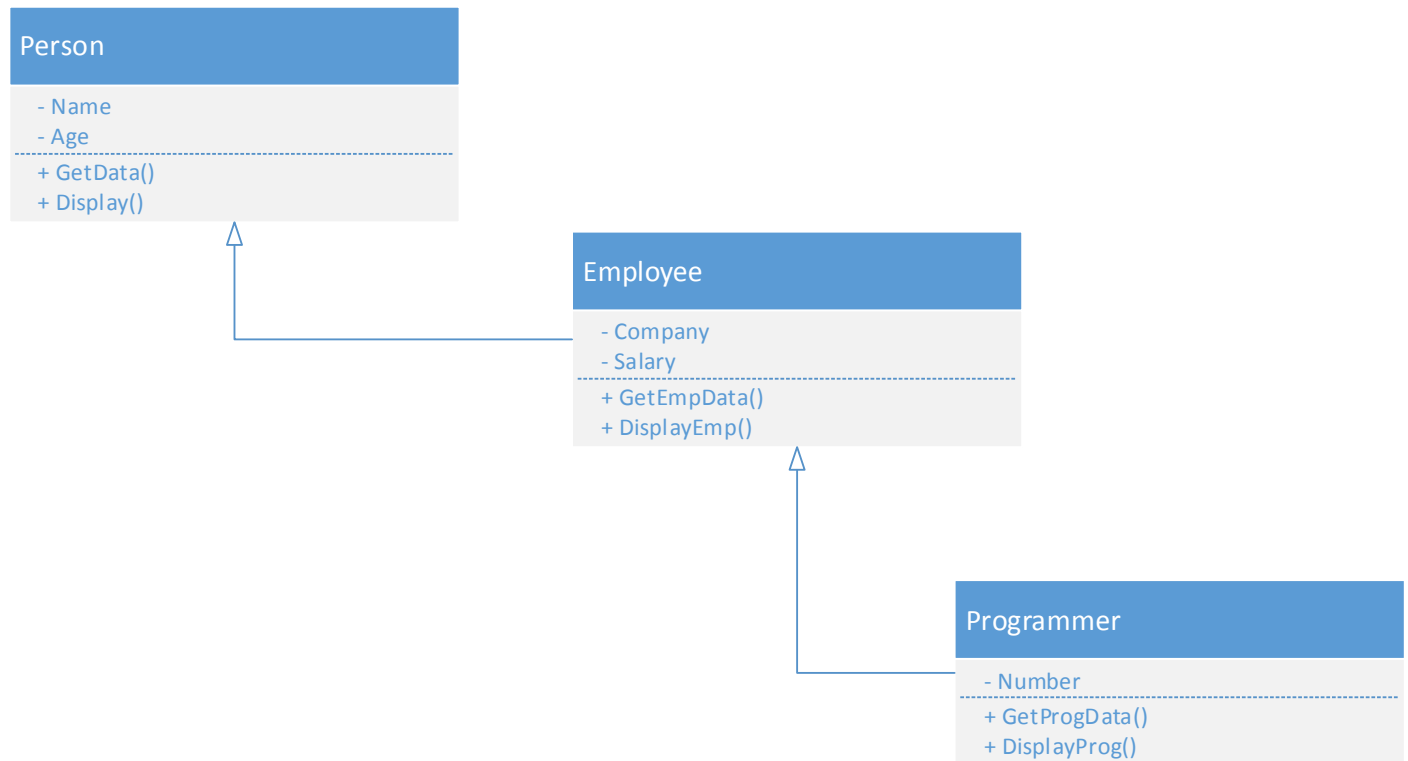
    }

}
```

```
class Cube extends Rectangle {  
  
    public void volume() {  
  
        System.out.println("Inside volume");  
  
    }  
}  
  
public class Tester {  
  
    public static void main(String[] arguments) {  
  
        Cube cube = new Cube();  
  
        cube.display();  
  
        cube.area();  
  
        cube.volume();  
  
    }  
}
```

Task-1

Write Java program to create a programmer derived from employee which is himself derived from person using Multilevel Inheritance. GetData(), GetEmpData(), GetProgData() functions take user inputs for Person, Employee and Programmer respectively. Display(), DisplayEmp() and DisplayProg() functions print data on scree for Person, Employee and Programmer respectively.



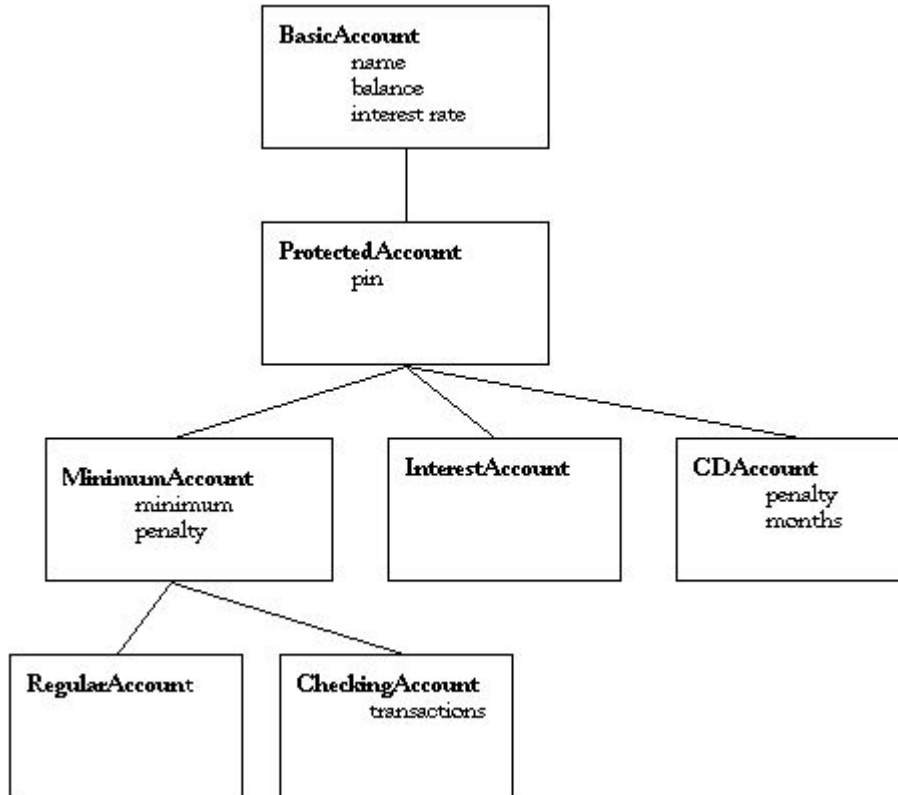
Task-2

In a bank, different customers have savings account. Some customers may have taken a loan from the bank. So, bank always maintains information about bank depositors and borrowers.

Design a Base class Customer (name, phone-number). Derive a class Depositor (accno, balance) from Customer. Again, derive a class Borrower (loan-no, loan-amt) from Depositor. Write necessary member functions to read and display the details of 'n' customers.

Assignment Question

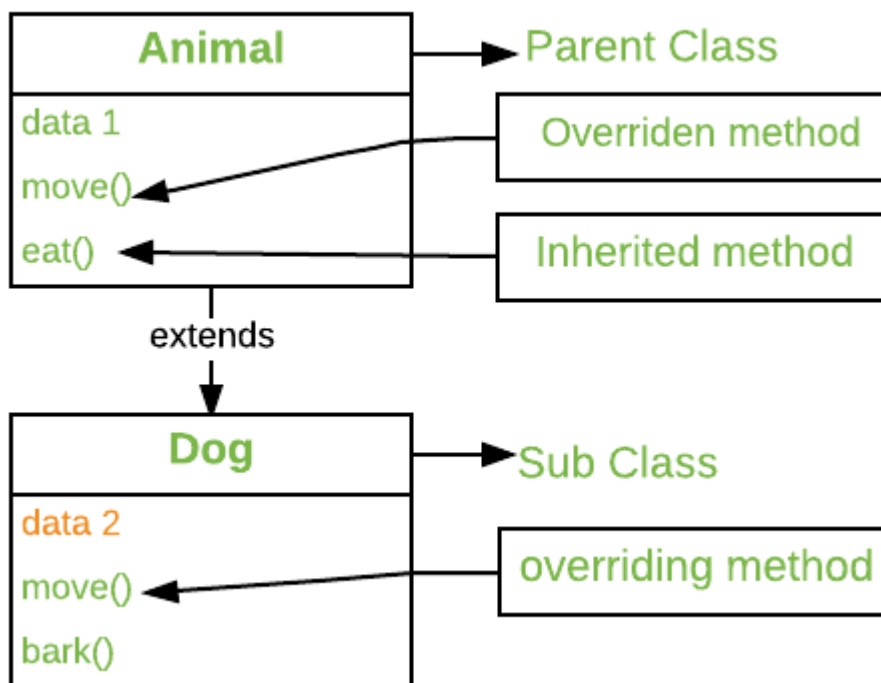
Write a java code to implement multilevel inheritance for following classes.



OOP (Function overriding)

LAB-7

In any object-oriented programming language, declaring a method in sub class which is already present in parent class is known as method overriding. Overriding is done so that a child class can give its own implementation to a method which is already provided by the parent class. In this case the method in parent class is called overridden method and the method in child class is called overriding method. When a method in a subclass has the same name, same parameters or signature and same return type (or sub-type) as a method in its super-class, then the method in the subclass is said to override the method in the super-class.



Method overriding is one of the way by which java achieve Run Time Polymorphism. The version of a method that is executed will be determined by the object that is used to invoke it. If an object of a parent class is used to invoke the method, then the version in the parent class will be executed, but if an object of the subclass is used to invoke the method, then the version in the child class will be executed. In other words, it is the type of the object being referred to (not the type of the reference variable) that determines which version of an overridden method will be executed.

Example:

```
// A Simple Java program to demonstrate
// method overriding in java

// Base Class
class Parent {
    void show()
    {
        System.out.println("Parent's show()");
    }
}

// Inherited class
class Child extends Parent {
    // This method overrides show() of Parent
    @Override
    void show()
    {
        System.out.println("Child's show()");
    }
}

// Driver class
class Main {
    public static void main(String[] args)
    {
        // If a Parent type reference refers
        // to a Parent object, then Parent's
        // show is called
        Parent obj1 = new Parent();
        obj1.show();

        // If a Parent type reference refers
        // to a Child object Child's show()
        // is called. This is called RUN TIME
        // POLYMORPHISM.
    }
}
```

```

        Parent obj2 = new Child();
        obj2.show();
    }
}

```

Task-1

Write a program that have three Classes “One” is base Class and “Two” and “Three” are child of base classes. Three Classes have same functions get() and display() . get() in “One” Class get two values and display the sum of these values and get() in “Two” Class get three number and display cube of that numbers. And in Class “Three” get four values and display the product of these values. Using “One” Class pointer.

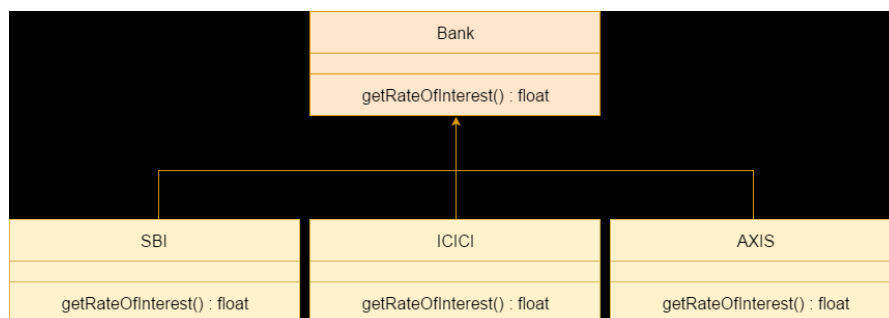
Task-2

Create a class called publication that stores the title (a string) and price (type float) of a publication. From this class derive two classes: book, which adds a page count (type int); and tape, which adds a playing time in minutes (type float). Each of the three classes should have a getdata() function to get its data from the user at the keyboard, and a put data() function to display the data.

Assignment Question

Write java code by considering a Consider a scenario where Bank is a class that provides functionality to get the rate of interest. However, the rate of interest varies according to banks. For example, SBI, ICICI and AXIS banks could provide 8%, 7%, and 9% rate of interest.

Hint: Use function float getRateOfInterest(){return 7;}



OOP (polymorphism)

LAB-8

Polymorphism is the ability of an object to take on many forms. The most common use of polymorphism in OOP occurs when a parent class reference is used to refer to a child class object.

```
class Animal {
    public void animalSound() {
        System.out.println("The animal makes a sound");
    }
}

class Pig extends Animal {
    public void animalSound() {
        System.out.println("The pig says: wee wee");
    }
}

class Dog extends Animal {
    public void animalSound() {
        System.out.println("The dog says: bow wow");
    }
}

class MyMainClass {
    public static void main(String[] args) {
        Animal myAnimal = new Animal(); // Create a Animal object
        Animal myPig = new Pig(); // Create a Pig object
        Animal myDog = new Dog(); // Create a Dog object
        myAnimal.animalSound();
        myPig.animalSound();
        myDog.animalSound();
    }
}
```

Task-1

Write a program that have three Classes “Animal” “Horse” “Duck” takes number of legs from the user according the nature of the animal and then display the legs according to the nature of the animal.

Task-2

Write the above program and in main using array take 10 choices from the user “1” for Horse “2” for Duck and finally display the Choices that how many time Duck select and how many time Horse select and display number of legs.

OOP (abstract classes and interfaces)

LAB-9

Data **abstraction** is the process of hiding certain details and showing only essential information to the user. Abstraction can be achieved with either **abstract classes** or interfaces.

The `abstract` keyword is a non-access modifier, used for classes and methods:

- **Abstract class:** is a restricted class that cannot be used to create objects (to access it, it must be inherited from another class).
- **Abstract method:** can only be used in an abstract class, and it does not have a body. The body is provided by the subclass (inherited from).

An abstract class can have both abstract and regular methods:

```
abstract class Animal {  
    public abstract void animalSound();  
    public void sleep() {  
        System.out.println("Zzz");  
    }  
}
```

From the example above, it is not possible to create an object of the `Animal` class:

```
Animal myObj = new Animal(); // will generate an error
```

Example:

```
// Abstract class  
abstract class Animal {  
    // Abstract method (does not have a body)  
    public abstract void animalSound();  
    // Regular method  
    public void sleep() {  
        System.out.println("Zzz");  
    }  
}
```

```
    }  
}  
  
// Subclass (inherit from Animal)  
class Pig extends Animal {  
    public void animalSound() {  
        // The body of animalSound() is provided here  
        System.out.println("The pig says: wee wee");  
    }  
}  
  
class MyMainClass {  
    public static void main(String[] args) {  
        Pig myPig = new Pig(); // Create a Pig object  
        myPig.animalSound();  
        myPig.sleep();  
    }  
}
```

Interface

An interface is a reference type in Java. It is similar to class. It is a collection of abstract methods. A class implements an interface, thereby inheriting the abstract methods of the interface.

Syntax:

```
/* File name : NameOfInterface.java */  
  
import java.lang.*;  
  
// Any number of import statements  
  
public interface NameOfInterface {  
  
    // Any number of final, static fields  
  
    // Any number of abstract method declarations\
```

```
}
```

Example:

```
// Interface
interface Animal {
    public void animalSound(); // interface method (does not have a body)
    public void sleep(); // interface method (does not have a body)
}

// Pig "implements" the Animal interface
class Pig implements Animal {
    public void animalSound() {
        // The body of animalSound() is provided here
        System.out.println("The pig says: wee wee");
    }
    public void sleep() {
        // The body of sleep() is provided here
        System.out.println("Zzz");
    }
}

class MyMainClass {
    public static void main(String[] args) {
        Pig myPig = new Pig(); // Create a Pig object
        myPig.animalSound();
        myPig.sleep();
    }
}
```

Multiple Interfaces

To implement multiple interfaces, separate them with a comma:

```
interface FirstInterface {
    public void myMethod(); // interface method
}
```

```
interface SecondInterface {
    public void myOtherMethod(); // interface method
}

class DemoClass implements FirstInterface, SecondInterface {
    public void myMethod() {
        System.out.println("Some text..");
    }
    public void myOtherMethod() {
        System.out.println("Some other text...");
    }
}

class MyMainClass {
    public static void main(String[] args) {
        DemoClass myObj = new DemoClass();
        myObj.myMethod();
        myObj.myOtherMethod();
    }
}
```

Task-1

Write a java program to create an abstract class named shape that contains two integers and an empty method named printArea() Provide three classes named Rectangle,, Triangle and Circle such that each one of the classes extends the class shape. Each one of the class contains only the method printArea() that print the area of the given shape.

Hint: To create an abstract class that shows the hiding of elements in a class. At the same time inheritance property is used to extend the class shape into different geometrical figures. This represents the reusability of code for a programmer.

Task-2

Provide an interface Measurable with a method double getMeasure() that measures an object in some way. Make Employee implement Measurable. Provide a method double

average(Measurable[] objects) that computes the average measure. Use it to compute the average salary of an array of employees.

Assignment Question

We have to calculate the percentage of marks obtained in three subjects (each out of 100) by student A and in four subjects (each out of 100) by student B. Create an abstract class 'Marks' with an abstract method 'getPercentage'. It is inherited by two other classes 'A' and 'B' each having a method with the same name which returns the percentage of the students. The constructor of student A takes the marks in three subjects as its parameters and the marks in four subjects as its parameters for student B. Create an object for each of the two classes and print the percentage of marks for both the students.

OOP (exception handling and file handling)

LAB-10

When executing Java code, different errors can occur: coding errors made by the programmer, errors due to wrong input, or other unforeseeable things. When an error occurs, Java will normally stop and generate an error message. The technical term for this is: Java will throw an exception (throw an error).

Java try and catch

The `try` statement allows you to define a block of code to be tested for errors while it is being executed.

The `catch` statement allows you to define a block of code to be executed, if an error occurs in the `try` block.

The `try` and `catch` keywords come in pairs:

Syntax

```
try {  
    // Block of code to try  
}  
catch(Exception e) {  
    // Block of code to handle errors  
}
```

Example

```
public class MyClass {  
    public static void main(String[] args) {  
        try {  
            int[] myNumbers = {1, 2, 3};  
            System.out.println(myNumbers[10]);  
        } catch (Exception e) {  
            System.out.println("Something went wrong.");  
        }  
    }  
}
```

```

    }
}
}

```

Java File Handling

File handling is an important part of any application. Java has several methods for creating, reading, updating, and deleting files. The File class from the java.io package, allows us to work with files. To use the File class, create an object of the class, and specify the filename or directory name.

Example

```
import java.io.File; // Import the File class
```

```
File myObj = new File("filename.txt"); // Specify the filename
```

Method	Type	Description
<code>canRead()</code>	Boolean	Tests whether the file is readable or not
<code>canWrite()</code>	Boolean	Tests whether the file is writable or not
<code>createNewFile()</code>	Boolean	Creates an empty file
<code>delete()</code>	Boolean	Deletes a file
<code>exists()</code>	Boolean	Tests whether the file exists
<code>getName()</code>	String	Returns the name of the file
<code>getAbsolutePath()</code>	String	Returns the absolute pathname of the file
<code>length()</code>	Long	Returns the size of the file in bytes
<code>list()</code>	String[]	Returns an array of the files in the directory
<code>mkdir()</code>	Boolean	Creates a directory

Create a File

To create a file in Java, you can use the `createNewFile()` method. This method returns a boolean value: true if the file was successfully created, and false if the file already exists. Note that the

method is enclosed in a try...catch block. This is necessary because it throws an IOException if an error occurs (if the file cannot be created for some reason).

Example

```
import java.io.File; // Import the File class
import java.io.IOException; // Import the IOException class to handle errors

public class CreateFile {
    public static void main(String[] args) {
        try {
            File myObj = new File("filename.txt");
            if (myObj.createNewFile()) {
                System.out.println("File created: " + myObj.getName());
            } else {
                System.out.println("File already exists.");
            }
        } catch (IOException e) {
            System.out.println("An error occurred.");
            e.printStackTrace();
        }
    }
}
```

Write To a File

In the following example, we use the FileWriter class together with its write() method to write some text to the file we created in the example above. Note that when you are done writing to the file, you should close it with the close() method:

Example

```
import java.io.FileWriter; // Import the FileWriter class
import java.io.IOException; // Import the IOException class to handle errors

public class WriteToFile {
    public static void main(String[] args) {
```

```
try {
    FileWriter myWriter = new FileWriter("filename.txt");
    myWriter.write("Files in Java might be tricky, but it is fun enough!");
    myWriter.close();
    System.out.println("Successfully wrote to the file.");
} catch (IOException e) {
    System.out.println("An error occurred.");
    e.printStackTrace();
}
}
```

Read a File

In the previous chapter, you learned how to create and write to a file. In the following example, we use the Scanner class to read the contents of the text file.

Example

```
import java.io.File; // Import the File class
import java.io.FileNotFoundException; // Import this class to handle errors
import java.util.Scanner; // Import the Scanner class to read text files

public class ReadFile {
    public static void main(String[] args) {
        try {
            File myObj = new File("filename.txt");
            Scanner myReader = new Scanner(myObj);
            while (myReader.hasNextLine()) {
                String data = myReader.nextLine();
                System.out.println(data);
            }
            myReader.close();
        } catch (FileNotFoundException e) {
            System.out.println("An error occurred.");
            e.printStackTrace();
        }
    }
}
```

```
}
```

Get File Information

To get more information about a file, use any of the File methods.

Example

```
import java.io.File; // Import the File class

public class GetFileInfo {
    public static void main(String[] args) {
        File myObj = new File("filename.txt");
        if (myObj.exists()) {
            System.out.println("File name: " + myObj.getName());
            System.out.println("Absolute path: " + myObj.getAbsolutePath());
            System.out.println("Writeable: " + myObj.canWrite());
            System.out.println("Readable " + myObj.canRead());
            System.out.println("File size in bytes " + myObj.length());
        } else {
            System.out.println("The file does not exist.");
        }
    }
}
```

Task-1

Write a Java program to get a list of all file/directory names from the given.

Hint: use `list()` function.

Task-2

Write a Java program to check if given pathname is a directory or a file.

Hint: use `isDirectory()` and `isFile()` Function of File.

Assignment Question

Write a program that will count the number of lines in each file that is specified on the command line. Assume that the files are text files. Note that multiple files can be specified, as in "`java LineCounts file1.txt file2.txt file3.txt`". Write each file name, along with the number of lines in that file, to standard output. If an error occurs while trying to read from one of the files, you should print an error message for that file, but you should still process all the remaining files.