# PRACTICAL 1

**Aim**: To create and run Java program using the basic constructs of the language.

**Prerequisite:**

* Basic knowledge of computer usage.

**Outcome:** After successful completion of this experiment students will be able to,

* Use Identifiers and create variables of different data types.
* Use Scanner class to take inputs from the console.
* Use Relational, Arithmetic, Increment and decrement operators, in their Java program
* Create and run simple Java programs.

**Theory:**

The following are the two methods to create and run a Java program.

**Using Notepad:**

1. **Open Notepad and type the following program:**

class Hello{

public static void main(String[] args){

System.out.println("Hello Everyone!");

}

}

1. **Save the program as Hello.java in the required directory**
2. **Open the command prompt:**

* Go to the directory in which you have saved your file.
* Set the path using the following code:

set path=C:\Program Files\Java\jdk1.6.0\_23\bin

* Type javac Hello.java
* Type java Hello

1. **Output obtained is:**

Hello Everyone

**Using Netbeans IDE:**

1. Start NetBeans IDE.
2. In the IDE, choose File > New Project,

* In the New Project wizard, expand the Java category and select Java Application and then click Next. In the Name and Location page of the wizard, do the following.
  + In the Project Name field, type name for your project.
  + Leave the Use Dedicated Folder for Storing Libraries checkbox unselected.

1. Click Finish

**Basic Constructs in Java:**

1. **Identifiers**

In programming languages, identifiers are used for identification purpose. In Java, an identifier can be a class name, method name, variable name or a label.

**Rules for defining Java Identifiers**

There are certain rules for defining a valid java identifier. These rules must be followed, otherwise we get compile-time error.

* Identifier cannot be a keyword.
* Identifiers are case-sensitive.
* It can have a sequence of letters and digits. However, it must begin with a letter, $ or \_. The first letter of an identifier cannot be a digit.
* It's convention to start an identifier with a letter rather and $ or \_.
* Whitespaces are not allowed.
* Similarly, you cannot use symbols such as @, #, and so on.

1. **Scanner Class in Java**

Scanner is a class in java.util package used for obtaining the input of the primitive types like int, double, etc. and strings. It is the easiest way to read input in a Java program, though not very efficient if you want an input method for scenarios where time is a constraint like in competitive programming.

* To create an object of Scanner class, we usually pass the predefined object System.in, which represents the standard input stream. We may pass an object of class File if we want to read input from a file.
* To read numerical values of a certain data type XYZ, the function to use is nextXYZ(). For example, to read a value of type short, we can use nextShort()
* To read strings, we use nextLine().
* To read a single character, we use next().charAt(0). next() function returns the next token/word in the input as a string and charAt(0) function returns the first character in that string.

The following are the list of some Scanner methods:

|  |  |
| --- | --- |
| **Method** | **Description** |
| nextBoolean() | Reads a boolean value from the user |
| nextByte() | Reads a byte value from the user |
| nextDouble() | Reads a double value from the user |
| nextFloat() | Reads a float value from the user |
| nextInt() | Reads a int value from the user |
| nextLine() | Reads a String value from the user |
| nextLong() | Reads a long value from the user |
| nextShort() | Reads a short value from the user |

1. **Operators in Java**

Java provides many types of operators which can be used according to the need. They are classified based on the functionality they provide. Some of the types are-

* [Arithmetic Operators](https://www.geeksforgeeks.org/operators-in-java/#Arithmetic%20Operators)
* [Unary Operators](https://www.geeksforgeeks.org/operators-in-java/#Unary%20Operators)
* [Assignment Operator](https://www.geeksforgeeks.org/operators-in-java/#Assignment%20Operator)
* [Relational Operators](https://www.geeksforgeeks.org/operators-in-java/#Relational%20Operators)
* [Logical Operators](https://www.geeksforgeeks.org/operators-in-java/#Logical%20Operators)

**Arithmetic Operators:** They are used to perform simple arithmetic operations on primitive data types.

* **\*:**Multiplication
* **/ :**Division
* **% :**Modulo
* **+ :**Addition
* **- :**Subtraction

**Unary Operators:** Unary operators need only one operand. They are used to increment, decrement or negate a value.

* **- :Unary minus**, used for negating the values.
* **+ :Unary plus**, used for giving positive values. Only used when deliberately converting a negative value to positive.
* **++ :Increment operator**, used for incrementing the value by 1. There are two varieties of increment operator.
* **Post-Increment :**Value is first used for computing the result and then incremented.
* **Pre-Increment :**Value is incremented first and then result is computed.
* **-: Decrement operator**, used for decrementing the value by 1. There are two varieties of decrement operator.
  + **Post-decrement :**Value is first used for computing the result and then decremented.
  + **Pre-Decrement :**Value is decremented first and then result is computed.
* **! : Logical not operator**, used for inverting a boolean value.

**Assignment Operator : ‘=’** Assignment operator is used to assign a value to any variable. It has a right to left associativity, i.e value given on right hand side of operator is assigned to the variable on the left and therefore right hand side value must be declared before using it or should be a constant.

**Relational Operators :** These operators are used to check for relations like equality, greater than, less than. They return boolean result after the comparison and are extensively used in looping statements as well as conditional if else statements.

Some of the relational operators are-

* **==, Equal to :**returns true of left hand side is equal to right hand side.
* **!=, Not Equal to :**returns true of left hand side is not equal to right hand side.
* **<, less than :**returns true of left hand side is less than right hand side.
* **<=, less than or equal to :**returns true of left hand side is less than or equal to right hand side.
* **>, Greater than :**returns true of left hand side is greater than right hand side.
* **>=, Greater than or equal to:**returns true of left hand side is greater than or equal to right hand side.

**Logical Operators:** These operators are used to perform “logical AND” and “logical OR” operation, i.e. the function similar to AND gate and OR gate in digital electronics. One thing to keep in mind is the second condition is not evaluated if the first one is false, i.e. it has a short-circuiting effect. Used extensively to test for several conditions for making a decision.  
Conditional operators are-

* **&&, Logical AND:**returns true when both conditions are true.
* **||, Logical OR:**returns true if at least one condition is true.

(TO BE COMPLETED BY STUDENTS)

|  |  |
| --- | --- |
| Roll No. K005 | Name: Jal Bafana |
| Class: Btech. Cyber Security | Batch: K1 |
| Date of Practical: 13.01.2025 | Date of Submission: 13.01.2025 |
| Grade: |  |

1. Assume that int a = 1 and double d = 1.0, and that each expression is independent. What are the results of the following expressions?

a = 46 / 9;

a = 46 % 9 + 4 \* 4 - 2;

a = 45 + 43 % 5 \* (23 \* 3 % 2);

a %= 3 / a + 3;

d = 4 + d \* d + 4;

d += 1.5 \* 3 + (++a);

d -= 1.5 \* 3 + a++;

public class ExpCheck{

public static void main(String[]args){

int a =1;

double d = 1.0;

a = 46/9;

System.out.println("Exp 1 = " + a);

a = 46 % 9 + 4 \* 4 - 2;

System.out.println("Exp 2 = " + a);

a = 45 + 43 % 5 \* (23 \* 3 % 2);

System.out.println("Exp 3 = " + a);

a %= 3 / a + 3;

System.out.println("Exp 4 = " + a);

d = 4 + d \* d + 4;

System.out.println("Exp 5 = " + d);

d += 1.5 \* 3 + (++a);

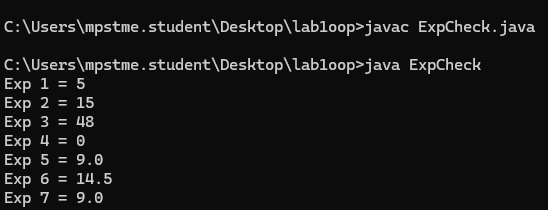
System.out.println("Exp 6 = " + d);

d -= 1.5 \* 3 + a++;

System.out.println("Exp 7 = " + d);

}

}



2. Write a program that reads an integer and checks whether it’s even or odd.

import java.util.Scanner;

public class OddEven{

public static void main(String[]args){

Scanner sc = new Scanner(System.in);

System.out.println("Enter a number = ");

int n = sc.nextInt();

if(n % 2 == 0){

System.out.println("The Number is even");

}

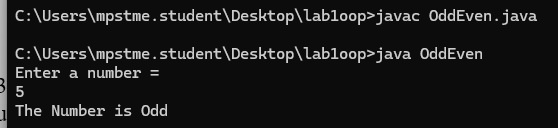
else{

System.out.println("The Number is Odd");

}

}

}



3. Program to implement solutions to the quadratic equation ax2 +bx+c = 0. Read in a, b, c and use the quadratic formula. If the discriminate b2 -4ac is negative, display a message stating that there are no real solutions.

Discriminate = 0 – roots are real and equal

Discriminate > 0 – roots are real and unequal (find x1, x2)

Discriminate < 0 – roots are imaginary

import java.util.Scanner;

public class Quadratic{

public static void main(String[]args){

Scanner sc = new Scanner(System.in);

System.out.println("Enter 3 numbers = ");

System.out.println("Enter 1st number = ");

double a = sc.nextDouble();

System.out.println("Enter 2nd number = ");

double b = sc.nextDouble();

System.out.println("Enter 3rd number = ");

double c = sc.nextDouble();

double dis = (b\*b) - (4\*a\*c);

if(dis == 0){

System.out.println("Roots are Real and Equal");

}

else if(dis > 0 ){

System.out.println("Roots are Real and Unequal ");

double rt1 = (-b + (Math.sqrt(dis)))/(2\*a);

double rt2 = (-b - (Math.sqrt(dis)))/(2\*a);

System.out.println("Roots are x1 = " + rt1 + " x2 = " + rt2);

}

else{

System.out.println("Roots are Imaginary");

}

}

}

