**Getting Input from the User.**

Four ways of reading input from the user in the command line environment.

1. Using Buffer Reader Class:   
   Wrapping the system in an Input Stream Reader which is wrapped in a Buffered Reader,   
   we can read input from the user in the command line. The input buffer is for efficient reading.

Syntax :

*import java.io.BufferReader;*

*import java.io.soException;*

*import java.io.InputStreamReader;*

*class Test*

*{*

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

*{*

*Buffered reader = new Buffered Reader (new InputStreamReader(System.in));*

*String name = reader.readLine();*

*System.out.println(name);*

*}*

*}*

1. Using Scanner Class:   
   Scanner class is used to input rare primitive types and strings using regular expressions however, it can be used. to read the input from the command line.

Convenient methods for parsing primitives(nextInt(), nextFloat(),etc.)

The reading methods are not synchronized.

Syntax :

*import java.util.\*;*

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

1. Using Console Class:   
   Used for reading password–like input without echoing the characters entered by the user; the format string syntax can also be used.

Reading methods are synchronized. Format String syntax can be used.

Syntax :

String name = System.console().readLine();

1. Using Command line argument:   
   The command line arguments are stored in the string format.   
   The parse int method of the integer class converts a string argument into an integer.   
   Similarly float and others during execution.

Syntax :

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

{

System.out.println(args[i]);

}

Example

import java.util.Scanner;

class inputeg

{

public static void main(String args[])

{

Scanner input = new Scanner(System.in); //input is a variable

System.out.print("Enter Name: ");

String Name = input.nextLine();

System.out.print("Enter Gender: ");

String Gender = input.next();

System.out.print("Enter Age: ");

int Age = input.nextInt();

System.out.print("Enter Phone no.: ");

long number = in.nextLong();

System.out.print("Enter CGPA: ");

float cgpa = in.nextFloat();

System.out.println("Name: " + Name);

System.out.println("Age: " + Age);

System.out.println("Gender: "+ Gender);

System.out.println("Phone no.: "+ number);

System.out.println("CGPA: "+ cgpa);

}

}

* OPERATORS

Relational Operators:

⦁== (Equal to)– This operator is used to check if both operands are equal.

⦁!= (Not equal to)– Can check if both operands are not equal.

⦁>(Greater than)– Can check if the first operand is greater than the second.

⦁< (Less than)- Can check if the first operand is lesser than the second.

⦁>= (Greater than equal to)– Check if the first operand is greater than or equal to the second.

⦁<= (Less than equal to)– Check if the first operand is lesser than or equal to the second

Logical Operators:

⦁&& (AND) – It is used to check if both the operands are true.

⦁|| (OR) – These operators are used to check if at least one of the operands is true.

⦁! (NOT) – Used to check if the operand is false

Bitwise Operators:

⦁ & (Bitwise AND) – Converts the value of both the operands into binary form and

performs AND operation bit by bit.

⦁ | (Bitwise OR) – Converts the value of both the operands into binary form and performs

OR operation bit by bit.

⦁ ^ (Bitwise exclusive OR) – Converts the value of both the operands into binary form and performs EXCLUSIVE OR operation bit by bit.

Function (static):

A static method is a method that belongs to a class rather than an instance of a class. The

method is accessible to every instance of a class, but methods defined in an instance are only

able to access by that object of the class. A static method is not part of the class definition.

Unlike instance methods, a static method is referred to by the class name can be invoked

without creating an object of the class.