## What is Java?

Java is a **programming language** and a **platform**. Java is a high level, robust, object-oriented and secure programming language.

## Types of Java Applications?

#### 1) Standalone Application

#### 2) Web Application

#### 3) Enterprise Application

#### 4) Mobile Application

# Features of Java?

### Simple

Java is very easy to learn, and its syntax is simple, clean and easy to understand. According to Sun Microsystem, Java language is a simple programming language

### Object-oriented

Java is an [object-oriented](https://www.javatpoint.com/java-oops-concepts) programming language. Everything in Java is an object. Object-oriented means we organize our software as a combination of different types of objects that incorporate both data and behavior.

1. [Object](https://www.javatpoint.com/object-and-class-in-java)
2. [Class](https://www.javatpoint.com/object-and-class-in-java#class)
3. [Inheritance](https://www.javatpoint.com/inheritance-in-java)
4. [Polymorphism](https://www.javatpoint.com/runtime-polymorphism-in-java)
5. [Abstraction](https://www.javatpoint.com/abstract-class-in-java)
6. [Encapsulation](https://www.javatpoint.com/encapsulation)

Platform Independent

Java is platform independent because it is different from other languages like [C](https://www.javatpoint.com/c-programming-language-tutorial), [C++](https://www.javatpoint.com/cpp-tutorial), etc. which are compiled into platform specific machines while Java is a write once, run anywhere language. A platform is the hardware or software environment in which a program runs.

Java code can be executed on multiple platforms, for example, Windows, Linux, Sun Solaris, Mac/OS, etc. Java code is compiled by the compiler and converted into bytecode. This bytecode is a platform-independent code because it can be run on multiple platforms, i.e., Write Once and Run Anywhere (WORA).

Secured

Java is best known for its security. With Java, we can develop virus-free systems. Java is secured because:

* **No explicit pointer**
* **Java Programs run inside a virtual machine sandbox**

Robust

The English mining of Robust is strong. Java is robust because:

* It uses strong memory management.
* There is a lack of pointers that avoids security problems.
* Java provides automatic garbage collection which runs on the Java Virtual Machine to get rid of objects which are not being used by a Java application anymore.

### Portable

Java is portable because it facilitates you to carry the Java bytecode to any platform. It doesn't require any implementation.

### High-performance

Java is faster than other traditional interpreted programming languages because Java bytecode is "close" to native code. It is still a little bit slower than a compiled language (e.g., C++).

### Multi-threaded

A thread is like a separate program, executing concurrently. We can write Java programs that deal with many tasks at once by defining multiple threads. The main advantage of multi-threading is that it doesn't occupy memory for each thread. It shares a common memory area. Threads are important for multi-media, Web applications, etc.

### Dynamic

Java is a dynamic language. It supports the dynamic loading of classes. It means classes are loaded on demand. It also supports functions from its native languages, i.e., C and C++.

## Parameters used in First Java Program?

Let's see what is the meaning of class, public, static, void, main, String[], System.out.println().

* **class** keyword is used to declare a class in Java.
* **public** keyword is an access modifier that represents visibility. It means it is visible to all.
* **static** is a keyword. If we declare any method as static, it is known as the static method. The core advantage of the static method is that there is no need to create an object to invoke the static method. The main() method is executed by the JVM, so it doesn't require creating an object to invoke the main() method. So, it saves memory.
* **void** is the return type of the method. It means it doesn't return any value.
* **main** represents the starting point of the program.
* **String[] args** or **String args[]** is used for [command line argument](https://www.javatpoint.com/command-line-argument). We will discuss it in coming section.
* **System.out.println()** is used to print statement. Here, System is a class, out is an object of the PrintStream class, println() is a method of the PrintStream class. We will discuss the internal working of [System.out.println()](https://www.javatpoint.com/system-out-println-in-java) statement in the coming section.

## What happens at compile time?

At compile time, the Java file is compiled by Java Compiler (It does not interact with OS) and converts the Java code into bytecode.

### JVM?

JVM (Java Virtual Machine) is an abstract machine. It is called a virtual machine because it doesn't physically exist. It is a specification that provides a runtime environment in which Java bytecode can be executed. It can also run those programs which are written in other languages and compiled to Java bytecode.

The JVM performs the following main tasks:

* Loads code
* Verifies code
* Executes code
* Provides runtime environment

### JRE?

The Java Runtime Environment is a set of software tools which are used for developing Java applications. It is used to provide the runtime environment. It is the implementation of JVM. It physically exists. It contains a set of libraries + other files that JVM uses at runtime.



### JDK?

JDK is an acronym for Java Development Kit. The Java Development Kit (JDK) is a software development environment which is used to develop Java applications and [applets](https://www.javatpoint.com/java-applet)

### . It physically exists. It contains JRE + development tools

### JDK

# Java Variables?

A variable is a container which holds the value while the [Java program](https://www.javatpoint.com/simple-program-of-java) is executed. A variable is assigned with a data type.

Types of Variables

There are three types of variables in [Java](https://www.javatpoint.com/java-tutorial):

* local variable
* instance variable
* static variable

#### 1) Local Variable

A variable declared inside the body of the method is called local variable. You can use this variable only within that method .A local variable cannot be defined with "static" keyword.

#### 2) Instance Variable

A variable declared inside the class but outside the body of the method, is called an instance variable. It is not declared as [static](https://www.javatpoint.com/static-keyword-in-java).

It is called an instance variable because its value is instance-specific and is not shared among instances.

#### 3) Static variable

A variable that is declared as static is called a static variable. It cannot be local. You can create a single copy of the static variable and share it among all the instances of the class. Memory allocation for static variables happens only once when the class is loaded in the memory.

### Example to understand the types of variables in java

1. **public** **class** A
2. {
3. **static** **int** m=100;//static variable
4. **void** method()
5. {
6. **int** n=90;//local variable
7. }
8. **public** **static** **void** main(String args[])
9. {
10. **int** data=50;//instance variable
11. }
12. }//end of class

# Data Types in Java?

Data types specify the different sizes and values that can be stored in the variable. There are two types of data types in Java:

1. **Primitive data types:** The primitive data types include boolean, char, byte, short, int, long, float and double.
2. **Non-primitive data types:** The non-primitive data types include [Classes](https://www.javatpoint.com/object-and-class-in-java), [Interfaces](https://www.javatpoint.com/interface-in-java), [Arrays](https://www.javatpoint.com/array-in-java), ArrayList and Hashmaps

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Type** | **Default Value** | **Default size** |  |
| boolean | false | 1 bit | True /false |
| char | '\u0000' | 2 byte | ‘A’ |
| byte | 0 | 1 byte | 1 |
| short | 0 | 2 byte |  |
| int | 0 | 4 byte | 10 |
| long | 0L | 8 byte |  |
| float | 0.0f | 4 byte | 231.5 |
| double | 0.0d | 8 byte | 10.5 |
| String |  |  | “Welcome"; |

# Operators in Java

1)Arithmetic Operators----> + - \* / %

2)Relational Operators------> == < > <= >= !=

3)Logical Operators--------> && || !

4)Increment and Decrement Operators------> ++ --

5)Assignment Operators------> =

3)Logical Operators--------> && || !`+

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x | y | x&y | x||y | !x | !y |
| TRUE | TRUE | TRUE | TRUE | FALSE | FALSE |
| TRUE | FALSE | FALSE | TRUE | FALSE | TRUE |
| FALSE | TRUE | FALSE | TRUE | TRUE | FALSE |
| FALSE | FALSE | FALSE | FALSE | TRUE | TRUE |

## Java Operator Precedence

|  |  |  |
| --- | --- | --- |
| **Operator Type** | **Category** | **Precedence** |
| Unary | postfix | expr++ expr-- |
| prefix | ++expr --expr +expr -expr ~ ! |
| Arithmetic | multiplicative | \* / % |
| additive | + - |
| Shift | shift | << >> >>> |
| Relational | comparison | < > <= >= instanceof |
| equality | == != |
| Bitwise | bitwise AND | & |
| bitwise exclusive OR | ^ |
| bitwise inclusive OR | | |
| Logical | logical AND | && |
| logical OR | || |
| Ternary | ternary | ? : |
| Assignment | assignment | = += -= \*= /= %= &= ^= |= <<= >>= >>>= |

## 

## List of Java Keywords?

1. [**abstract**](https://www.javatpoint.com/abstract-keyword-in-java)**:** Java abstract keyword is used to declare an abstract class. An abstract class can provide the implementation of the interface. It can have abstract and non-abstract methods.
2. [**boolean:**](https://www.javatpoint.com/boolean-keyword-in-java) Java boolean keyword is used to declare a variable as a boolean type. It can hold True and False values only.
3. [**break**](https://www.javatpoint.com/java-break)**:** Java break keyword is used to break the loop or switch statement. It breaks the current flow of the program at specified conditions.
4. [**byte**](https://www.javatpoint.com/byte-keyword-in-java)**:** Java byte keyword is used to declare a variable that can hold 8-bit data values.
5. [**case**](https://www.javatpoint.com/case-keyword-in-java)**:** Java case keyword is used with the switch statements to mark blocks of text.
6. [**catch**](https://www.javatpoint.com/try-catch-block)**:** Java catch keyword is used to catch the exceptions generated by try statements. It must be used after the try block only.
7. [**char**](https://www.javatpoint.com/char-keyword-in-java)**:** Java char keyword is used to declare a variable that can hold unsigned 16-bit Unicode characters
8. [**class**](https://www.javatpoint.com/class-keyword-in-java)**:** Java class keyword is used to declare a class.
9. [**continue**](https://www.javatpoint.com/java-continue)**:** Java continue keyword is used to continue the loop. It continues the current flow of the program and skips the remaining code at the specified condition.
10. [**default**](https://www.javatpoint.com/default-keyword-in-java)**:** Java default keyword is used to specify the default block of code in a switch statement.
11. [**do**](https://www.javatpoint.com/java-do-while-loop)**:** Java do keyword is used in the control statement to declare a loop. It can iterate a part of the program several times.
12. [**double**](https://www.javatpoint.com/double-keyword-in-java)**:** Java double keyword is used to declare a variable that can hold 64-bit floating-point number.
13. [**else**](https://www.javatpoint.com/java-if-else)**:** Java else keyword is used to indicate the alternative branches in an if statement.
14. [**enum**](https://www.javatpoint.com/enum-in-java)**:** Java enum keyword is used to define a fixed set of constants. Enum constructors are always private or default.
15. [**extends**](https://www.javatpoint.com/inheritance-in-java)**:** Java extends keyword is used to indicate that a class is derived from another class or interface.
16. [**final**](https://www.javatpoint.com/final-keyword)**:** Java final keyword is used to indicate that a variable holds a constant value. It is used with a variable. It is used to restrict the user from updating the value of the variable.
17. [**finally**](https://www.javatpoint.com/finally-block-in-exception-handling)**:** Java finally keyword indicates a block of code in a try-catch structure. This block is always executed whether an exception is handled or not.
18. [**float**](https://www.javatpoint.com/float-keyword-in-java)**:** Java float keyword is used to declare a variable that can hold a 32-bit floating-point number.
19. [**for**](https://www.javatpoint.com/java-for-loop)**:** Java for keyword is used to start a for loop. It is used to execute a set of instructions/functions repeatedly when some condition becomes true. If the number of iteration is fixed, it is recommended to use for loop.
20. [**if**](https://www.javatpoint.com/java-if-else)**:** Java if keyword tests the condition. It executes the if block if the condition is true.
21. [**implements**](https://www.javatpoint.com/interface-in-java)**:** Java implements keyword is used to implement an interface.
22. [**import**](https://www.javatpoint.com/package)**:** Java import keyword makes classes and interfaces available and accessible to the current source code.
23. [**instanceof**](https://www.javatpoint.com/downcasting-with-instanceof-operator)**:** Java instanceof keyword is used to test whether the object is an instance of the specified class or implements an interface.
24. [**int**](https://www.javatpoint.com/int-keyword-in-java)**:** Java int keyword is used to declare a variable that can hold a 32-bit signed integer.
25. [**interface**](https://www.javatpoint.com/interface-in-java)**:** Java interface keyword is used to declare an interface. It can have only abstract methods.
26. [**long**](https://www.javatpoint.com/long-keyword-in-java)**:** Java long keyword is used to declare a variable that can hold a 64-bit integer.
27. **native:** Java native keyword is used to specify that a method is implemented in native code using JNI (Java Native Interface).
28. [**new**](https://www.javatpoint.com/new-keyword-in-java)**:** Java new keyword is used to create new objects.
29. [**null**](https://www.javatpoint.com/null-keyword-in-java)**:** Java null keyword is used to indicate that a reference does not refer to anything. It removes the garbage value.
30. [**package**](https://www.javatpoint.com/package)**:** Java package keyword is used to declare a Java package that includes the classes.
31. [**private**](https://www.javatpoint.com/private-keyword-in-java)**:** Java private keyword is an access modifier. It is used to indicate that a method or variable may be accessed only in the class in which it is declared.
32. [**protected**](https://www.javatpoint.com/protected-keyword-in-java)**:** Java protected keyword is an access modifier. It can be accessible within the package and outside the package but through inheritance only. It can't be applied with the class.
33. [**public**](https://www.javatpoint.com/public-keyword-in-java)**:** Java public keyword is an access modifier. It is used to indicate that an item is accessible anywhere. It has the widest scope among all other modifiers.
34. [**return**](https://www.javatpoint.com/return-keyword-in-java)**:** Java return keyword is used to return from a method when its execution is complete.
35. [**short**](https://www.javatpoint.com/short-keyword-in-java)**:** Java short keyword is used to declare a variable that can hold a 16-bit integer.
36. [**static**](https://www.javatpoint.com/static-keyword-in-java)**:** Java static keyword is used to indicate that a variable or method is a class method. The static keyword in Java is mainly used for memory management.
37. [**strictfp**](https://www.javatpoint.com/strictfp-keyword)**:** Java strictfp is used to restrict the floating-point calculations to ensure portability.
38. [**super**](https://www.javatpoint.com/super-keyword)**:** Java super keyword is a reference variable that is used to refer to parent class objects. It can be used to invoke the immediate parent class method.
39. [**switch**](https://www.javatpoint.com/java-switch)**:** The Java switch keyword contains a switch statement that executes code based on test value. The switch statement tests the equality of a variable against multiple values.
40. [**synchronized**](https://www.javatpoint.com/synchronization-in-java)**:** Java synchronized keyword is used to specify the critical sections or methods in multithreaded code.
41. [**this**](https://www.javatpoint.com/this-keyword)**:** Java this keyword can be used to refer the current object in a method or constructor.
42. [**throw**](https://www.javatpoint.com/throw-keyword)**:** The Java throw keyword is used to explicitly throw an exception. The throw keyword is mainly used to throw custom exceptions. It is followed by an instance.
43. [**throws**](https://www.javatpoint.com/throws-keyword-and-difference-between-throw-and-throws)**:** The Java throws keyword is used to declare an exception. Checked exceptions can be propagated with throws.
44. [**transient**](https://www.javatpoint.com/transient-keyword)**:** Java transient keyword is used in serialization. If you define any data member as transient, it will not be serialized.
45. [**try**](https://www.javatpoint.com/try-catch-block)**:** Java try keyword is used to start a block of code that will be tested for exceptions. The try block must be followed by either catch or finally block.
46. **void:** Java void keyword is used to specify that a method does not have a return value.
47. [**volatile**](https://www.javatpoint.com/volatile-keyword-in-java)**:** Java volatile keyword is used to indicate that a variable may change asynchronously.
48. [**while**](https://www.javatpoint.com/java-while-loop)**:** Java while keyword is used to start a while loop. This loop iterates a part of the program several times. If the number of iteration is not fixed, it is recommended to use the while loop

# Java Control Statements | Control Flow in Java

1. Decision Making statements/Conditional statament(selection statament)
   * if statements
   * switch statement
2. Loop statements
   * do while loop
   * while loop
   * for loop
   * for-each loop
3. Jump statements
   * break statement
   * continue statement

### Decision-Making statements:

As the name suggests, decision-making statements decide which statement to execute and when. Decision-making statements evaluate the Boolean expression and control the program flow depending upon the result of the condition provided. There are two types of decision-making statements in Java, i.e., If statement and switch statement.

### 1) If Statement:

In Java, the "if" statement is used to evaluate a condition. The control of the program is diverted depending upon the specific condition. The condition of the If statement gives a Boolean value, either true or false.

### if..else

if(condition)

{

stataments;

}

else

{

stataments;

}



### Nested if..else

if(condition)

{

stataments;

}

else if(condition1)

{

stataments;

}

else if(condition2)

{

stataments;

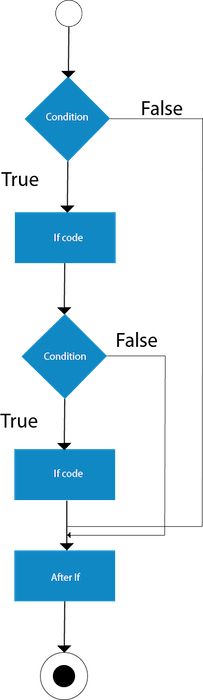
}

else

{

stataments;

}



### switch case

### The Java switch statement executes one statement from multiple conditions.

switch(expression)

{

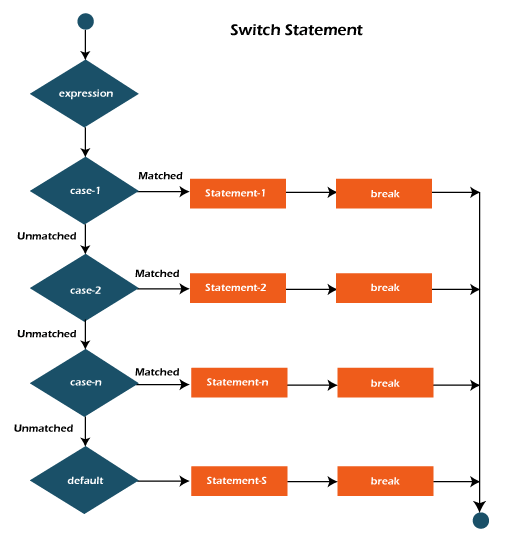
case <condition1>: stataments; break;

case <condition2>: stataments; break;

case <condition3>: stataments; break;

default:stataments;

}



### Loop Statements

In programming, sometimes we need to execute the block of code repeatedly while some condition evaluates to true. However, loop statements are used to execute the set of instructions in a repeated order. The execution of the set of instructions depends upon a particular condition.

1. for loop
2. while loop
3. do-while loop

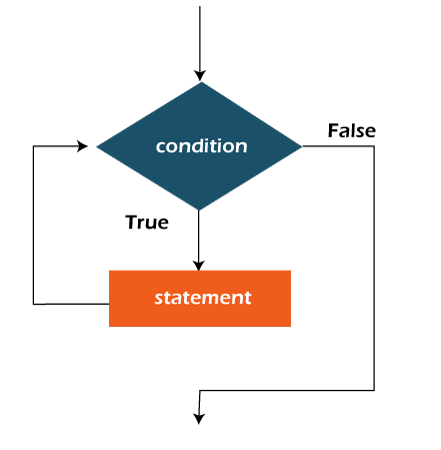
### while loop

while(condition)

{

stataments;

}



### do..while

do

{

statament;

}

while(condition);

### for loop

initilization

condition

increment/decrement

for(initilization;condition;inc/dec)

{

statament;

}



Jump Statements

Jump statements are used to transfer the control of the program to the specific statements. In other words, jump statements transfer the execution control to the other part of the program. There are two types of jump statements in Java, i.e., break and continue.

Java break statement

As the name suggests, the [break statement](https://www.javatpoint.com/java-break)

is used to break the current flow of the program and transfer the control to the next statement outside a loop or switch statement. However, it breaks only the inner loop in the case of the nested loop

.

Java continue statement

Unlike break statement, the [continue statement](https://www.javatpoint.com/java-continue)

doesn't break the loop, whereas, it skips the specific part of the loop and jumps to the next iteration of the loop immediately.

# Java Comments

comments are the statements in a program that are not executed by the compiler and interpreter.

## Why do we use comments in a code?

* Comments are used to make the program more readable by adding the details of the code.
* It makes easy to maintain the code and to find the errors easily.
* The comments can be used to provide information or explanation about the [variable](https://www.javatpoint.com/java-variables)

, method, [class](https://www.javatpoint.com/object-and-class-in-java)

, or any statement.

* It can also be used to prevent the execution of program code while testing the alternative code.

## Types of Java Comments

1. Single Line Comment : Single line comments starts with two forward slashes **(//)**. Any text in front of // is not executed by Java.
2. Multi Line Comment : Multi-line comments are placed between /\* and \*/. Any text between /\* and \*/ is not executed by Java.
3. Documentation Comment : The documentation comments are placed between /\*\* and \*/.

## OOPs (Object-Oriented Programming System)

**Object** means a real-world entity such as a pen, chair, table, computer, watch, etc. **Object-Oriented Programming** is a methodology or paradigm to design a program using classes and objects.

* [Object](https://www.javatpoint.com/object-and-class-in-java)
* Class
* [Inheritance](https://www.javatpoint.com/inheritance-in-java)
* [Polymorphism](https://www.javatpoint.com/runtime-polymorphism-in-java)
* [Abstraction](https://www.javatpoint.com/abstract-class-in-java)
* [Encapsulation](https://www.javatpoint.com/encapsulation)

## Object



Any entity that has state and behavior is known as an object. For example, a chair, pen, table, keyboard, bike, etc. It can be physical or logical.

An Object can be defined as an instance of a class. An object contains an address and takes up some space in memory. Objects can communicate without knowing the details of each other's data or code. The only necessary thing is the type of message accepted and the type of response returned by the objects.

**Example:** A dog is an object because it has states like color, name, breed, etc. as well as behaviors like wagging the tail, barking, eating, etc.

## Class

Collection of objects is called class. It is a logical entity.

A class can also be defined as a blueprint from which you can create an individual object. Class doesn't consume any space.

Inheritance

*When one object acquires all the properties and behaviors of a parent object*, it is known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.



Polymorphism

If *one task is performed in different ways*, it is known as polymorphism. For example: to convince the customer differently, to draw something, for example, shape, triangle, rectangle, etc.

In Java, we use method overloading and method overriding to achieve polymorphism.

Another example can be to speak something; for example, a cat speaks meow, dog barks woof, etc.

#### Abstraction

Hiding internal details and showing functionality is known as abstraction. For example phone call, we don't know the internal processing.

In Java, we use abstract class and interface to achieve abstraction.

Encapsulation



*Binding (or wrapping) code and data together into a single unit are known as encapsulation*. For example, a capsule, it is wrapped with different medicines.

A java class is the example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here.

## Advantage of OOPs over Procedure-oriented programming language

1) OOPs makes development and maintenance easier, whereas, in a procedure-oriented programming language, it is not easy to manage if code grows as project size increases.

2) OOPs provides data hiding, whereas, in a procedure-oriented programming language, global data can be accessed from anywhere.

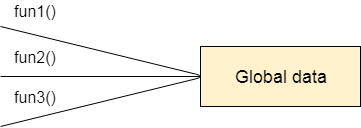


Figure: Data Representation in Procedure-Oriented Programming

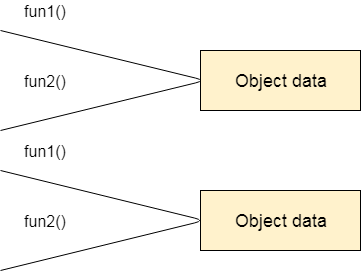


Figure: Data Representation in Object-Oriented Programming

3) OOPs provides the ability to simulate real-world event much more effectively. We can provide the solution of real word problem if we are using the Object-Oriented Programming language.

## Array:it is collection of element of same data type.

## 2types of array

1. single dimensional array

2. two/multi dimensional array

1.single dimensional array

1.Declare an array

2.insert values into array

3.how to find the size of an array

4.how to read value from an array

int a[]=new int[5]; //int []a=new int[5];

clasic for loop

for(initilization;condition;incrementation)

{

statements;

}

Advanced for loop/Enhanced for loop/for each loop

for(initilization)

{

statements;

}

2. two/multi dimensional array

1.Declare an array

2.insert values into array

3.how to find the siez of an array

4.how to read vaule from an array

int a[][]=new int[3][2]; // int [][]a=new int[3][2];

3->row 2->columns

String is colection of characters

String s="Welcome"

string methods

length()

concat()

trim()

contains()

equals()

equalsIgnoreCase()

replace()

toLowerCase()

toUpperCase()

OOP- object oriented programming.

class-->collection of variables and methods

object-->object is an instance of a class

method-->block of code which contains a logic

1)main() method can be prasent in a same class

2)main() method can be present in a saparate class

class varibles

we can assign values to variables in 3ways

1)by using reference variable

2)by using method

3)by using constuctore

constuctore

1)constuctore a special type of method

2)constuctore udes for initilizing the class variabls

3)constuctore name should be same as class name

4)constuctore will not return any value(not even void

5)constuctore will be invoked at the time of object creation

method-->Block of code

1)case1-not taking parameters and also not returned any value

2)case2-not taking parameters but returning value

3)case3-taking parameters, but not returning any value

4)case4-method is taking parameters and also returning values

type of constructor

1)default Constructor:A constructor is called "Default Constructor" when it doesn't have any parameter.

2)parameter Constructor:A constructor which has a specific number of parameters is called a parameterized constructor.

Java inheritance

method overriding

super keyword

final keyword

Inheritance : Aquiring all the properties & behavior from one class to another class is called as inheritence.

perent/base class/super class

child class/sub class/derived class

5type of inheritence

1)single

2)multilevel

3)hierarchical

4)multiple

5)hybrid



#### Note: Multiple inheritance is not supported in Java through class.

## Single Inheritance Example

When a class inherits another class, it is known as a single inheritance. In the example given below, Dog class inherits the Animal class, so there is the single inheritance.

## Multilevel Inheritance Example

When there is a chain of inheritance, it is known as multilevel inheritance. As you can see in the example given below, BabyDog class inherits the Dog class which again inherits the Animal class, so there is a multilevel inheritance.

## Hierarchical Inheritance Example

When two or more classes inherits a single class, it is known as hierarchical inheritance. In the example given below, Dog and Cat classes inherits the Animal class, so there is hierarchical inheritance.

## Multiple inheritance in Java by interface

If a class implements multiple interfaces, or an interface extends multiple interfaces, it is known as multiple inheritance.

## Q) Why multiple inheritance is not supported in java?

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

# Method Overloading in Java

If a [class](https://www.javatpoint.com/object-and-class-in-java) has multiple methods having same name but different in parameters, it is known as **Method Overloading**.

rewrite the method in child classes

1)method definition should be same

2)body should be changed

# Super Keyword in Java

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.

# Final: The final keyword can be applied with the variables, a final variable that have no value it is called blank final variable or uninitialized final variable.

1. variable-->you can not chage the value of the variable
2. class-->method can not be overridein the child classes
3. method-->class cannot be extended into another

# Interfaces

1. An interface in java is a blueprint of class.
2. Interface contains final and static variables.
3. Interface contains abstract methods.
4. An abstract method is a method contains definition but not a body.
5. methods in interfaces are public by default.
6. interface supports the functionality of multiple interfaces.
7. we can define interface with interface keyword.
8. A class extends lass, an interface extends another interface but class impliments an interface.
9. we can create object reference for interfacebut we can not instantiate interface.

#### The relationship between classes and interfaces



*package* is collection of classes/interfaces/other files.

# Access Modifiers in Java

1. **Private**: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.
2. **Default**: The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.
3. **Protected**: The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.
4. **Public**: The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Access Modifier** | **within class** | **within package** | **outside package by subclass only** | **outside package** |
| **Private** | Y | N | N | N |
| **Default** | Y | Y | N | N |
| **Protected** | Y | Y | Y | N |
| **Public** | Y | Y | Y | Y |

# Exception handling

Exception is an abnormal condition.

In java, exception is an event that disrupts the normal flow of the program.

# Type of exceptions

1)checked exception: exceptions checked/identified by the compiler.

**examples:**

Interrupted exception

Ioexception

File Not found exception

2)un-checked exception:exceptions which are not checked/identified by the compiler.

**examples:**

ErithmeticException

NullpointerException

NumberformatException

ArrayIndexOutOfBoundsException

**ErithmeticException:**

ex-> int a=50/0

**NullpointerException**

ex-> String s=null;

System.out.println(s.length());

**NumberformatException**

ex-> string s="abc";

int i=integer.parselnt(s);

**ArrayIndexOutOfBoundsException**

ex-> int a[] =new int[5];

a[10]=50;

# handaling exceptions

**Syntax1:**

try

{

statments;

}

catch(AithmeticException type)

{

statments;

}

**Syntax2:**

try

{

statments;

}

catch(Exception1 type)

{

statments;

}

catch(Exception2 type)

{

statments;

}

**Syntax3:**

try

{

statments;(to open a file)

}

catch(Exception type)

{

statments;

}

finally

{

close

}

one try block can have multuple catch blocks

# finally block is always executes....

1)Exception doesn't occur.

2)Exception occurs and not handled.

3)Exception occurs and handled.

# checked exceptions can be handled in 2ways

1)try..catch

2)throws keyword

|  |  |
| --- | --- |
| **Keyword** | **Description** |
| try | The "try" keyword is used to specify a block where we should place an exception code. It means we can't use try block alone. The try block must be followed by either catch or finally. |
| catch | The "catch" block is used to handle the exception. It must be preceded by try block which means we can't use catch block alone. It can be followed by finally block later. |
| finally | The "finally" block is used to execute the necessary code of the program. It is executed whether an exception is handled or not. |
| throw | The "throw" keyword is used to throw an exception. |
| throws | The "throws" keyword is used to declare exceptions. It specifies that there may occur an exception in the method. It doesn't throw an exception. It is always used with method signature. |

# Java HashMap



Java **HashMap** class implements the Map interface which allows us to store key and value pair, where keys should be unique. If you try to insert the duplicate key, it will replace the element of the corresponding key. It is easy to perform operations using the key index like updation, deletion, etc. HashMap class is found in the java.util package.

Points to remember

* Java HashMap contains values based on the key.
* Java HashMap contains only unique keys.
* Java HashMap may have one null key and multiple null values.
* Java HashMap is non synchronized.
* Java HashMap maintains no order.
* The initial default capacity of Java HashMap class is 16 with a load factor of 0.75

**package** Programs;

**public** **class** FibonacciExample1 {

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

// **TODO** Auto-generated method stub

{

**int** n1=0,n2=1,n3,i,count=10;

System.***out***.println(n1);

System.***out***.println(n2);

**for**(i=2;i<count;i++)

{

n3=n1+n2;

System.***out***.println(n3);

n1=n2;

n2=n3;

}

}

}

ANS

0

1

1

2

3

5

8

13

21

34

**package** Programs;

**public** **class** FactorialExample2 {

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

// **TODO** Auto-generated method stub

//4! = 4\*3\*2\*1 = 24

//5! = 5\*4\*3\*2\*1 = 120

**int** i,fact=1;

**int** number=5;

**for**(i=1;i<=number;i++)

fact=fact\*i;

{

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

}

}

}

ANS

120

