Java Notes

Anees Rehman Khan

Java is one of the most popular programming languages because it is used in various tech fields like app development, web development, client-server applications, etc.

Java is an object-oriented programming language developed by Sun Microsystems of the USA in 1991.

It was originally called Oak by James Goslin. He was one of the inventors of Java.
Java = **Purely Object-Oriented**.

How **Java** Works -> Source Code(Compiled) -> Byte Code ->(interpreted) Machine Code

Java Installations :

1. Downloading the java development kit JDK
   https://www.oracle.com/java/technologies/downloads/#jdk19-windows

2. JDK->Collections of tools used for developing and running java programs
3. Download Eclipse or Intellege

What is function?

- It is something which is going to get execute the task
- whenever we need we call a function to perform specific task
- void function doesn't return any value

Naming Conventions

Classes we use Pascal- EmployeeDetails
Functions we use camelCase- employeeDetails

Write a Java Program to explain the working flow.

(Javaa01.java)

1. Package : com.arkprocoder

- Packages are used to group the related classes.

2. public class Main :

- In Java, every program must contain a class.
- The filename and name of the class should be the same.
- Here, we've created a class named "Main".
- It is the entry point to the application.

3. public static void main(String[]args)

- This is the main() method of our Java program.
- Every Java program must contain the main() method.

4. System.out.println("Hello World"):

- The above code is used to display the output on the screen.
- Anything passed inside the inverted commas is printed on the screen as plain text.

Variables and Data Types in Java Programming (Javaa02.java)

Variables which stores a values acts like container

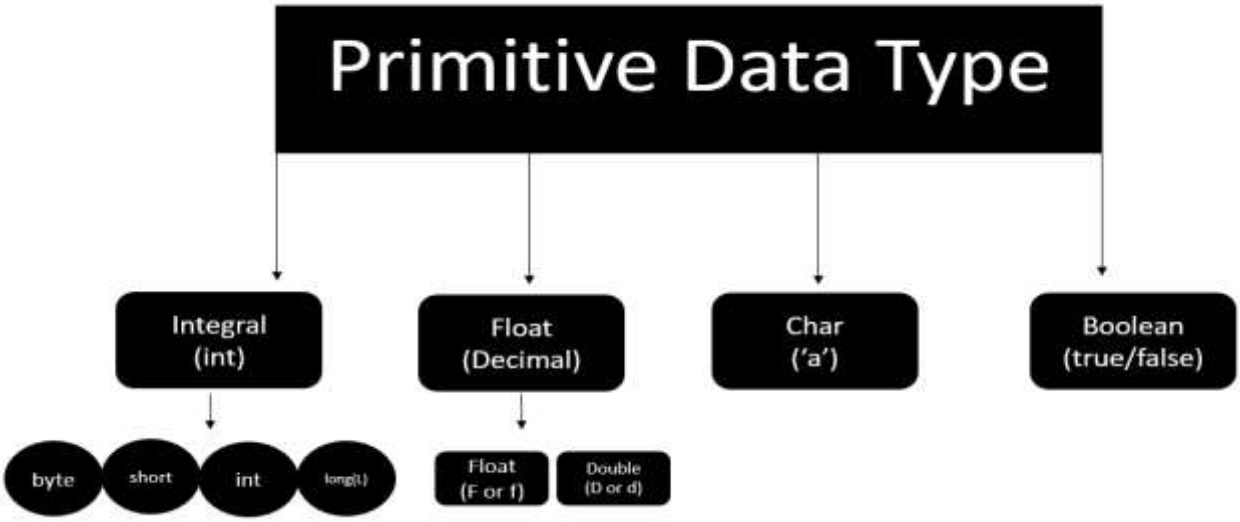
RULES:

- Should not starts with number -1example
- Should not be a keywords like short int void
- Should not starts with special chars #$%@^$^@
- Should not contains white spaces
- Variables names are case sensitive

- Use chars with number valid anee23lku

Data Types:

1. Primitive Data Types
2. Non-Primitive Data Types (Derived)



int(4 bytes), long(8), double(8 0.0d), short(2), byte(1), float(4 0.0f), char(2 (0-65535), boolean(True/False (default-false)

Literals : A constant value that can be assigned to a variable is called a literals.

- 101 – Integer literal
- 10.1f – float literal
- 10.1 – double literal (default type for decimals)
- 'A' – character literal
- true – Boolean literal
- "Harry" – String literal

Keywords: Words that are reserved and used by the Java compiler. They cannot be used as an Identifier.

Ex: short, int, long, Double etc

Taking Input From User: (Javaa03.java)

Scanner class of java.util package is used to take input from the user's keyboard. The Scanner class has many methods for taking input from the user depending upon the type of input. To use any of the methods of the Scanner class, first, we need to create an object of the Scanner class as shown in

Operators in java (Javaa04.java)

* Arithmetic operators -> +,-,*,/,%,++,--

* Assignment operators -> =,+=,-=,*=,/=

* Comparison operators -> ==,>=,<=,!=

* Logical operators -> &&, ||, !

* Bitwise operators -> &, |

* Boolean operators -> true , false

Result of two data types

- b+s=int
- s+i=int
- l+f=float
- i+f=int
- c+i=int
- c+s=int
- l+d=double
- f+d=double
- a++, ++a => auto increment mode
- a--,--a => auto decrement mode

Strings in Java (Javaa05.java)

A string is a sequence of characters

- Strings are immutable and cannot be changed.
- java.lang.String class is used to create a String object.
- The string is a class but can be used as a data type.

In Java, strings can be created in two ways :

1. By using String Literal (String s="Anees")
2. By using new (String s=new String("Anees"))

Note:

- We use double quotes("") to create string using string literal. Before creating a new string instance, JVM verifies if the same string is already present in the string pool or not. If it is already present, then JVM returns a reference to the pooled instance otherwise, a new string instance is created.

- When we create a string using "new", a new object is always created in the heap memory

Different ways to print in Java :

We can use the following ways to print in Java:

- System.out.print() // No newline at the end
- System.out.println() // Prints a new line at the end
- System.out.printf()
- System.out.format()
- %d for int
- %f for float

- %c for char
- %s for string

String Methods : (Javaa06.java)

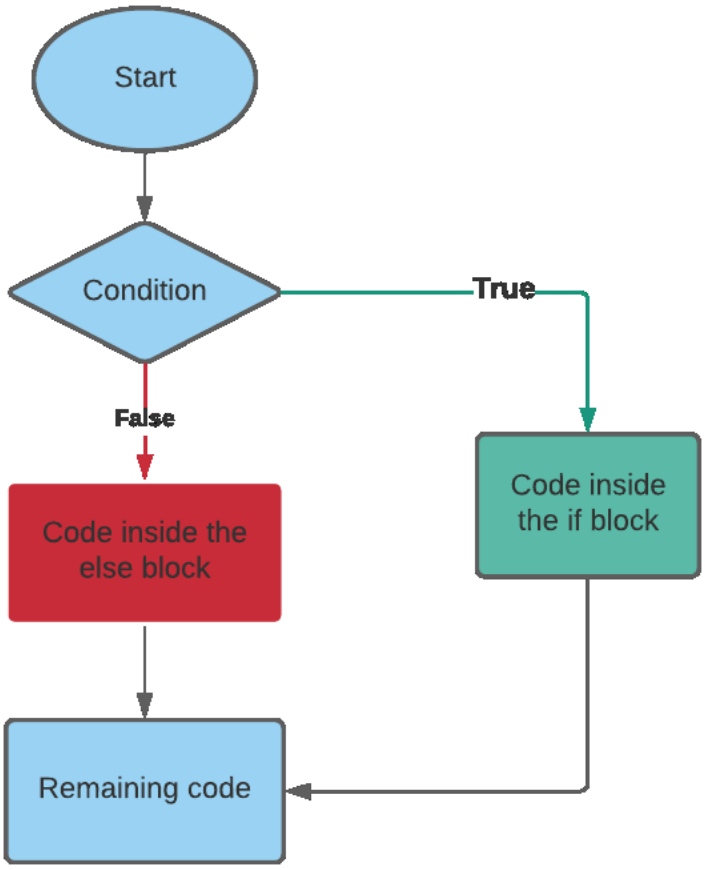
- length() : Returns the length of String name.
- toLowerCase(): Converts all the characters of the string to the lower case letters.
- toUpperCase(): Converts all the characters of the string to the upper case letters.
- trim(): Returns a new String after removing all the leading and trailing spaces from the original string.
- substring(int start): Returns a substring from start to the end. Substring(3) returns [Note that indexing starts from 0]
- substring(int start, int end): Returns a substring from the start index to the end index. The start index is included, and the end is excluded.
- replace('Ark', "Anees"): Returns a new string after replacing Ark with Anees.
- startsWith("A"): Returns true if the name starts with the string "A".
- endsWith("r"): Returns true if the name ends with the string "r".
- charAt(2): Returns the character at a given index position.
- indexOf("s") : Returns the index of the first occurrence of the specified character in the given string.
- lastIndexOf("r"): Returns the last index of the specified character from the given string
- equals("Ark Pro Coder"):Returns true if the given string is equal to "Ark Pro Coder" false otherwise [Case sensitive]

- equalsIgnoreCase("ark pro Coder"): Returns true if two strings are equal, ignoring the case of characters.

Escape Sequence Characters :

- The sequence of characters after backslash '\' = Escape Sequence Characters
- Escape Sequence Characters consist of more than one character but represent one character when used within the strings.
- Examples: \n (newline), \t (tab), \' (single quote), \\ (backslash), etc.

If-else Statement in Java (Javaa07.java)



Relational Operators in Java :

Relational operators are used to evaluate conditions (true or false) inside the if statements. Some examples of relational operators are:

== (equals)

>= (greater than or equals to)

> (greater than)

< (less than)

<= (less than or equals to)

!= (not equals)

Note: '=' is used for an assignment whereas '==' is used for equality check. The condition can be either true or false.

Logical Operators :

Logical operators are used to provide logic to our Java programs.

There are three types of logical operators in Java :

&& - AND

|| - OR

! – NOT

AND Operator :

Evaluates to true if both the conditions are true.

Y && Y = Y

Y && N = N

N && Y = N

N && N = N

Convention: # Y – True and N - False

OR Operator :

Evaluates to true when at least one of the conditions is true.

Y || Y = Y

Y || N = Y

N || Y = Y

N || N = N

Convention: Y – True and N - False

NOT Operator :

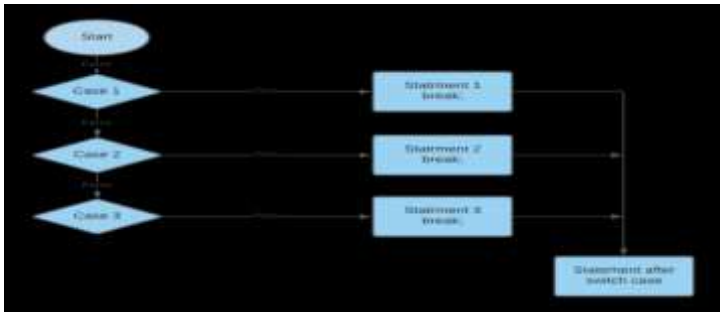
Negates the given logic (true becomes false and vice-versa)

!Y = N

!N = Y

Switch Case in Java (Javaa07.java)

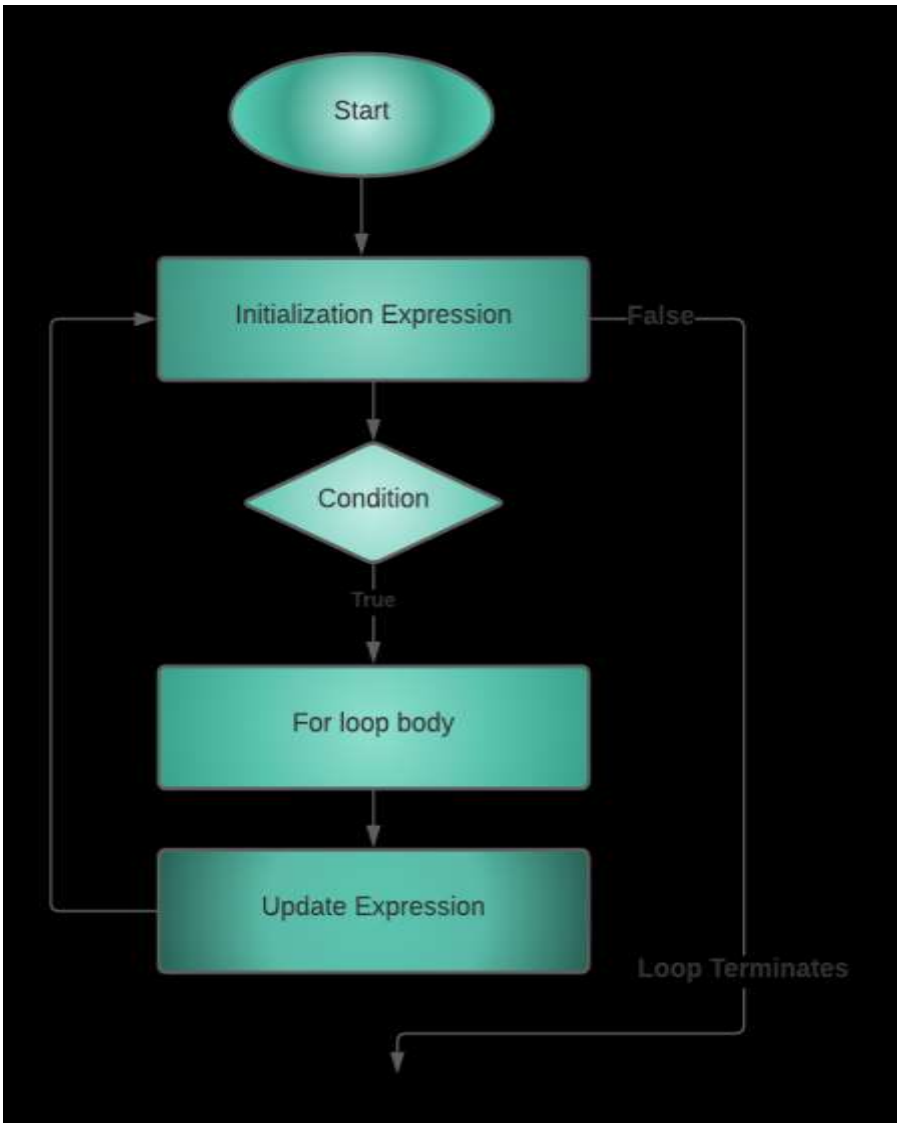
- Switch-Case is used when we have to make a choice between the number of alternatives for a given variable.
- Variable can be an integer, character, or string in Java.
- Every switch case must contain a default case. The default case is executed when all the other cases are false.
- Never forget to include the break statement after every switch case otherwise the switch case will not terminate



For Loops & While Loops (Javaa08.java)

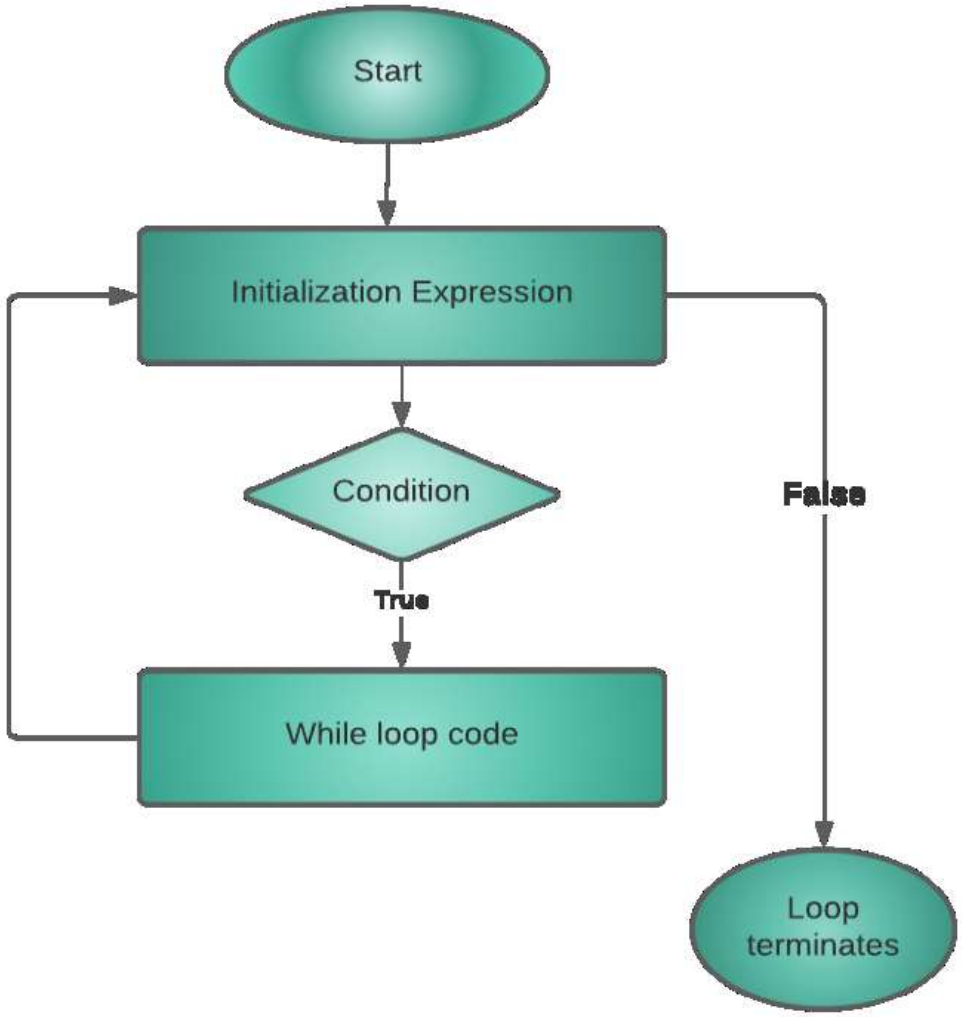
For loop:

- For loop in java is used to iterate a block of code multiple times.
- Use for loop only when the exact number of iterations needed is already known to you.
- Initializer: Initializes the value of a variable. This part is executed only once.
- Check Boolean expression: The code inside the for loop is executed only when this condition returns true.
- update: Updates the value of the initial variable.



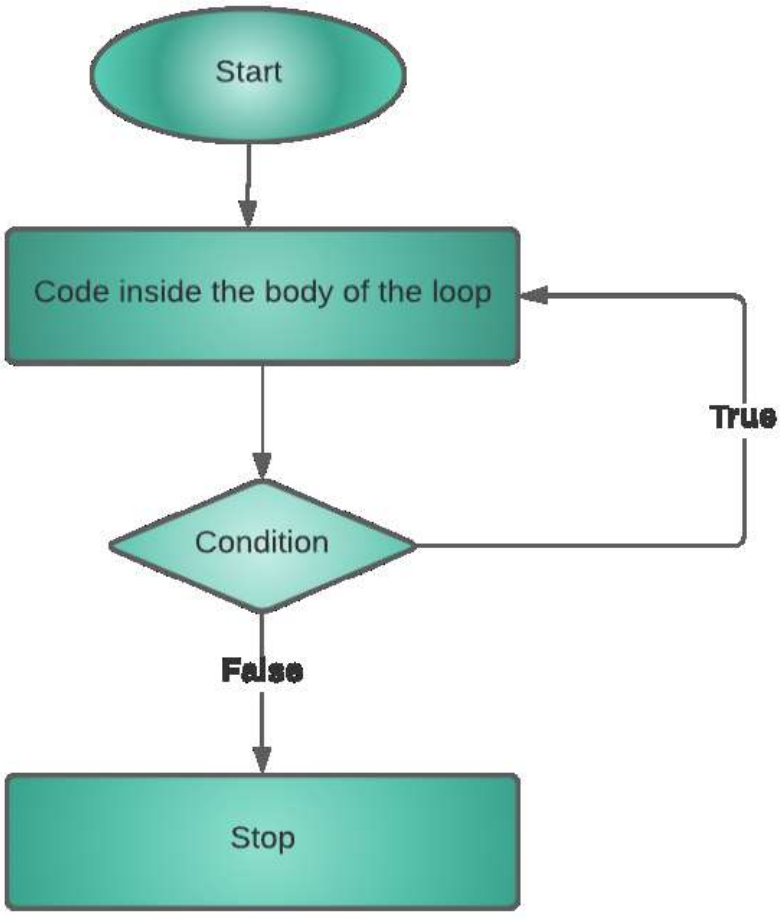
While Loop

- The while loop in Java is used when we need to execute a block of code again and again based on a given Boolean condition.
- Use a while loop if the exact number of iterations is not known.
- If the condition never becomes false, the while loop keeps getting executed. Such a loop is known as an infinite loop.



Do-While Loop

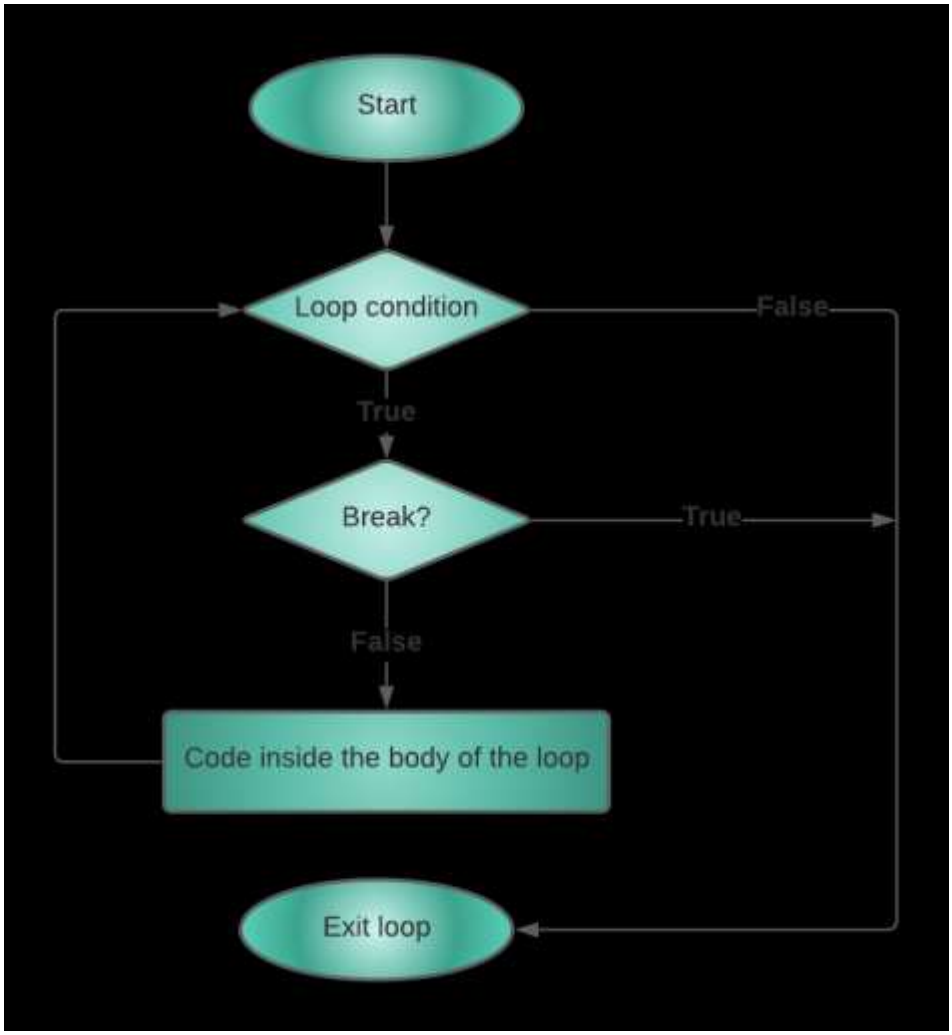
- Do- while loop is similar to a while loop except for the fact that it is guaranteed to execute at least once.
- Use a do-while loop when the exact number of iterations is unknown, but you need to execute a code block at least once.
- After executing a part of a program for once, the rest of the code gets executed on the basis of a given Boolean condition.



Break Statement :

The break statement is used to exit the loop irrespective of whether the condition is true or false.

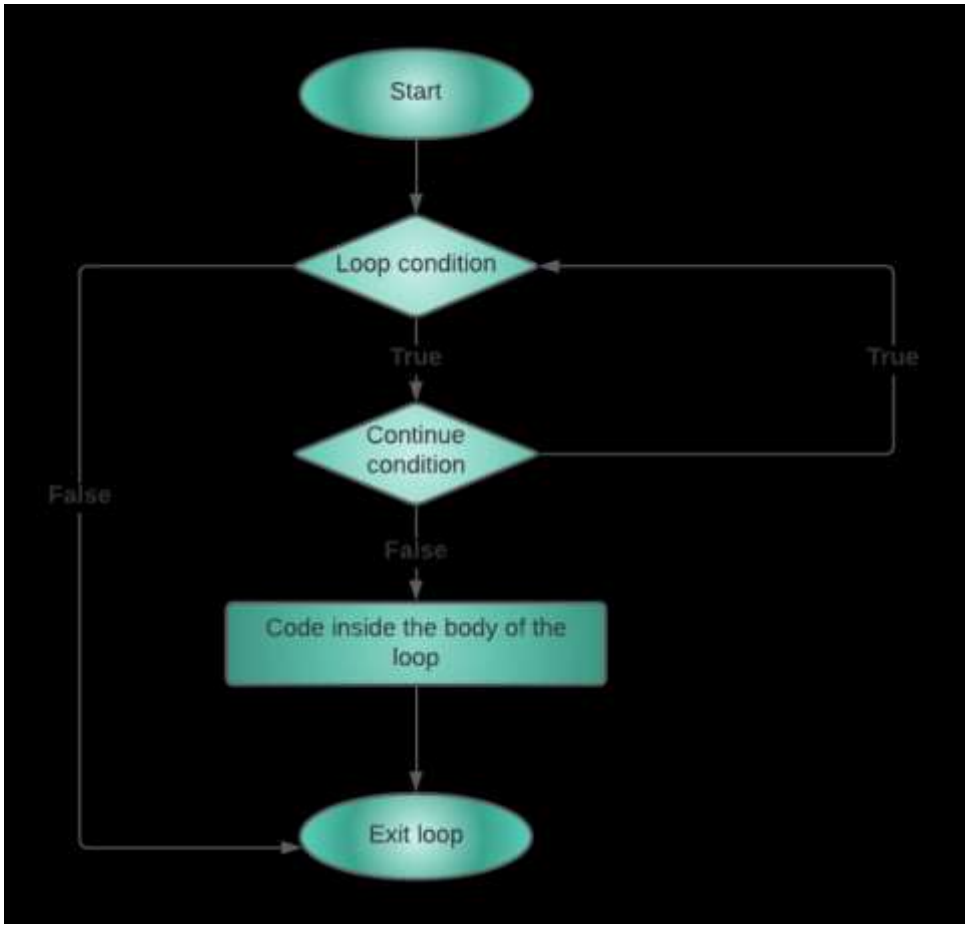
Whenever a 'break' is encountered inside the loop, the control is sent outside the loop



Continue Statement :

The continue statement is used to immediately move to the next iteration of the loop.

The control is taken to the next iteration thus skipping everything below 'continue' inside the loop for that iteration.



Arrays in Java (Javaa09.java)

- An array is a collection of similar types of data having contiguous memory allocation.
- The indexing of the array starts from 0 i.e 1st element will be stored at the 0th index, 2nd element at 1st index, 3rd at 2nd index, and so on.
- The size of the array cannot be increased at run time therefore we can store only a fixed size of elements in array.
- int[] marks; //Declaration!
- marks = new int[5]; //Memory allocation!
- int[] marks = new int[5]; //Declaration + Memory allocation!
- int[] marks = {100,70,80,71,98} // Declare + Initialize!

For Each Loop in Java

- For each loop is an enhanced version of for loop.
- It travels each element of the data structure one by one.
- Note that you cannot skip any element in for loop and it is also not possible to traverse elements in reverse order with the help of for each loop.
- It increases the readability of the code.
- If you just want to simply traverse an array from start to end then it is recommended to use for each loop.

Multidimensional Arrays in Java

Multidimensional Arrays are an Array of Arrays. Each elements of an M-D array is an array itself.

	[0]	[1]	[2]
	Col 1	Col 2	Col 3
[0] Row 1	(0,0)	(0,1)	(0,2)
[1] Row 2	(1,0)	(1,1)	(1,2)

Java Methods : (Javaa10.java)

A method is a function written inside a class. Since Java is an object-oriented language, we need to write the method inside some class.

Sometimes our program grows in size, and we want to separate the logic of the main method from the other methods.

For instance, if we calculate the average of a number pair 5 times, we can use methods to avoid repeating the logic. [DRY – Don't Repeat Yourself.

Calling a Method :

A method can be called by creating an object of the class in which the method exists followed by the method call

Void return type :

When we don't want our method to return anything, we use void as the return type.

Static keyword :

The static keyword is used to associate a method of a given class with the class rather than the object.

You can call a static method without creating an instance of the class.

In Java, the main() method is static, so that JVM can call the main() method directly without allocating any extra memory for object creation.

All the objects share the static method in a class.

Method Overloading (Javaa11.java)

In Java, it is possible for a class to contain two or more methods with the same name but with different parameters. Such methods are called Overloaded methods.

Method overloading is used to increase the readability of the program

Ways to perform method overloading :

In Java, method overloading can be performed by two ways listed below :

- By changing the return type of the different methods

  By changing the number of arguments accepted by the method Now, let's have an example to understand the above ways of method overloading

- By changing the return type :
  - These two methods are overloaded because they have the same name but their return is different.
  - The return type of 1st method is int while the return type of the other method is double

- By changing the number of arguments passed :
  - Methods with different arguments passed
  - Methods with different return types

Java Recursion : **(Javaa12.java)**

- In programming, recursion is a technique through which a function calls itself.
- With the help of recursion, we can break down complex problems into simple problems.

Java Object Oriented Programming

Object-Oriented Programming tries to map code instructions with real-world, making the code short and easier to understand.

With the help of OOPs, we try to implement real-world entities such as object, inheritance, abstraction, etc.

OOPs helps us to follow the DRY(Don't Repeat Yourself) approach of programming, which in turn increases the reusability of the code.

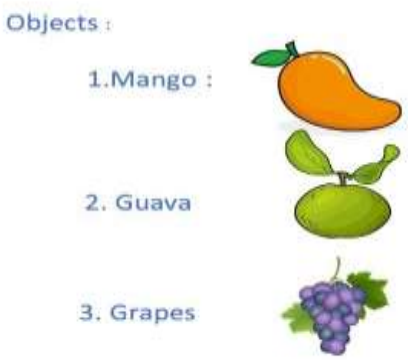
Two most important aspects of OOPs

Class : (Javaa13.java)

- A class is a blueprint for creating objects.
- Classes do not consume any space in the memory.
- Objects inherit methods and variables from the class.
- It is a logical component.

Object:

- An object is an instantiation of a class. When a class is defined, a template (info) is defined.
- Every object has some address, and it occupies some space in the memory.
- It is a physical entity.



Basic Terminologies in Object Oriented Programming

1. **Abstraction** :

Data abstraction is the way through which only the essential info is shown to the user, and all the internal details remain hidden from the user.

2. **Polymorphism** :

One entity many forms.

- The word polymorphism comprises two words, poly which means many, and morph, which means forms.
- In OOPs, polymorphism is the property that helps to perform a single task in different ways.
- Let us consider a real-life example of polymorphism. A woman at the same time can be a mother, wife, sister, daughter, etc. Here, a woman is an entity having different forms.
- A Smartphone can work like a camera as well as like a calculator. So, you can see the a smartphone is an entity having different forms.

3. **Encapsulation** :

The act of putting various components together (in a capsule).

- In java, the variables and methods are the components that are wrapped inside a single unit named class.
- All the methods and variables of a class remain hidden from any other class.
- A automatic cold drink vending machine is an example of encapsulation.
- Cold drinks inside the machine are data that is wrapped inside a single unit cold drink vending machine.

4. **Inheritance** :

- The act of deriving new things from existing things.
- In Java, one class can acquire all the properties and behaviours of other some other class
- The class which inherits some other class is known as child class or sub class.
- The class which is inherited is known as parent class or super class.
- Inheritance helps us to write more efficient code because it increases the reusability of the code.

Access modifiers, getters & setters in Java (Javaa14.java)

Access Modifiers specify where a property/method is accessible. There are four types of access modifiers in Java :

1. private
2. default
3. protected
4. public

Access Modifier	within class	within package	outside package by subclass only	outside package
public	Y	Y	Y	Y
protected	Y	Y	Y	N
Default	Y	Y	N	N
private	Y	N	N	N

Getters and Setters :

Getter : Returns the value [Accessors]

Setter : Sets / updates the value [Mutators]

Constructors in Java (Javaa15.java)

Constructors are similar to methods,, but they are used to initialize an object.

Constructors do not have any return type(not even void).

Every time we create an object by using the new() keyword, a constructor is called.

If we do not create a constructor by ourselves, then the default constructor(created by Java compiler) is called.

Rules for creating a Constructor :

- The class name and constructor name should be the same.
- It must have no explicit return type.
- It cannot be abstract, static, final, and synchronized.

Types of Constructors in Java :

There are two types of constructors in Java :

1. Defaut constructor : A constructor with 0 parameters is known as default constructor.

2. Paramerterized constructor **:** A constructor with some specified number of parameters is known as a parameterized constructor

Constructor Overloading in Java with Access Modifiers :

Just like methods, constructors can also be overloaded in Java. We can overload the constructor

Note:

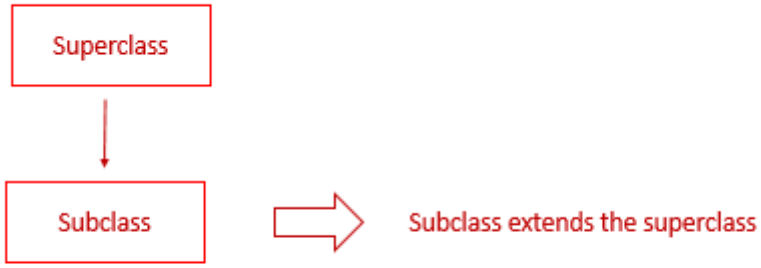
- Constructors can take parameters without being overloaded
- There can be more than two overloaded constructors

Inheritance in Java (Javaa16.java)

You might have heard people saying your nose is similar to your father or mother. Or, more formally, we can say that you've inherited the genes from your parents due to which you look similar to them.

The same phenomenon of inheritance is also valid in programming.

- In Java, one class can easily inherit the attributes and methods from some other class. This mechanism of acquiring objects and properties from some other class is known as inheritance in Java.
- Inheritance is used to borrow properties & methods from an existing class.
- Inheritance helps us create classes based on existing classes, which increases the code's reusability.
- The **Extends** keyword is used to inherit a subclass from a superclass.



Constructors in Inheritance in Java

When a derived class is extended from the base class, the constructor of the base class is executed first followed by the constructor of the derived class. For the following Inheritance hierarchy, the constructors are executed in the order:

- Parent
- Child
- Grandchild

Constructors during constructor overloading:

When there are multiple constructors in the parent class, the constructor without any parameters is called from the child class.

If we want to call the constructor with parameters from the parent class, we can use the **super** keyword.

super(a, b) calls the constructor from the parent class which takes 2 variables

this in Java:

- this is a way for us to reference an object of the class which is being created/referenced.
- It is used to call the default constructor of the same class
- this keyword eliminates the confusion between the parameters and the class attributes with the same name

Super in Java:

- A reference variable used to refer immediate parent class object.
- It can be used to refer immediate parent class instance variable.
- It can be used to invoke the parent class method.

Method Overriding in Java (Javaa17.java)

- If the child class implements the same method present in the parent class again, it is known as method overriding.
- Method overriding helps us to classify a behaviour that is specific to the child class.
- The subclass can override the method of the parent class only when the method is not declared as final.

Dynamic Method Dispatch in Java

- Dynamic method dispatch is also known as run time polymorphism.
- It is the process through which a call to an overridden method is resolved at runtime.
- This technique is used to resolve a call to an overridden method at runtime rather than compile time.

To properly understand Dynamic method dispatch in Java, it is important to understand the concept of up-casting because dynamic method dispatch is based on up-casting.

UpCasting :

It is a technique in which a superclass reference variable refers to the object of the subclass.

Abstract Class & Abstract Methods (Javaa18.java)

What does Abstract mean?

Abstract in English means existing in through or as an idea without concrete existence.

Abstract class :

- An abstract class cannot be instantiated.
- Java requires us to extend it if we want to access it.
- It can include abstract and non-abstract methods.
- If a class includes abstract methods, then the class itself must be declared abstract
- Abstract class are used when we want to achieve security & abstraction(hide certain details & show only necessary details to the user

Abstract method :

- A method that is declared without implementation is known as the abstract method.
- An abstract method can only be used inside an abstract class.
- The body of the abstract method is provided by the class that inherits the abstract class in which the abstract method is present.

Interfaces in Java (Javaa19.java)

- Just like a class in java is a collection of the related methods, an interface in java is a collection of abstract methods.
- The interface is one more way to **achieve abstraction** in Java.
- An interface may also contain constants, default methods, and static methods.
- All the methods inside an interface must have empty bodies except default methods and static methods.
- We use the interface keyword to declare an interface.
- There is no need to write abstract keyword before declaring methods in an interface because an interface is implicitly abstract.
- An interface **cannot** contain a **constructor** (as it cannot be used to create objects)

- In order to implement an interface, java requires a class to use the **implement** keyword.

Default methods In Java:

- An interface can have static and default methods.
- Default methods enable us to add new functionality to existing interfaces.
- This feature was introduced in java 8 to ensure backward compatibility while updating an interface.
- A class implementing the interface need not implement the default methods.
- Interfaces can also include private methods for default methods to use.
- You can easily override a default method like any other method of an interface.

Abstract class	Interface
1. It can contain abstract and non-abstract method	1. It can only contain abstract methods. We do not need to use the "abstract" keyword in interface methods because the interface is implicitly abstract.
2. abstract keyword is used to declare an abstract class.	2. Interface keyword is used to declare an interface.
3. A sub-class extends the abstract class by using the "extends" keyword.	3. The "implements" keyword is used to implement an interface.
4. Abstract class in Java can have class members like private, protected, etc.	4. Members of a Java interface are public by default.
5. Abstract class doesn't support multiple inheritance.	5. Multiple inheritances is achieved in Java by using the interface.

Why multiple inheritance is not supported in java?

Multiple inheritance faces problems when there exists a method with the same signature in both the super classes.

Due to such a problem, java does not support multiple inheritance directly, but the similar concept can be achieved using interfaces.

A class can implement multiple interfaces and extend a class at the same time.

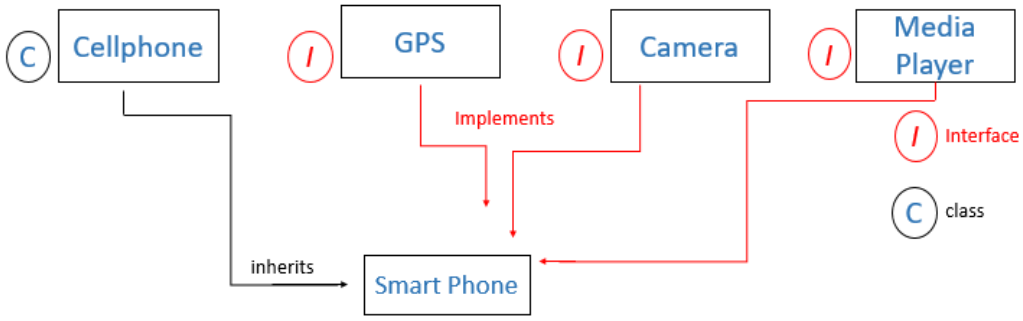
Note :

- Interfaces in java are a bit like the class but with a significantly different.
- An Interface can only have method signatures field and a default method.
- The class implementing an interface needs to declare the methods (not field)
- You can create a reference of an interface but not the object
- Interface methods are public by default.

Inheritance in Interfaces (Javaa20.java)

We have inheritance in interfaces as well refer javaa20 program

Polymorphism in Interfaces: One entity many forms. (Javaa21.java)



Interpreted V/S Compiled Languages

The interpreter translates one statement at a time into machine code. On the other hand, the compiler scans the entire program and translates the whole of it into machine code

Interpreter :

- one statement at a time
- An interpreter is needed every time
- Partial execution if an error occurs in the program.
- Easy for programmers.

Compiler :

- Entire program at a time
- Once compiled, it is not needed
- No execution if an error occurs
- Usually not as easy as interpreted once

Is Java interpreted or compiled?

- Java is a hybrid language both compiled as well as interpreted.
- A JVM can be used to interpret this byte code
- This byte code can be taken to any platform (win/ mac / Linux) for education.
- Hence java is platform-independent (write once run everywhere).

Packages in Java

- A package is used to group related classes.
- packages help in avoiding name conflicts

There are two types of packages :

- Build-in packages - java API
- User-defined packages - Custom packages

Using a java package :

Import keyword is used to import packages in the java program.
Example :

- import java lang * - import
- import java string - import string from java long

import java.util.* : it imports all the packages from java.

Create package in java : javac –d arkprocoder java -> this will create a arkprocoder packages folder

Multithreading in Java (Javaa22.java)

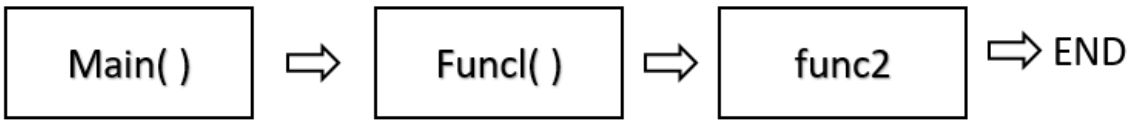
Multiprocessing and multithreading both are used to achieve multitasking.

- threads use shared memory area
- threads faster content switching
- A thread is light-weight where a process is heavyweight

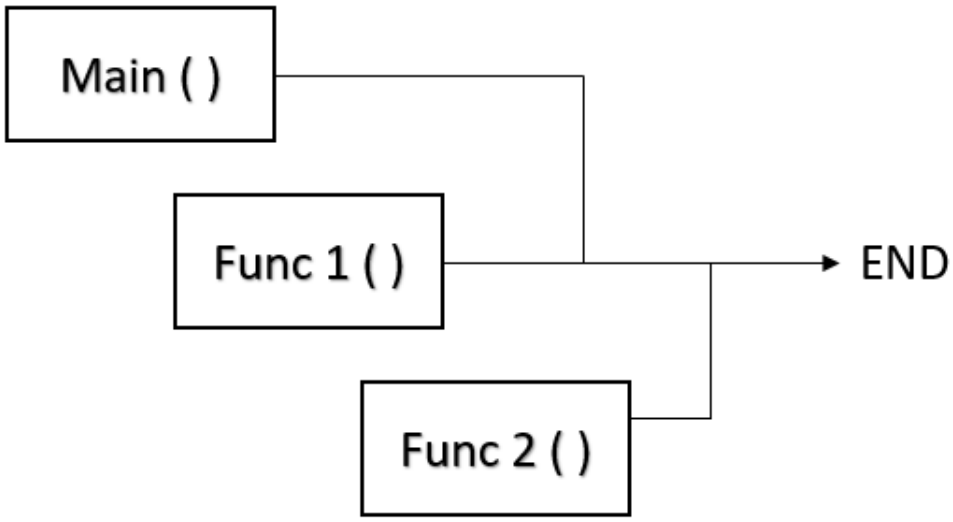
Example = A word processor can have one thread running in foreground as an editor and another in the background auto saving the document.

Flow of control in java

Without threading:



With Threading :



Creating a Threading

There are two ways to create a thread in java

1. By extending thread class
2. By implementing Runnable interface

In order to execute the thread, the start() method is used. start() is called on the object of the MyThread class. It automatically calls the run() method, and a new stack is provided to the thread. So, that's how you easily create threads by extending the thread class in Java.

Creating a Java Thread Using Runnable Interface

Steps To Create A Java Thread Using Runnable Interface:

- Create a class and implement the Runnable interface by using the implements keyword.
- Override the run() method inside the implementer class.
- Create an object of the implementer class in the main() method.

- Instantiate the Thread class and pass the object to the Thread constructor.
- Call start() on the thread. start()will call the run()method.

See Javaa23.java file

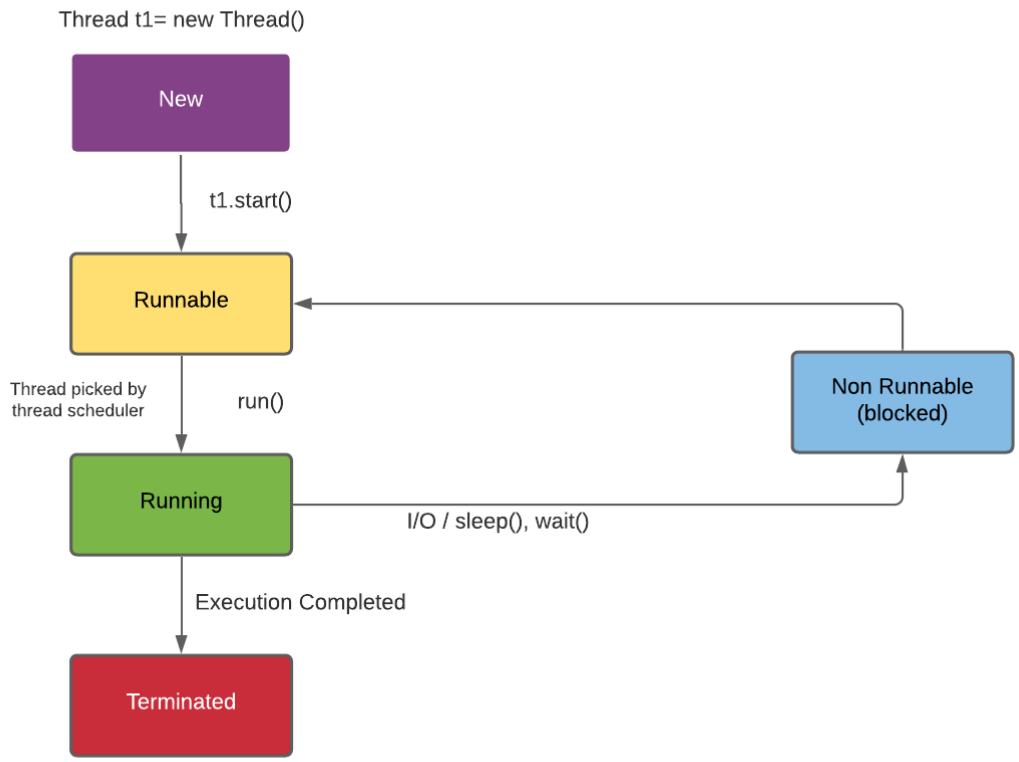
1. class t1 is implementing the Runnable interface.
2. Overriding of the run() method is done inside the t1 class.
3. In the main() method, obj1, an object of the t1 class, is created.
4. The constructor of the Thread class accepts the Runnable instance as an argument, so obj1 is passed to the constructor of the Thread class.
5. Finally, the start()method is called on the thread that will call the run() method internally, and the thread's execution will begin.

Runnable Interface V/s Extending Thread Class :

- Since we've discussed both the ways to create a thread in Java. There might be a question in your mind that should we use the Runnable interface or Thread class to implement a thread in Java. Let me answer this question for you. The Runnable interface is preferred over extending the Thread class because of the following reasons :
- As multiple inheritance is not supported in Java, it is impossible to extend the Thread class if your class had already extended some other class.
- While implementing Runnable, we do not modify or change the thread's behaviour.

- More memory is required while extending the Thread class because each thread creates a unique object.
- Less memory is required while implementing Runnable because multiple threads share the same object.

Java Thread Life Cycle



1. **New** - Instance of thread created which is not yet started by involving start(). In this state, the thread is also known as the born thread.
2. **Runnable** - After invocation of start() & before it is selected to be run by the scheduler.
3. **Running** - After the thread scheduler has selected it.
4. **Non-runnable** - thread alive, not eligible to run.
5. **Terminated** - run() method has exited.

Constructors from Thread class in Java (Javaa24.java)

The Thread class

- Thread ( )
- Thread (string)
- Thread (Runnable r)
- Thread (Runnable r, String name)

Java Thread Priorities

In a Multithreading environment, all the threads which are ready and waiting to be executed are present in the Ready queue. The thread scheduler is responsible for assigning the processor to a thread. But the question is on what basis the thread scheduler decides that a particular thread will run before other threads. Here comes the concept of priority in the picture.

- Every single thread created in Java has some priority associated with it.
- The programmer can explicitly assign the priority to the thread. Otherwise, JVM automatically assigns priority while creating the thread.
- In Java, we can specify the priority of each thread relative to other threads. Those threads having higher priorities get greater access to the available resources than lower priorities threads.
- Thread scheduler will use priorities while allocating processor.
- The valid range of thread priorities is 1 to 10 (but not 0 to 10), where 1 is the least priority, and 10 is the higher priority.

- If there is more than one thread of the same priority in the queue, then the thread scheduler picks any one of them to execute.
- The following static final integer constants are defined in the Thread class:
- MIN_PRIORITY: Minimum priority that a thread can have. Value is 1.
- NORM_PRIORITY: This is the default priority automatically assigned by JVM to a thread if a programmer does not explicitly set the priority of that thread. Value is 5.
- MAX_PRIORITY: Maximum priority that a thread can have. Value is 10.

Java Thread Methods (Javaa25.java)

Join() method In Java :

- The join () method in Java allows one thread to wait until the execution of some other specified thread is completed.
- If t is a Thread object whose thread is currently executing, then t.join () causes the current thread to pause execution until t's thread terminates.
- Join () method puts the current thread on wait until the thread on which it is called is dead.

Sleep () Method :

- The sleep() method in Java is useful to put a thread to sleep for a specified amount of time.

- When we put a thread to sleep, the thread scheduler picks and executes another thread in the queue.
- Sleep () method returns void.
- Sleep () method can be used for any thread, including the main() thread also.

Syntax :

- ❖ public static void sleep(long milliseconds)throws Interrupted Exception
- ❖ public static void sleep(long milliseconds, int nanos)throws Interrupted Exception
- ❖ Parameters Passed To Sleep() Method :
- ❖ long millisecond: Time in milliseconds for which thread will sleep.
- ❖ nanos : Ranges from [0,999999]. Additional time in nanoseconds.

Interrupt () method :

A thread in a sleeping or waiting state can be interrupted by another thread with the help of the interrupt () method of the Thread class.

- The interrupt() method throws Interrupted Exception.
- The interrupt() method will not throw the Interrupted Exception if the thread is not in the sleeping/blocked state, but the interrupt flag will be changed to true.

Errors & Exception in Java

There are three types of errors in java.

1) Syntax errors

2) Logical errors

3) Runtime errors- also called Exceptions

Logical errors

- A logical error or a bug occurs when a program
- compiles and round but does the wrong thing.
    - Message delivered wrongly
    - Wrong time of chats being displayed
- Incorrect redirects!

Runtime errors

- Java may sometimes encounter an error while the program is running.
- These are also called Exceptions!
- These are encountered due to circumstances like bad input and (or) resource constraints.
- Ex: User supplies 'S' + 8 to a program that adds 2 numbers.

Syntax errors and logical errors are encountered by the programmers, whereas Run-time errors are encountered by the users.

Exceptions & Try-Catch Block in Java (Javaa26.java)

Exceptions in Java

An exception is an event that occurs when a program is executed the normal flow of instructions.

There are mainly two types of exceptions in java:

1) **Checked exceptions -** compile-time exceptions (Handle by the compiler)

2) **Unchecked exceptions** - Runtime exceptions

Commonly Occurring Exceptions

Following are few commonly occurring exceptions in java:

1) Null pointer exception

2) Arithmetic Exception

3) Array Index out of Bound exception

4) Illegal Argument Exception

5) Number Format Exception

Handling Specific Exceptions in Java (Javaa27.java)

Handling nested try/catch (Javaa28.java)

Throw v/s Throws: (Javaa29.java)

The Throw Keyword:

The throw keyword is used to throw an exception explicitly by the programmer

The throw Keyword

Throws java throws keyword is used to declare an exception.

This gives an information to the programmer that there might be an exception so it is better to be prepared with a try catch block!

Finally Block in Java

- Java finally block
- Finally block contains the code which is always executed whether the exception is handled or not.
- It is used to exception is handled or not.
- It is used to execute code containing instructions to release the system resources, close a connection etc.

Java Collections Framework

Collection Framework

A collection represents a group of object Java collections provide classes and Interfaces for us to be able to write code interfaces for us to be able to write code quickly and efficiently

Why do we need Collection :

We need collections for efficient storage and better manipulation of data in java

For ex: we use arrays to store integers but what if we want to

- Resize this array?
- Insert an element in between?
- Delete an element in Array?
- Apply certain operations to change this array?

Collections Hierarchy in Java

Collections in java are available as class and interfaces are few commonly used collections in java :

ArrayList -> For variables size collections

Set -> For distinct collection

Stack-> A LIFO data structure

HashMap -> For strong key - value pairs

Collections class is available in java util package collection class also provides static methods for sorting , searching etc.

1. ArrayList in Java Methods (Javaa30.java)

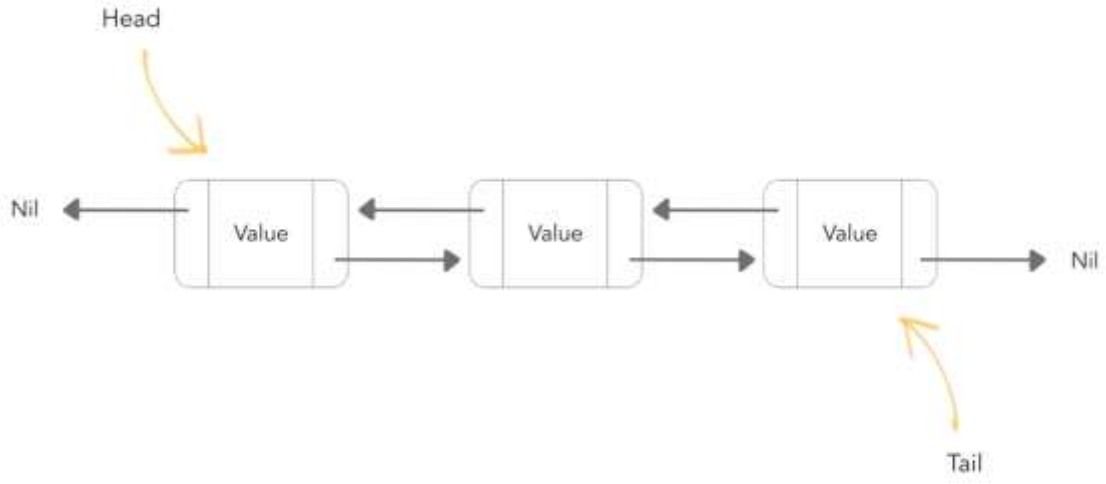
- The ArrayList class is the dynamic array found in the java.util package.
- The size of the normal array cannot be changed, but ArrayList provides us the ability to increase or decrease the size.
- ArrayList is slower than the standard array, but it helps us to manipulate the data easily.
- Look out **(Javaa30.java)** example for more details

2. LinkedList in Java (Javaa31.java)

The LinkedList class in Java provides us with the doubly linked list data structure.

- Each element of the linked list is known as a node.

- Each node points to the address of the next node & its previous node.



1. Linked lists are preferred over the Array list because the insertion & deletion in the linked lists can be done in a constant time. But, in arrays, if we want to add or delete an element in between then, we need to shift all the other elements.
2. In a linked list, it is impossible to directly access an element because we need to traverse the whole linked list to get the desired element.

ArrayList Vs. LinkedList :

Although **ArrayList** & **LinkedList** both implement the List interface and have the same methods, it is important to understand when to use which one.

The insertion & deletion can be done in constant time in Linked List, so it is best to **use** the **linked list when you need to add or remove elements frequently.**

Use ArrayList when you want to access the random elements frequently, as it can't be done in a linked list in constant time.

3. ArrayDeque in Java (Javaa32.java)

1. ArrayDeque = Resizable array + Deque interface.
2. ArrayDeque implements the Queue & Deque interface.

There are no capacity restrictions for ArrayDeque, and it provides us the facility to add or remove any element from both sides of the queue.

Also known as Array Double Ended Queue.

It is **faster** than Linked list and stack.

Constructors of ArrayDeque class :

- ArrayDeque(): Used to create an empty array deque that has the capacity to hold 16 elements.
- ArrayDeque(int numElements): Used to create an empty array deque that has the capacity to hold the specified number of elements.
- ArrayDeque(Collection<? extends E> c): Used to create an array deque containing all the elements of the specified collections.

4. HashSet in Java

HashSet class uses a hash table for storing the elements.

- It implements the set interface.
- Duplicate values are not allowed.
- Before storing any object, the hashset uses the hashCode() and equals() method to check any duplicate entry in the hash table.
- Allows null value.
- Best suited for search operations.

Constructors Of HashSet : (Javaa33.java)

- **HashSet():** This constructor is used to create a new empty HashSet that can store 16 elements and have a load factor of 0.75.
- **HashSet(int initialCapacity):** This constructor is used to create a new empty HashSet which has the capacity to store the specified number of elements and having a load factor of 0.75.
- **HashSet(int initialCapacity, float loadFactor):** This constructor is used to create a new empty HashSet with the capacity & load factor equal to specified integer and float value.
- **HashSet(Collection<? extends E> c):** This constructor is used to create a HashSet using the elements of collection c.

Date ,Time, calender in Java (Javaa34.java)

java time -> package for date & time in java from java onwards

Before java 8, java util package used to hold the date time class now these classes are deprecated

How java stores a Date?

- Date in java is stored in the form of a long numer. This long number holds the number of miliseconds passed since 1 jan 1970
- Java assumes that 1900 is the start year which means it calculates years passed since 1900 whenever We ask it for years passed
- System current Time Millis () returns no of sound passed Once no. of ms are calculated, we can calculate minutes, seconds & years passed

Calendar Class in Java

- The calendar class in java provides the methods that helps in converting date between a specific instant in time.
- It is an abstract class.
- Since it is an abstract class, we can not create an instance of this class with the help of a constructor.
- We use the static method Calender.getInstance() in order to implement a sub-class.