JAVA

Object oriented programming language

Enterprise application

Function: Set of code which perform particular task

Constructor:

* Should have same name as class name
* Every class has a constructor
* Class name along with parenthesis
* By default any class will be supplied with zero arg constructor

Access modifiers: 4 types

Define visibility of class

* Private
* Public
* Protected
* Default

**OOP concept**

* Encapsulation: Wrapping up of data, hiding data from outside world
* Inheritance: inheriting the class of parenthood
* Polymorphism: To behave differently in different situation

Two types

* + Overload: same method different argument
  + Override: occurs when child class override parentclass
* Abstraction: Displaying only necessary contents and hiding all unnecessary contents

Object Creation:

3 steps:

* Declaration
* Instantiation
* Initiation

Switch

Default statement not necessary in Switch

Value of case should be constant

Loop

For, While, do while, if, if-else

Array:

* Defined using[]
* Can hold elements of same data
* Give the size of array while declaring
* Arrays work out w.r.t index
* Index starts from zero
* Array.length give exact size of array

Casting:

* Upcasting
* Downcasting

Static:

* global class variable
* Not instance variable
* Variables class methods blocks--- static can be applied on these

THIS OPERATOR:

* Reference variable which points to current object
* Used on instance variable
* Passed as argument with constructor

SUPER:

* Reference variable
* Point out superclass or point class object
* Can be used with methods, constructor and variable

Count:

Static, can be used globally

Abstraction:

* Hiding the implementation to the outside world
* Keyword: Abstract
* Class should have atleast one method which is abstract
* May or maynot contain abstract method
* It can have abstract method or non abstract method
* To use an abstract, inherit it from subclass

Interfaces : 100% abstract

**Exceptional handling:** handle exception using alternate source

It maintains the normal flow of program

Exception->run-time error>>unexpected, unwanted situation occurred at run time

Types:

1. FileNotFoundException
2. ArithmeticException
3. NumberFormatException
4. NullPointException
5. ClassCastException
6. ArrayIndexOutOfBoundException

MechaNism to work with Exception:

>>Try >>catch >>throw >>throws >>finally

Class A{

Main(){

Int a=10,b=0,c;

Try{

C= a/b;

s.o.p(c);

}

Catch(Exception e)

{

s.o.p()

}}

**File I/O:**

Class in java

Represent i/p source and o/p destination

Streams in java.io supports many data like primitives, object, localised characters, etc

Stream:

Sequence of data or flow of data/series of data

2 types:

* Byte Stream
* Character stream

2 type:

* InPutStream: read data from source
* OutPutStream: write data to destination.

Source>> Program>> Destination::: pulling the data from one file to storing the data to another file

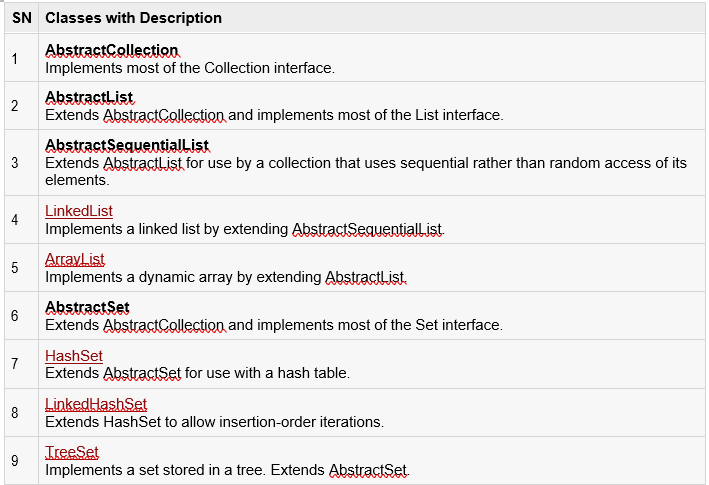
inputStream:

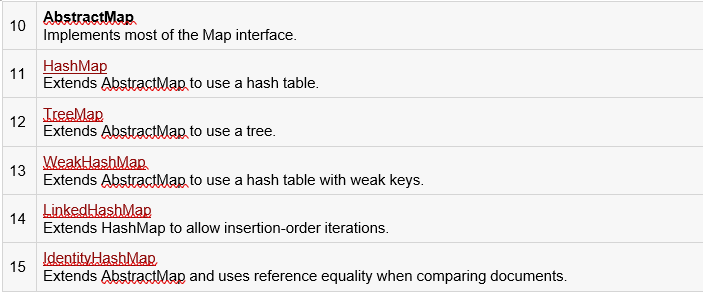
* FileInputStream: Read byte data (8bit)
* ByteArrayOutputStream: allows to read data to multiple file together( byte data in ‘n’no.of files)
* FilterInputStream: 2types:
  + DataInputStream: supports primitive data types, and can go through in future.
  + BufferedInputStream: to improve performance (fast). By storing data in buffer and then from buffer data return in destination file. (for others directly giving data onto destination)
* ObjectInputStream: read all primitive as well as graphs of javaobjects that is written to output stream.
* (Same for outputStream)

**Collection:**

* Can increase, decrease, or shrink dynamically
* Class in java
* Contains heterogenous elements also
* Can modify size dynamically
* It is an interface
* List, map, set, queue- interfaces in java
* Queue-first come first implement
* Stack- first come last implement
* List- ordered elements but can have duplicate also
* Map- key, value pair(unique elements no duplicants)

Collection classes:





Iterator:

Iterator enables you to cycle through a collection, obtaining or removing elements. ListIterator extends Iterator to allow bidirectional traversal of a list and the modification of elements.

Collection only works on object type

To overcome this ***Generic type*** came into picture

Java Generics:

Following are the rules to define Generic Methods:

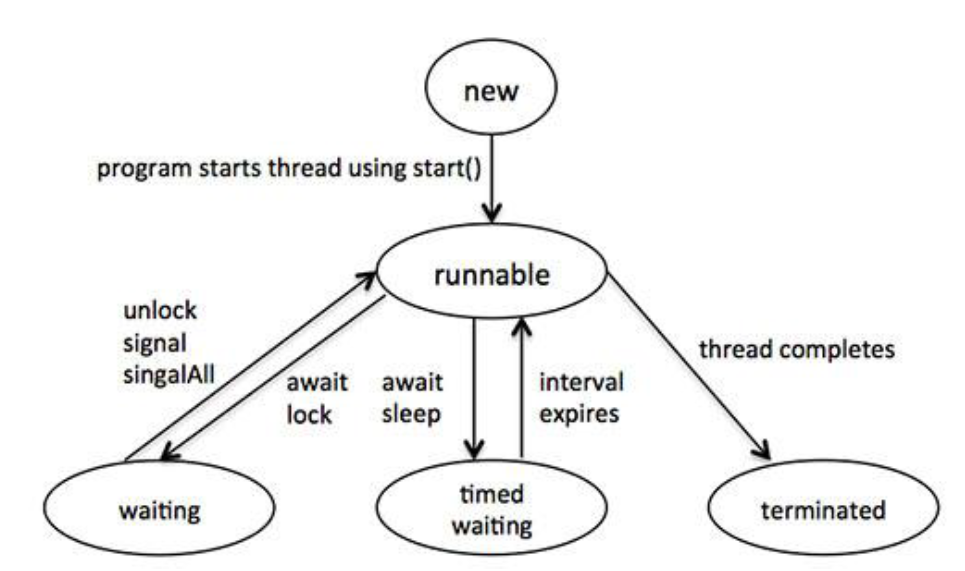
* + All generic method declarations have a type parameter section delimited by angle brackets (< and >) that precedes the method's return type ( < E > in the next example).
  + Each type parameter section contains one or more type parameters separated by commas. A type parameter, also known as a type variable, is an identifier that specifies a generic type name.
  + The type parameters can be used to declare the return type and act as placeholders for the types of the arguments passed to the generic method, which are known as actual type arguments.
  + A generic method's body is declared like that of any other method. Note that type parameters can represent only reference types, not primitive types (like int, double and char).

Thread:

Class which allows multitasking

A thread goes through various stages in its life cycle

Thread life cycle:



* **New:** A new thread begins its life cycle in the new state. It remains in this state until the program starts the thread. It is also referred to as a born thread.
* **Runnable:** After a newly born thread is started, the thread becomes runnable. A thread in this state is considered to be executing its task.
* **Waiting:** Sometimes, a thread transitions to the waiting state while the thread waits for another thread to perform a task.A thread transitions back to the runnable state only when another thread signals the waiting thread to continue executing.
* **Timed waiting:** A runnable thread can enter the timed waiting state for a specified interval of time. A thread in this state transitions back to the runnable state when that time interval expires or when the event it is waiting for occurs.
* **Terminated:** A runnable thread enters the terminated state when it completes its task or otherwise terminates.

JAVA INNER CLASS:

* **Java inner class** or nested class is a class that is declared inside the class or interface.
* We use inner classes to logically group classes and interfaces in one place to be more readable and maintainable.
* Additionally, it can access all the members of the outer class, including private data members and methods.

Advantages:

1. Nested classes represent a particular type of relationship that is **it can access all the members (data members and methods) of the outer class,** including private.
2. Nested classes are used **to develop more readable and maintainable code** because it logically group classes and interfaces in one place only.
3. **Code Optimization**: It requires less code to write.

Need of Java Inner class

* Sometimes users need to program a class in such a way so that no other class can access it. Therefore, it would be better if you include it within other classes.
* If all the class objects are a part of the outer object then it is easier to nest that class inside the outer class. That way all the outer class can access all the objects of the inner class.