**Stacks and Heaps**

**Stack**-Methods, local variables, referance of objects are stored in Stack(LIFO)..these are allocated and deallocated(last allocated gets deallocated 1st)

- Fster than Heap

- Life till current method or var's are executed

**Heap** - Objects are created(stored) inside heap, global vars

Each object in heap has refrence to refrence variable in stack. Once ref var are deallocated(when program executes), objects(becomes null or without ref) are removed/destroyed from heaps and goes into GarBage Collector

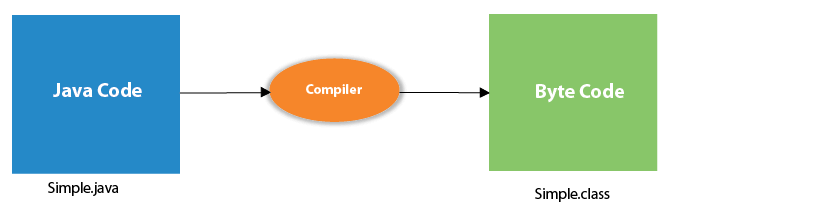
- Slower than stack

- life trhougout program execution

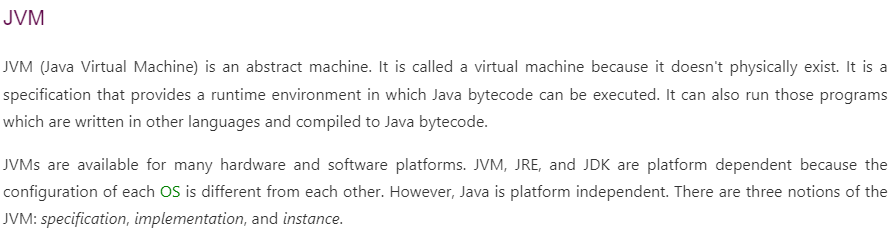
**MetaSpace**- Static variables and methods

**How Java works:**

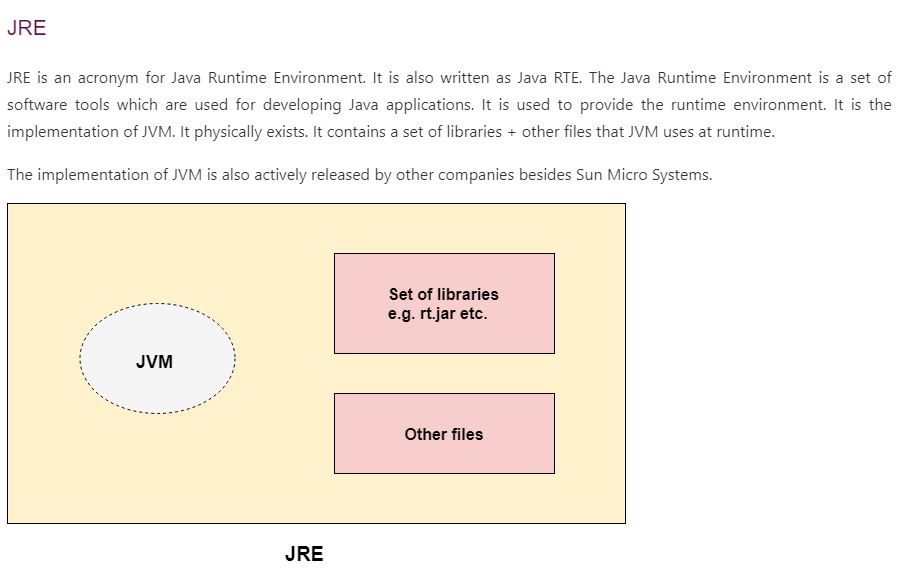
When we compile Java program using javac tool, the Java compiler converts the source code into byte code. At compile time, the Java file is compiled by Java Compiler (It does not interact with OS) and converts the Java code into bytecode(.claass file)

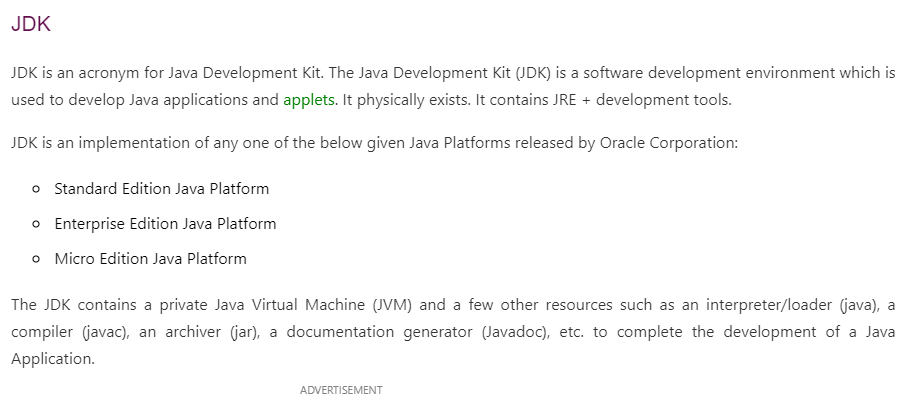


**JVM,JRE,JDK**



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





**Wrapper class:**

Programe in eclipse

**Compile time vs run time:**

Compile-time is the time at which the source code is converted into an executable code while the run time is the time at which the executable code is started running.

### Compile-time errors

Compile-time errors are the errors that occurred when we write the wrong syntax. If we write the wrong syntax or semantics of any programming language, then the compile-time errors will be thrown by the compiler. The compiler will not allow to run the program until all the errors are removed from the program. When all the errors are removed from the program, then the compiler will generate the executable file.

### Runtime errors

The runtime errors are the errors that occur during the execution and after compilation. The examples of runtime errors are division by zero, etc. These errors are not easy to detect as the compiler does not point to these errors.

Difference:

Compile-timeRuntimeThe compile-time errors are the errors which are produced at the compile-time, and they are detected by the compiler.The runtime errors are the errors which are not generated by the compiler and produce an unpredictable result at the execution time.

In this case, the compiler prevents the code from execution if it detects an error in the program.In this case, the compiler does not detect the error, so it cannot prevent the code from the execution.

It contains the syntax and semantic errors such as missing semicolon at the end of the statement.It contains the errors such as division by zero, determining the square root of a negative number.

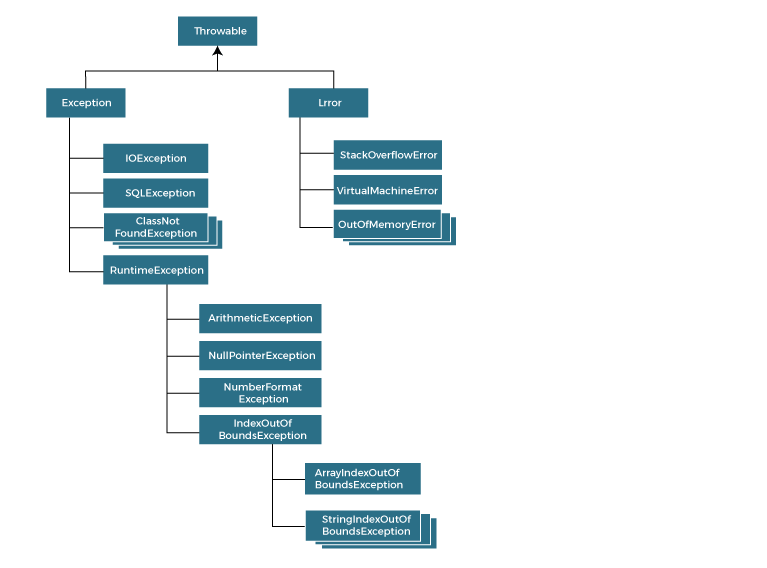
**Exceptions handling:**

The Exception Handling in Java is one of the powerful mechanism to handle the runtime errors so that the normal flow of the application can be maintained.

In Java, an exception is an event that disrupts the normal flow of the program. It is an object which is thrown at runtime.

Exception Handling is a mechanism to handle runtime errors such as ClassNotFoundException, IOException, SQLException, RemoteException, etc.

The java.lang.Throwable class is the root class of Java Exception hierarchy inherited by two subclasses: Exception and Error. The hierarchy of Java Exception classes is given below:

****

1. Checked Exception
2. Unchecked Exception
3. Error

Checked Exception: The classes that directly inherit the Throwable class except RuntimeException and Error are known as checked exceptions. For example, IOException, SQLException, etc. Checked exceptions are checked at compile-time.

**Unchecked Exception:** The classes that inherit the RuntimeException are known as unchecked exceptions. For example, ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException, etc. Unchecked exceptions are not checked at compile-time, but they are checked at runtime.

#### Note: Final block can exist with try block without catch block. **For each try block there can be zero or more catch blocks, but only one finally block.**

#### **The finally block will not be executed if the program exits (either by calling System.exit() or by causing a fatal error that causes the process to abort).**

**Order of exceptions:**

Since IndexOutOfBoundsException is a superclass of ArrayIndexOutOfBoundsException exceptions related to both classes are handled in the first catch block making the later one non-reachable.

Since **Exception** is the superclass of all the exception classes, if you place the catch block that catches it earlier to the catch blocks catching any other exceptions, all exceptions are handled in the Exception block itself making the remaining blocks unreachable.

Note: Nested try block

When any try block does not have a catch block for a particular exception, then the catch block of the outer (parent) try block are checked for that exception, and if it matches, the catch block of outer try block is executed.

If none of the catch block specified in the code is unable to handle the exception, then the Java runtime system will handle the exception. Then it displays the system generated message for that exception.

## **Java throw keyword**

The Java throw keyword is used to throw an exception explicitly.

We specify the **exception** object which is to be thrown. The Exception has some message with it that provides the error description. These exceptions may be related to user inputs, server, etc.

# **Java throws keyword**

The **Java throws keyword** is used to declare an exception. It gives an information to the programmer that there may occur an exception.

**Why string is Immutable?**

String once created its value cannot be changed only its reference can be changed.

For more details: see this site [scalar link](https://www.scaler.com/topics/why-string-is-immutable-in-java/)

**String Pool**

In java, a String pool refers to a storage area in heap memory. Memory allocation to a String object is a costly process in terms of both time as well as memory. The JVM (Java Virtual Machine) performs a sequence of steps while initializing string literals to reduce memory overhead and increase performance efficiency.

To reduce the number of String objects produced and hence the overall memory consumption, the String class keeps a pool of strings. Every time a string literal is generated, the JVM checks for its existence in the string pool first. A reference to the instance of the String is returned if the string already exists in the string pool, otherwise a new String object is instantiated and added to the pool.

// "Hi" is added to the pool as it doesn't exist in it already.

String s = "Hi";

// s2 refers to the pre-existing "Hi" in the pool from s1

String s2 = "Hi";

// new operator prevents the pool searching mechanism and thus s3 doesn't share the reference with s1 or s2

String s3 = new String("Hi");

**To run testNg files from maven command:** [testng through maven](https://www.seleniumeasy.com/maven-tutorials/choose-selected-testng-xml-files-to-execute-using-maven)

**Design Pattern**

Design pattern:

A design pattern represents an idea, not a particular implementation.

By using the design patterns you can make your code more flexible, reusable and maintainable.

It is the most important part because java internally follows design patterns.

---------------------------

statement:

Problem Given:

Suppose you want to create a class for which only a single instance (or object) should be created and that single object

can be used by all other classes.

Solution:

Singleton design pattern is the best solution of above specific problem.

So, every design pattern has some specification or set of rules for solving the problems.

Advantage of design pattern:

They are reusable in multiple projects.

help to define the system architecture.

They provide transparency to the design of an application.

----------------------------

Singleton Desgin Pattern

Singleton Pattern says that just"define a class that has only one instance and provides a global point of access to it".

In other words, a class must ensure that only single instance should be created and single object can be used by all other classes.

Saves memory because object is not created at each request. Only single instance is reused again and again.

Singleton pattern is mostly used in multi-threaded and database applications. It is used in logging, caching, thread pools, configuration settings etc.

2 types of singleton pattern:

Early Instantiation: creation of instance at load time. Class c = new Class()

Lazy Instantiation: creation of instance when required.

**Creation:**

To create the singleton class, we need to have static member of class, private constructor and static factory method.

Static member: It gets memory only once because of static, it contains the instance of the Singleton class.

Private constructor: It will prevent to instantiate the Singleton class from outside the class.

Static factory method: This provides the global point of access to the Singleton object and returns the instance to the caller.

**Final. Finally, Finalize**

**final keyword** can be used with variable, metod as well as class

Variable: fianl used in variables to make it constant value. the variable cant be change anywhere in program.

Class: fianl if used with call restrict inheritance. If a class is final, then it cant be inherit by other class

Method: within a class if some method is final, then it can;t be override in child class.`

**Finally:**

used with try catch block

finally block will be executed at last after try and catch

if there is no catch block or if exception is not caught in catch, still finally h=gets execute

**Finalize:**

The finalize method in java is called by the garbage collector before an object is garbage collected.

This method can be used to perform any necessary cleanup before the object is destroyed, such as releasing resources or detaching event listeners

finalize is the method in Java which is used to perform clean up processing just before object is garbage collected.

finalize method performs the cleaning activities with respect to the object before its destruction.

finalize method is executed just before the object is destroyed.