**JavaStudy (Source: JavaTPoint)**

# Java Basics

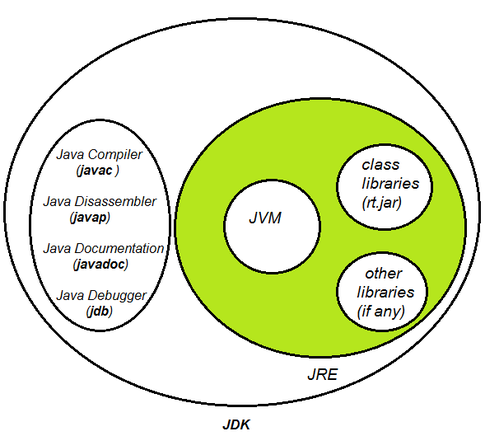
Java is a language and a platform as well.

Now, what do you understand with java as a platform? See, java is a software platform which means that it is the software and specification that is used to develop applications (EE, SE or ME) using the java programming language. Java programming language is a language to develop different kind of application software (any kind means here is for mobile, enterprise, web sites and others). Now, java is computing platform independent where computing platform is the combination of hardware and software. Hardware the literally an intel chip and software is the operating system.

Java as a language needs a compiler and run time environment which runs it.

Firstly you write code which is written using alphabets, numbers and special character. However, computer understands only byte code. So, we need to convert our code into byte code. Now, the compiler comes into picture. Compiler is solely responsible for conversion of the code written by you into byte code. JDK is the compiler. JDK is physically present in the computer. Adding one more fact for JDK, which is very important, is that it is just Java SDK.

Then the byte code needs to run to produce the result of the code written. For running the code we require runtime environment. JRE is the runtime environment for running the code and it is inside JDK only. It too physically present in the computer. There is another thing called JVM (Java Virtual Machine). Java virtual machine is actually a concept which provides java a run time environment which, if taken in details have many important task. JVM is platform dependent and that make java platform independent. That will be explained later.



**How actually you run a java program** –

Firstly, let’s talk about the most basic way of doing it.

You write code on a notepad and save it in your workspace in .java format. Then, open the command prompt do some basic stuff like setting the CLASSPATH and PATH. Then, you compile the program using the “javac” command which lead to create the source course file (.class file). Then, you can run your program using “java” command.

**Below are the steps to run your java code through CMD.**

1. Open cmd.Firstly, you have to set the JAVA\_HOME and Path.
2. Once you do that give command “javac” to check if the environment variables are set properly or not. If javac is not identified then it is not set properly. So try again.
3. So we have set the environmental variable which means our computer knows where JDK and JRE are stored. When you open the command prompt, you write everything after a path mentioned. I hope you understand what I mean. That path is the path where the .java and .class files are found.
4. To change to location (directory) where the .java and .class files are stored. We have cd command. Type cd which is followed by the new path.
5. Then, to compile a program use javac command. Like “javac program.java”. Compilation error come when the code is written incorrect. Exception occurs when an abnormal situation arises.
6. For running the multiple classes which are dependent. We have to create them and put into one folder. Then, if one file is dependent over other than that file needs to be compiled first. For this, we will compile all the classes firstly and then execute any. Though, this needs more practice but currently out of scope.

This way you can simply run a java code on CMD.

**Important Notes:**

1. How to look for .java files of a project made? – go to properties and check the Location.
2. How to change the directory in eclipse? – Just go to file and select change work space. If you want to create a new workspace then just create folder where you want to make work space and copy its path.

However, as the complexity increases, working with command prompt get difficult.

So, we use an IDE (Integrated Development Enviroment). Eclipse is an IDE which is mostly used. Other IDE available is the market are netbeans, android studio and etc.

Summary – so, for compiling the java code we need JDK which is java SDK. After compiling, we need JRE which is present in the JDK itself. Now, JDK can be of SE, EE or ME which are standard, enterprise, micro (mobile) kind of application. Until sometime, we had no option but to use command prompt to write, compile and run code. To escape the pain due to complexity of code, some great group of people invented IDE. So, IDE makes the programmer work easier. Famous IDE for java are eclipse, netbeans (oracle offers netbeans) etc.

## Properties of java

1. Java is platform independent

Java is platform independent means that it can run on any platform (windows, OS, etc) without changing the code. A dependent language will require a change in code when the platform is changed. Now, how is this possible that java is platform independent? The credit goes to JVM i.e. Java virtual machine. As the name implies, it is virtual machine itself which in term means that the java code run on JVM and this JVM is installed on the platform used. Also, JVM is different for different platforms. Here platform isn’t just the operating system but the combination of hardware and software like mac OS with Intel processor. In short, unlike java, JVM is platform dependent.

Note: I have got a point that since .class file is given to different platform then all should be able to understand the code (actually byte code). Then, what is the need to have JVM. Actually, different platform will not be able to understand the byte code since they do not know about java and so won’t be able to recognize that what is meant by the byte code. So, platform dependent JVM is required.

1. Java program are “write one, run anywhere” type code. That again portrays that it is platform independent but in a slightly different way. Let’s suppose you have written a java code on windows and want the same code to run on MAC OS. Then, that code will run smoothly but how? Actually, every machine needs JVM, so that is needless to say that OS will also have JVM installed. So, no matter of which platform written java code you are trying to run on some other platform, it will run because JVM is there to translate and run the code for the platform.

Note: We have major operation systems which are windows by Microsoft, Mac OS by Apple, LINUX by developer community and iOS platform for mobiles. Hardware (process like Intel, AMD or ARM) can differ by the architecture. So, the different combination of software and hardware makes a different platform.

## Uses of java

1. Creating web application – JSP (Java server pages with normal HTML can create dynamic web pages)
2. Applets – kind of java program used to add new feature to the Web browser.
3. J2EE – use by enterprise to create big web applications to transfer XML structure documents.
4. Javabeans – javabeans is mainstream java component model. It is similar to visual basics\*. It is also use the number of javabean (a component of java program) to create something new. This is done by visual manipulation in a buildertool. Sun has named that builderbox as beanbox.

(\*visual basics is created by Microsoft for using a GUI to buildsome new using a language BASIC which similar to beanbox.)

1. Mobiles – many games, service and operating system is based on Java. Most famous and successful java based operating system is android.

## Types of application java can build

1. Standalone application – these are also called as desktop application. These are the applications which do not interact with any other application. Examples are media player, antivirus.
2. Web application – these are also called as server side applications or web site more commonly. These are actually gets stored in server and used by the end user through browser.
3. Enterprise application – enterprise application are also web application but are more complex and have more feature. However, technical difference among them is that the enterprise application runs in a special container called enterprise container which have special features like transaction, JMS, etc (I have no clue about these special features). Example – IBM Webshere, oracle IAS, etc. While the web application runs in a normal web container. Examples – tomcat, JavaWebServer.
4. Mobile application – these are better called as apps. But before this, java is responsible for the creation of many games, services, operating system (like android).

## Facts about java

1. Object oriented – Object oriented language is object centric. Now, an object is any physical or virtual entity (an object like car, boy, animal). So, every object has some properties and states. The property is defined by the data

Note: Though some people say that java is not completely object oriented language since it uses the primitive data types. How does primitive data type influence this?

1. Platform independence – ability to run the code on different set of hardware and software without changing the code.
2. Simple – java is simple to learn and implement.
3. Secure – few of its property makes it secure like efficient security model which do mean that takes care the download of program from untrusted sources through sendbox, no use of pointer i.e. no access to memory locations, exception handling (prevents the crash of the application). Conversion of raw code into java byte code is a key feature and garbage collection, access control functionality to add a few.

Note: Do more research on how exactly these properties of java make it secure?

1. Architectural neutral – well, at first glance looked like it is an trivial topic but no I was wrong. Being platform independent means that java can work with the same code on any pair of hardware and software for which JVM is available. Therefore, any software (operating system) supports the same code. Similarly, architectural neutrality signifies the ability of java to run of any hardware (processor).
2. Portable – since java is architectural neutral and platform independent, so its code can be run over any system or processor. So, java code written on any machine can run over other machine with different hardware and software is meant by being portable.
3. Multi-thread – as the name suggest, java program can have multiple threads which perform some different task and can run simultaneously and hence ending up better utilization of resources available. Though this topic is not that simple so will study in detail later.
4. Interpreted –See, there is a difference in compilation and interpretation. Java is compiled language because it surely converts the java code to java byte code. Right? Now, there are two possibilities after the JVM receives the byte code. First, JVM will interpret it and run it by itself. Second, later invented technique, JVM has JIT (JUST IN TIME) compiler which compiles the code further from byte code to the code understood by the machine (hardware and software). Now, why is the requirement of JIT? That will be answer in the point below.
5. High Performance – let’s say that JIT is not yet developed and we use the general method java code to byte code and byte code was interpreted and run only by JVM. But JVM is capable of handling one byte code at a time. Now, the JIT is developed and which takes the byte code from JVM and again compile it to the code types understandable to the processor or the machine. So, then the code run more quickly and efficiently.
6. Distributed – we can create distributed application through java. These distributed application are those which can be used on different computers at the same time and stored in the cloud using cloud computing.
7. Dynamic – java is called as dynamic (more dynamic than C and C++) because a lot of run time information can be used verify and resolve access to object. Also, a language is called dynamic when loads the resources at run time. Java loads the byte code at the run time and hence a dynamic language.

Different editions of java –

SE, EE and ME are the editors types which we already know. ☺

## History

Well, I guess this much history is sufficient to know the James gosling with his team at sun microsystem invented java in 1992 (named Oak while invention but while patent it was named java).

Java SE versions – last version SE 8 released on 14th march, 2014. While the next release SE9 is supposed to release in 2016.

Java EE version – last version released is EE 7 in early 2013 while the next version can be released in 2016 or 2017.

## Java Virtual Machine

Well we know about it already. Because of JVM, java is platform independent, architectural neutral, efficient and have many cutting edge advantages. JVM provides a run time environment. JVM receives the byte code after the compilation and then run the code by itself. But executing byte code is comparatively slower than executing machine language code. This leads the development of JIT compiler of JVM. Now, the byte code is given to JIT compiler at the run time and it converts the byte into machine language code during the runtime and then JVM can execute the code faster. Hence, the process become fast, as processor can run the code faster.

## Java Installation

Well, if you are running the java program on command prompt then *javac* is the command for compiling the java program and *java* for running the program.

Also, the IDEs like eclipse, netbeans and etc are available in the market which you can use for creating software. I believe that I already know what the benefits of using it are.

Now, after installing eclipse or even using the command prompt, environment variable needed to set up.

Before looking at how we set these variables. Let’s have a look on what these environment variables are?

Environment variables are the variable accessible to all the programs running under operating system. These are used for storing the system wide values. In simplest words, a lot a programs on an operating system and need to refer information like which are the file to be executed, where to store the temporary data and similar. So, these environmental variables have such information and for the smooth functioning of the program all the required environment variables should be set.

Examples of environment variable of windows OS –

1. COMPUTER NAME AND USER NAME – stores the computer and current user name.
2. OS – operating system
3. System root – System root directory.
4. PATH – a list of the directories for searching executable program (in simplest word, it tells your machine where is the executable program is kept). Now, what is that executable program? In simple words, the executable program is the one which is directly readable by the operating system and these executable programs are in .exe file. For example, javac.exe, java.exe. so, until the operating system finds your jdk (javac.exe) for compilation then how will the program will run. So, setting path is highly important.
5. CLASSPATH – a list of the directories for searching the java class files (.class file containing the java byte code) and JAR files (a package file of many class files, its meta data and resources to distribute the application on the platfrom). Java compiler and java runtime searches these classpath entries for java classes referenced in your program.

Difference between PATH and CLASSPATH -

(Does the two, path and classpath, looks same? Yes. But only looks while they are completely different. Path indicates the files which are executable to the operating system (like javac.exe) but the classpath indicates the file which the java compiler and java runtime looks for (HelloWorld.java and HelloWorld.class). Path is for setting the environment for operating system which classpath set the environment for java (another way of differentiating the path and classpath is that Path is set to locate the jdk and class path is set for .class and JAR file).

Now, how to set the environment variable:

This can be done on command prompt or going into computer properties. (jo direct property wala hai wo path set karne ka hai, class path set karne ka nahi).

Note: what is the difference between PATH,javahome and CLASSPATH? PATH is to point location of the executable files of JDK while CLASSPATH is to point the path of the .class file.

## Hello java program

I have made the first java program hello world. There are few key words which I need to understand.

Public – it represent a access modifier which means visible to all.

Class – keyword to introduce class.

Static – For main method it is made a rule while creating java language that it must be called static. Its basic reason could be that it can instantiate itself.

Void – no return type.

Main –main method is mandatory for every class to have.

String args[] – it is command line arguments which are used to take the input typed in from command line/console. It is taken as the input for the program and has better control over the working of the program. Check the program for printing the value of the string.

Note:

1. These keywords can change the positions within themselves.
2. You may or may not give semicolon at the end of the class.
3. Public static void all keyword should come while declaring main method.

## Program internal

Although, we know the process of how a code is compiled and executed. However, there are few things to add. Compilation byte code gets created. This is nothing but “.class” file. Now, there are a few process happens during run which I feel is much of depth then the scope of the topic. But we should at least know that JVM then interpret the byte code and run the code. If not that, then JIT will compile the code again and machine itself will run the code.

Some questions:

1. Can we save a source file by other name than class name? Yes. If the class is not public and the compiled byte code will be at the class name anyway.

## How to set path

Again, we do know already how to set path variable and why is it important.

## JVM, JRE and JDK

We already know all this.

## Internal details of JVM

Well, JVM basics we already know that it provides run time environment. Also, it is platform dependent and that why java in platform independent. However, if we further simplify its function than we can say that it has to load, verify, execute and provide runtime environment.

It also deals with memory, garbage collection heap. Also describe about JIT.

## Variable and Data Type

**Variable** – Variable are nothing but the memory locations which can hold data. They are of three type local, instance and static.

**Local Variable** – well, these are written inside a method, constructor or a block. Local variable is only visible only within the declared method, constructor or block. They do not have any default value so they must to be initialized before accessedi.e. to be assigned an initial value.

**Instance Variable** –these are written outside the method, constructor or a block. These are used when a certain variable we require in many methods in the class (local have scope only to one method). Whenever object is created, memory is allocated to instance variable as well. Same happens when the object is destroyed. Instance variable can be declared before and after use and have some default values. These can be made public or private.

**Static Variable** – these are also declared inside the class but outside any method, constructor or block but uses static keyword. Now what is the need of static variable? Everything has its own purpose. An instance variable declare in one class cannot be used in the other class, but if we write static before the instance variable then it becomes static variable and can be used in other class without creating the instance. Static variable are not much used except they are made constant with the use of FINAL keyword. Static variables also hold the default values same as instance variable. Also, the static variable is one for the entire class so save memory. Static variable cannot access instance variable. Check static keyword section that has better explanation.

Note: local variable does not have the default value while instance and static variable have. Therefore, we must give some value to the local variable before using it. The purpose of giving default value is to avoid the program to go in an unstable situation. Because if no default value is set and no value given explicitly to the variable and if used then the program will go into unstable condition.

**Data Type**

Data is basically of two types which are primitive and non-primitive. Primitive (boolean, char, byte short, int, long, float, double) are those which are predefined data type while the non-primitive (string, array, etc) are those which are derived by the primitive data type. The non-primitive data types are those which refer to an object which can contain some data. String, array, Integer (all wrapper data type), creating object of the classes and many more are the non-primitive data types which are highly used in java.

Java literals – they are represented directly as a constant value without any computation. Like, int a = 1;.

Another important thing here is that the integers can be expressed as octal and hexa decimal. For example, int octa = 0114 (0 indicate base 8 i.e. octal) and hexa decimal ass int hexa = 0x1144 (0x indicates hexa dcimal).

Also, learn the special escape sequence used in java - \n, \r,\f,\b, \s, \t, \”, \’, \\, \ddd, \uxxxx

\n = new line, \r = carriage return i.e. at the beginning of line, \f = form feed i.e. at the beginning of new page, \b = backspace, \s = white space, \t = tab, \” = print quote, \ddd = Octal character and \uxxxx = Unicode.

## Unicode system

Well, for the standardization of all languages, Unicode was created which is followed by all languages. Actually, there are many language standards like ASCII, KOI-8, ISO 8859-1 which are American, Russian and Europian language standards. So, according to the place the character code change and so we got Unicode which are used as standard everywhere.

## Operators in Java

**Arithmetic operator** – there are basically 7 arithmetic operators. Then, total of 6 **relational operators**. Until here everything looks fine. Then**, bitwise operator** comes in picture. They have &, |, ^, ~ and shift (left, right, fill zero right shift). Let’s have a close look at the shift bitwise operator.

1.       Left shift – operand is 5: 0101. Operation to be performed is operand << 2. This command means to shift the number of bits left side the number that you mentioned (2 is the number of shift in this case). The answer will be 10100.

2.      Right shift – Simple right shift symbol is “>>”. It is different for positive and negative numbers. With positive number, it simply sift the bits the number of times it is asked. If we have to right shift a negative number then we shift the bits as the number of times mentioned and the places that become vacant on the left side will be filled by 1’s. Check examples through the link.

3.       Fill zero right shift – This is shown by symbol “>>>”. This is also different for positive and negative numbers. With positive numbers, it simply shifts the number as right shift. But, it behaves differently with negative numbers. It shifts the bits as the number of times asked and fills the left side vacant position with zeros.

(For explanation check<http://www.javaranch.com/campfire/StoryBits.jsp>).

**Logical operator**– These are logical operator which are three types. 1. Logical AND - if a is T and b is F. then, the normal And table is studied earlier will make a && b = F. 2. Logical OR – normal OR table will make a OR b = T. 3. Logical NOT. If a is T. Then, !a will be F.

**Assignment operator** – they are basically of 11 types but easy to understand. 1. =, 2. +=  (a+=2 means a = a+2), 3. -= (a-= 2 means a = a-2), 4. \*=, 5/=, %= (a%=2 means a = a%2), <<= (a<<=2, means a = a<<2), >>=, &=, ^=, |=. X`

Note: a^=2 is a=a^2. This is means as XOR for a and 2. Here, 2 is not the integer 2 but we have to operate the bits equal to integer 2.

**Miscellaneous operator** –

1.    Conditional Operator (?:) – lets learn it through an example. A = (b==2)?20:30; which means that A will be 20 if b==2 and A will be 30 if b is not 2.

2.    InstanceOf Operator – this instance of operator is used for reference data types. This only checks if the data belong to a particular reference data type or not (like String, Array, etc) and produce a Boolean result.  For example – first a string name is declared and instantiated as “John”. Then, Boolean result = john instance of String. Result will be true. Example 2, Vehicle obj = new Car(). (vehicle is parent class and car is child class and this is called as upcasting) then, Boolean result = obj instance of Car. Result true.

3.    Precedence of java operator – check the precedence of the operator in an expression. Though, this is not a thing to learn.

**Frequently asked Java Programs**

These are some frequently asked java questions.

1.       Fibonacci Series – using recursion and without recursion.

2.       Check prime number

3.       Check pelindrome number

4.       Print factorial number

5.       Check Armstrong number

6.       Sort an array using bubble sort algorithm.

7.       Selection sort

8.       Insertion sort

# Java OOPs Concept

## Advantage of OOPs

The advantages of OOPs languages are its special features such as abstraction, encapsulation, inheritance and polymorphism. We have taken a glance at what these feature are all about and their advantages. There is something called as object based programming language (Javascript, VBscript) are different the OOPs only as do they not have inheritance feature. Else both are same.

Please answer DO YOU KNOW section.

Do You Know ?

|  |
| --- |
| * Can we overload main method ? * Constructor returns a value but, what ? * Can we create a program without main method ? * What are the 6 ways to use this keyword ? * Why multiple inheritance is not supported in java ? * Why use aggregation ? * Can we override the static method ? * What is covariant return type ? * What are the three usage of super keyword? * Why use instance initializer block? * What is the usage of blank final variable ? * What is marker or tagged interface ? * What is runtime polymorphism or dynamic method dispatch ? * What is the difference between static and dynamic binding ? * How downcasting is possible in java ? * What is the purpose of private constructor? * What is object cloning ? |

## Java Naming Conventions

These are not rules which mandatory to be followed but are conventions to make your code standardize and easy to understand. Now, just be a little careful with starting a name with upper case or lower case. Also, use camel casing. That’s it.

## Objects and Classes in Java

Object – An object has three properties which are state (data or variables), behavior (method) and identity (object identity is implemented by unique ID, infect as a class can have multiple object and but always every object is unique). To relate an object to a class, we can say that an object is an instance of a class.

Class – A class is a group of objects having similar properties. For example, a student class can have object as S1. Here, student class can have n number of student but an object belongs to a unique student. A class has data member, methods, constructor, block, class and interface. (Here you can see that a class can be contains in another class which means that class nesting is possible. However, method nesting is not possible in java).

So far the difference between a class and object is clear. (nahi hua to ek bar sunnle hindi mai, ek class agar student name ki hai to wo koi particular student k bar mai nahi hai balki group of student having same properties like name, roll name, college name. lekin jo object hai wo ek unique student k bare mai. Isiliye ek class mai multiple objects ho sakte hai). Practice the programs for declaring the object and simply displaying its values. Second, declaring object and first passing value to one method and displaying through other method.

Anonymous object – if we have to create only one object than use anonymous object. It’s like “new Student()”. It may be used as Class c = new class(new student(system.in)). Creating multiple object in one statement – “student obj1 = new student(), obj2 = new student();”

Note: “An object can be physical or logical but a class can be logical only”. Firstly, many people do not accept these terms logical and physical in regard to classes and objects. But, others do. Classes can be understood logical because they just contain the logic of processing the data while object can be logical and physical because they actually contain the data in the memory. But there is another argument. Class also resides on the memory otherwise how they can exist. In the nutshell, it is quite ambiguous.

## Method Overloading

Method overloading is having multiple methods with the same name but with different parameters (here parameter is different in terms of number of arguments and types of arguments). There are two ways of overloading which are –

1. Through number of arguments
2. Through type of arguments

We cannot do method overloading with the return type of the method, if did so will give compile time error. The return type overloaded methods can be different. Method overloading is also the compile time overloading.

Now, can we overload main method? Yes, we can. But main method with string args[] will run first because it is the entry point of any class.

TypePromotion – well, this is something interesting. Let’s say that the arguments are int and long. But, the inputs given in main method were int and int. then, also argument will be accepted, even if exact match was not found. Some set of data type can be promoted to other.

**Benefits of Overloading** – There are two advantages which I tracked. First, it is definitely good to remember the methods have the same function but accept different set of parameters. So, you have to remember lesser names. It really helps developers in big applications. Second, we can have different types of constructors. Without method overloading we could not have multiple constructors of a class. Also, it saves some memory. The logic behind this is that the byte code of all the overloaded methods set is same. However, some people deny this as well. Infect, I also do not think so.

Note:

1. We can overload a static method but we cannot override it.
2. What is operator overloading and why is it not possible in java? Operator overloading is having different meaning of a same operator. This is not possible in java. It is done to avaoid complexity and bring simple and clean implementations.
3. Return type in overloaded method can be same or different. However, return type alone is not sufficient for compiler to identify the method of be called.

## Constructor

Constructor is the special type of method with the name same as class name and does not have a return type. They are of two types which are default and parameterized.

**Default constructor**–A constructor with no parameter is known as default constructor. The default constructor is used to create an object of the class. If we do not create any constructor by our own then the compiler creates a default constructor with empty body which then used to create objects. Constructors are also used to give initial value to an object. If your main method does not have an object created (through constructor), then you cannot call any non-static method or variable of the same class. So, constructor is very important in java.

**Parameterized Constructor –** A constructor with one or more parameters is called as parameterized Constructor. We can pass values to the variables of the constructors. (Though, I found that that’s not the only way). When we declare constructor in class, default or parameterized, the compiler does not declares a constructor itself (that is because the constructor provided by the compiler is empty body while even if you write your own default constructor with body then compiler may get confuse). We can also use parameterized constructors for creating object.

Note: An advantage of constructor over a method is that we do not need to call it explicitly. Also, we can change the value of a final variable (with no value set) using a constructor (default or parameterized).

**Constructor Overloading** – this also done the same way we do method overloading. Either the number of argument or the type of argument is different.

**Copy constructor** – Chaho to kya nahi hai assan! Well, I had spent around 2 hour to just understand a topic called copy constructor. At some point in time, I felt that I should leave it for now and move forward. But, I understood at last. In copy constructor, we create a simple parameterized constructor to invoke the instance variables. Then, we build a copy constructor which has the argument as the obj type (look at the example at javatpoint). Now, first constructorpasses the argument and first result is displayed. Then, we pass the first obj in the second obj that invokes the copy constructor and mystery solves here. The invoked copy constructor will pass the object one and display it again.

Also, check the way for copying the object without copy constructor.

Note: constructors cannot be inherited nor final keyword can be used with them.

## Static Keyword

The Static keyword is used for memory management in java. It can be used with a variable, method, block or nested class. We cannot use it with the outer class.

1. Java static variable – Whichever variable stated static will have same value for all objects of a class but will be declared only once while loading the class. When its value is changed then all the objects will use the changed value (instance variable can have different values for different objects). To make the value of the static variable constant and does not allow it to change later, we require final keyword. Also, Static variable can be called in the main method of the same class in which they are declared without creating an object(object needed for instance variable) and in other class of same package without creating any object but with the class name. Instance variable cannot do this. Practice counter variable with and without static keyword. Another very important aspect of static variable is shown with these count programs. Static variable are associated to the class but not with the instance. So, they get the memory at the time of class loading and they retain their value even after the new object is called.
2. Java static Method – Similar to a variable, a static method also belongs to class rather than object of class. A static method can be invoked without instance in its class and in the other class like this “ClassName.StaticMethodName”. Static method can access the static variable and change its value. We cannot use non-static value in the static methods including main method.
3. Static nested classes and static block – nested class can be made static and they cannot have non static variables and methods. Static block are the special blocks use to initialize class and executed according to the order they are written in. Try and collect more information if interested. Static block also cannot use the non-static value.

Note:

1. Why we make main method static? There are few logics behind why we have main method as static. First, if it is not declared static then we need to create an instance for calling the main method as well. Second, java language is designed in such a way that we need to have main method as static. Thirdly, main method is an entry point for the class. Then, it will be of great help it can be instantiate itself.
2. If you do not want to use the object rather just want to run the default constructor written by you then just write this “new Constructor()” instead of complete sentence.
3. A top level class cannot be declared static while an inner class can definitely be declared.

## This() Java Keyword

This keyword is a reference variable which refers to the current object. In tutorial, we have given 6 usage of this keyword but suggested to do only first two for the beginner.

1. *This* keyword is used to refer to the current object instance variable when the argument (parameter) and the instance variable are named same. Check example for clarity.
2. *This* keyword can be used to invoke any of the current class constructors. In simple words, to call one constructor from the other constructor using this() keyword. This is also called as constructor chaining. Well, if your understand the difference between this(something inside or not) and this.something then you will understand the trick. “This ()” means calling the constructor with no parameter. Similarly, this(int a) means calling a constructor with int a as an argument (Parameter). Now, “this.something = something” means some (ex-city) variable of the current constructor is equal to the something (city) variable. This() should be the first statement if used.
3. We can use *this* keyword for calling a non-static method is other non-static method. Though, this can be done without *this* keyword or without anything else.
4. *This* keyword (without “.” or “()”) can also be used as a parameter for a method (object type parameter). In such case, the parameter is class type. Check the example for better understanding.
5. This keyword (without “.” or “()”) can also be used for using the object of one class as the parameter in the constructor of other class and passing its object. Suppose when one class calls a constructor of other class and passed *this* keyword as parameter then through *this* keyword other class would be able to use variables of the original class. Check the example for better understanding.
6. This keyword can be used to return an instance of class. Check the example.

## Inheritance

Well, we already know about the basics of inheritance that a child class can use the methods and variables of the parent class. This offers code reusability. This is called attributed as IS-A relationship(sub class IS-A super class i.e. if programmer is extended by employee then, programmer is an employee shows the IS-A relationship). Along with code reusability benefit, we also get benefit of implementing the run time polymorphism called as overriding. Extend keyword is used to inherit the super class by the sub class. We can call the method of the super class directly by the object of sub class or by the super keyword in the non-static methods.

How to actually use members of parent class? Let’s say a class Father has one default constructor and two methods, method A, method B and static method C. Class Son extends the class Father.

1. Now, without creating the object of father, if Son wants to use the constructor of Father then it can be used this way “super()” and super() only be as the first statement of constructor of Son. The constructor call must be done in the constructor only.
2. Son has overridden amethod of Father but still want to use the father’s implementation of the same method. This can be done by using the “super.methodName()” without creating father’s object. But, it can be done by creating the father’s objectin main method.
3. Son can use the method which it does not override without creating father’s object is by super method in any non-static method. Also, in main method by fathers object.
4. For accessing any static method (obviously not overridden), it should be used by both class objects.
5. If father class wants to use the unique method of son class, then it could only be done by creating a downcast or creating an object of son. We cannot use simply by the object of father. Same goes with static method as well.

There are three types of inheritance allowed in java through classes which are single, multilevel, hierarchical while multiple and hybrid inheritance is not allowed. Single inheritance is done when one class extends other class, multilevel inheritance occurs when a class extends a class and that class extends some other class, hierarchical is done when two classes inherits a single class to form a hierarchy, when a class inherits multiple class it is called as multiple inheritance and for hybrid check diagram. When a class inherits two classes then the two classes may have same method and when child class calls it then it will be ambiguous to call method from which parent class. So, even when to try to inherit two classes, a compile time error will be generated.

Note:

1. We do not allow multiple inheritance (explained in interface section) through multiple classes but with interfaces. Also, we can simply use any method of the super class without implementing them. This happens in abstraction.
2. We should not think of inheritance as class extending class but it is also about implementing the interfaces.
3. In multilevel inheritance, let suppose class b extends class a and class c extends class b. Now, class a and class b inheritance will be same. If class c can call the class a constructor by creating an object and can call the constructor of class b by creating an object or by using “super()” keyword. If class c wants to use the class b implementation of overridden method of class b then use super keyword otherwise create an object of class a for class a implementation of the same method. Class c can call the unique method of class a by creating its own object as well just like class b. Lastly, class c can call the static methods directly of any of the class a or b.
4. In hierarchical inheritance, class a is extended by class b and class c. The relation between class a with class b and class c will be same as a single inheritance.

## Aggregation in java

I must say that aggregation is similar to inheritance. It is one directional association between two classes. They share HAS-A relationship. The need of aggregation is also for code reusability (like inheritance). For example, one class is a square of a number and other is for finding the area of the circle. So, the square class can give the square of the radius to help calculating the area of the circle. However, area class cannot help square class in anyway. So, they have one directional Has-A relation. Try an example.

Note: Inheritance – “Car is an automobile” while “Car has an engine”. So, the other class in aggregate is a part of the original class and has no meaning individually. Sometimes, we can use both of them (inheritance and aggregation) then how to decide that which one to use. For this, if we want a polymorphic condition then choose inheritance and when you just want a class to give certain result the other class can use then use aggregation. However, inheritance is difficult to undo after lot of development is done in an application because it affect will be more than aggregation.

## Method Overriding

Method overriding is the runtime polymorphism. When sub class has a method with same name but with different implementation as the parent class, then it is called as method overriding in java. The parent and the child class must have a IS-A relation (Inheritance).

Now, let’s understand the need of method overriding. Suppose a class extends a class and overrides any of its method. Now, if the sub-class wants to give some specific functioning to the inherited method related to it (the sub class), then method overriding is the only way.

Note:

**Question 1** – Can we override a static method?We cannot override a static method. This is because polymorphism is allowing the different instances (cat, dog of animal) to implement a method of parent class in different way. However, the static method does not belong to any instance. So, we cannot do this. We can change the return value of the same type in sub class.

**Question 2** – Can we override the main method? No, we cannot override a main method. Firstly, main method is always a static method. Also, it belongs to a class and only that class it is written in. This is how it is made while creating java language. While creating java language it was decided that the only way to startup your application (or class) is main() method and it belongs to that class as entry point. Also, overriding the main method does not give any sense.

**Question 3** – Difference between overloading and overriding?It is like comparing oranges with apples. We already know the implementation difference in overloading and overriding. However, the purpose of overloading is improve code readability and overriding is to provide different functionality in the sub class. Overloading is compile time polymorphism (which means overloaded method can be identified at the compilation time) while overriding is run time polymorphism (difference or selection between the overridden methods is seen in the runtime). Return type can be same or different in overloading while must be same in overriding (only till java 5).

## Covariant Return Type

Before Java 5, the return type of two overridden methods must be same. However, we can override two methods with different return types after java 5 but the return type of parent class method implementation must be non-primitive type and sub class implementation must return its own type(Sub class type). Also, the covariant return type should be narrower to the original return type. Understood the concept but not able to confirm on eclipse.

## Super Keyword

The super in keyword is a reference variable and is used to refer immediate parent class object.

Below are three usage of super class –

1. For referring the immediate parent class instance variable – let’s suppose that parent and child class has one instance variable with the same name. Now, when the same instance variable was printed then by default the instance variable of the child class will be printed. To print the parent class instance variable, we will need super keyword. Aur ha, “super” ka s capital nai hai.
2. To invoke parent class constructor – super() will invoke the parent class constructor in the child class. Though, even if you explicitly do not invoke parent class constructor, compiler will still invoke one, that too before the child class constructor. Parent class constructor using super() can be invoked in sub class constructor only, but not in any method (not even in main method).
3. To invoke the parent class method – simple by “Super.MethodName”.Interesting hai kark dekh. Write super keyword as first statement in the method.

## Instance Initializer Block

We already know what a block is. Instance initializer block is something special and used to initialize the data members of the instance once any object is created. This block gets invoked as soon as the object is created. Also, if you have any column written even before the block, the block will be called first. This is because the block gets called once the object created and then other methods are called. In case if the class extends other class then super() method will be calling the parent class constructor first followed by instance initializer block and then rest of the methods called explicitly. Also, see the example for the better understanding of the topic. Remember all the rules.

Note: they are called before main method so that all the variables must be initialized before they are used.

Note: what is the difference between static and non-static initialize block? Static block are used to initialized static variables and is loaded only once the class is loaded. They are different from instance initializer block.

## Final Keyword

We will take two cases where a variable is declared final and initialized final. Whenever a variable is declared final (I mean declare but not initialized. Alright?), then its value is finalized as 0 (default value for intiger). But we can change its value through parameterized constructor. Now, if a variable is initialized by making it final then there is no way of changing its value. Also, if the variable is static then static initialize block does the same thing as parameterized constructor.

Final method –We cannot override a method when Final keyword is used with it. It is written in the signature after public or static and before void. Try it. It is pretty simple.

Final class – If a class is made final then it cannot be extended by any other class.

Extra information – Arguments can also be made final. Then, these argument values cannot be changed.

Note – Can we declare a constructor final? No, we cannot use final keyword with constructor.

## Runtime Polymorphism

Before understanding about runtime polymorphism, we should know what is polymorphism? The most general meaning of polymorphism in java is that the methods with the same name can have different actions or forms. It is possible only through polymorphism. Polymorphism is of two types which are Compile-time polymorphism (Overloading) and Run-Time polymorphism (Overriding).

1. Compile-Time Polymorphism – The compile time polymorphism allow methods with the same name having different internal structure and perform same/different actions in the same class. The difference in the same name method here is of the number and type of parameters they have. It includes constructors as well. The same name methods that have different versions are called as polymorphic methods. Also, the compiler got to know that which version of the polymorphic methods to call during the compile time only and thus it is known as compile time polymorphism. Check example.
2. Run-Time Polymorphism – The Run-Time Polymorphism allows a method to have different actions in the classes extending the class in which they are defined. This way a class can have some implementations of its own while adding some generic behavior of the class extended or the interface implemented. Compiler gets to know which version of polymorphic method to call during run time only. Thus, it is called as run time polymorphism. Check an example.

Casting – when a class B inherits the class A, then instance of class B is used as object of class B but as well as for class A. This means the object of class B can access the members of class A as well. For this we do not need to do casting (sometimes people still do upcasting). Now, the instance of class A is object for class A but not for class B. This means the object of parent class cannot access the members of child class. So, for using the instance of class A for accessing the members of class B, we do a casting called downcasting (object A casted for class B). It is checked at run time. If the object of class A actually belongs to class B otherwise an exception is thrown which is classCastException.

Look at its types –

Upcasting– It is a process where the object of the child class gets the label of the parent class. This means the child class can use members of parent class. Syntax: Parent p = new Child(). However, this is not required to be done explicitly and is done automatically.

Downcasting – I have taken a lot time to understand this concept. Though, it is better now. Below is the way we do downcasting.

Statement 1: Animal a = new Dog();.

Statement 2: Dog d = (Dog) a;.

In the statement 1, we are creating a Animal which is a dog. In the second statement, we are creating a dog and making it equal to the animal as dog type created. We will not get type cast exception in this case. Now, if we do this the below then we will get the type cast exception.

Statement 1: Animal a = new Animal();.

Statement 2: Dog d = (Dog) a/ (Dog) new animal();.

We will get casting exception in the above case because in the statement 1, we have created an animal but necessary a dog. In the second statement, we have created an instance of dog and making it equal to unknow animal. Thus, we will receive an error.

Important link for Casting:

<http://www.cs.utexas.edu/~cannata/cs345/Class%20Notes/14%20Java%20Upcasting%20Downcasting.htm>

Question – runtime polymorphism mai asia kya hota hai ki wo runtime mai hi hota hai aur aisa aur same with compile time polymorphism.

Answer – I have a little justification to offer. Check <http://www.studytonight.com/java/dynamic-method-dispatch.php> for the example of dynamic method dispatch or runtime polymorphism. In this case, “gm.type()” at compile time will look like referring the game class object but at run time it will refer to child class object. Now, why not it can be detected at compile time is because at compile time the syntax of the code is checked.

Note –

1. Difference between declaring and defining a variable, method and class specifically in java. In java, classes are declared and defined at the same time. Method, if abstract, are declared but not defined. Normal methods are like classes i.e. they are declared and defined at the same time. Variable are declared like this “int a;” and declared and defined like this “int a = 10;”.Please check this, if needed: <http://stackoverflow.com/questions/11715485/what-is-the-difference-between-declaration-and-definition-in-java>.
2. Static and Dynamic typed language – Static typing is the one in which the variable is needed to be defined/declared (the data type) before use. For example, writing “a=10” before declaring that a is of which data type i.e. int a or double then a will give error at compile time. Also, writing “int a = 10” is fine because you have declared the type at the time of defining. While the dynamic typed language can use the variable before declaring the type which means writing “a=10” will not give error before declaring a. Now, some people advocate the declaring the variable type (static typed language) is an essential requirement for any programming language and other says that using variable even before declaration saves time coding.
3. Strong and weak typed languages – strong typed languages are those which will give error at the compile time if similar to this happen “char a = 10”. This means that “a” is char and giving int value to it will give an error at compile time. While in weak language, no error arises in such situations.
4. Java is static and strong typed language.
5. Suppose, we have created the object of child class with child class reference. Now, we can call any method of parent or child class (overridden or unique). Now, if we have created object of parent class with the reference of parent class. Then, we can call all parent class method and overridden methods. But we cannot call unique methods of child class with this object. To do this, do downcasting.

## Static and dynamic binding

Connecting a method call to the method body is called as binding. They are of two types, namely, static and dynamic. In static binding, the type of object is checked at the compile time (whether object is of parent or child class). In dynamic binding, the object types get to know at the runtime only. Check examples for better understanding. Both static and dynamic binding can be seen in java.

Note: In the example of static binding, the class does not even extending any class. So, only one type of object of object is available. So, it is known at compile time only.

## Instanceof Operator

Java instanceof is used to test to which class an object belongs to. It is also called as comparison operator the instance with type. It is very simple yet interesting. Please try it. If we apply *instanceof* operator for null object (parent obj = null), then test will always come false. We already know how to do downcast. But we can downcasting involving the instanceof operator. This is done by create an object of child class in the main method. Then do upcasting. In any method, pass the upcast object as parameter. Finally, apply if condition and downcast it back. It is best to check the example. I have done an example and it was downcasting pretty well. Also, the instanceof works with object data types and wrapper classes.

## Abstract Class in Java

A class that is declared with a key word abstract is called as abstract class. It can have abstract and/or non-abstract methods (Concrete method). Abstract methods are those that do not have body. Also, we put semicolon after declaring an abstract class.

Abstraction – Abstraction used to hide the implementation and only showing the functionality. Abstraction can be achieved by abstract classes and interfaces.

1. Abstraction through Abstract classes – A class extending an abstract class must implement all its abstract methods otherwise should use abstract keyword in its signature. Abstract class may also contain the constructor and simple methods which may or may not be called in the main method. Abstract keyword is also used in the signature of the abstract method. The extending class must implement all the abstract method of the abstract class otherwise it should also use abstract keyword. So, it is not compulsory to implement all the abstract methods of the class extended, but then the extending class must also be use abstract keyword in its signature. Abstraction through abstract class is less than or equal to 100%.
2. Abstraction through interfaces – A class implements the interface must implement all the method of the interface otherwise use abstract keyword in its signature. A class extending the class A can only implement the left over methods of interface which are not implemented by class A. Abstraction through interface is 100%.

Note(s):

1. Why abstract classes are not instantiated? No. This is because of the simple reason that they have at least one abstract method and that makes them incomplete. If they are instantiated and any of the abstract method called then what the compiler will do? So, to avoid such condition the abstract methods are not instantiated by rule.
2. What advantage abstract classes have over interfaces? Interfaces do not implement any of its method whereas abstract classes may implement some of its methods. Suppose class A and class B, both implement an interface first. There are some methods of interface which are implemented in same way in both class A and class B. However, by implementing an interface both have to implement that method need to be implemented in both the classes separately. So, we need to write the same code in both the classes. In real situations, thousands of classes implement the same interface. Then, we have to write the same code thousand times. Here, abstract class can help. Abstract class can implement those methods which will have same implementation in all the sub classes and keep those methods abstract which will be implemented differently in different class.
3. What is virtual class? Virtual classes may be the one that has virtual method. Virtual method can be overridden inside the class inheriting the virtual class. There is not keyword “virtual” in java language. However, all the methods except static or final which can be overridden are virtual method by default in java. In the nutshell, all the classes which have methods that can be overridden are classed virtual classes.

## Interfaces

**Interface** - Interfaces are the blueprint of the class. An interface contains only abstract methods unlike abstract classes and hence, gives 100% abstraction. Interfaces also support the multiple inheritance and loose coupling. The variables in the interface are public, static and final and methods are made public and abstract by default.The variable of interface is made static because they cannot be instantiated so it should be made static to be used without instance and final because the programmer cannot change the value of variable of interface. An interface can extend other interface. You already know about abstraction through interfaces.

Let’s study multiple inheritance (which tells you the need of interfaces and advantage over abstract classes) and loose coupling.

**Multiple inheritances** –Multiple inheritance means one class extending or implementing (inheriting) more than one class or interface. Multiple inheritance with abstract class is not possible because a method which is implemented in both the classes which are extended and if that method is called by the extending class then it will be ambiguous to call which class method. Still confused? Check this: <http://beginnersbook.com/2013/05/java-multiple-inheritance/>

Now, what makes interfaces allow multiple inheritance. Every method in an interface is abstract. If a class implement multiple interface and suppose all interfaces have a common method A. When this common method is called in the sub class it will not create a situation of ambiguity as in abstract class because no interface implements the method. Every method is implemented by the class implementing the interfaces.

**Marker or tagged interface** – These interfaces do not have any member (neither data nor method). These are very few and we do not create them. Infect, the recent versions of jdk not required to use these marker interfaces. But for knowledge perspective, these are used just to give some information to the compiler. For ex – serialization marker interface gives an idea about the implementing class.

We can have nested interface as well. (like nested classes but nested method is not possible!)

Note:

1. An interface extends other interfaces (not implements). It does not extend classes.
2. Can we instantiate interface? No. we cannot instantiate an interface because they do not have any implementation of the method so no point of instantiating them. Though, in one situation they are instantiated. While creating the anonymous inner classes. These anonymous classes override the methods of interface.
3. What is loose coupling? When one method does not depend on other method much then the coupling is loose. Interface lets a class to interact with a class through it and use its methods which are nothing but the implementation of the interface methods. So, other classes can still be used if the implementation is later changed. This lessens the dependency and makes the code easy to maintain.
4. Interfaces cannot have private members. It is an illegal modifier.

## Abstract Class and Interface

1. Both cannot be instantiated.
2. Abstract class can have private, non-static and non-final variable unlike interface.
3. Abstract class can implement an interface but interface cannot extend/implement an abstract class.

## Packages

A package is formed when similar type of classes, interfaces, and sub-packages. Packages can be built-in or user defined. Packages are provided to better manage the classes, to provide access protection and removes naming collision. Check the method to run a package in cmd, if needed.

Now, how to access a package from other package? There are three ways actually.

1. Import package.\*;
2. Import package.classes;
3. Fully qualified name – it is “PackageName.ClassName”. we have write to it every time we will use it unlike importing which is done once and used as many time as required.

Sub packages needs to be separately need to be imported for use.

Now, what are sub packages? These are nothing but the packages under other package. For example, sun microsystem made one package for java and placed other packages like io, util as sub packages. The standard way for defining a package is “domain.company.package”. Check how to change the directory or classpath switch if need.

To save the file temporarily, either save setting classpath in cmd or switch classpath. To save file permanently, save classpath environmental variable .

Static import (Java 5 fetaure) – The static import feature of java 5 facilitates java programmer to access any static member of a class directly. This happens because the package.class.variable gets import. This is similar to importing the package.class to avoid lengthy codes.

## Access Modifiers

Access modifier defines the scope of data member, method, constructor and class. There are four types of access modifiers and these are of the following types.

1. Private access modifier – these modifiers are accessible within the class which means that the method or variable made private will give compile time error if tried using outside the class. If you make constructor private, then you cannot create instance outside the class.
2. Default access modifiers (No Access Modifier) – if you do not use any access modifier then they come use the default category and can be used within the package.
3. Protected access modifier – this category of members has scope within the package and outside as well but that’s through inheritance. This rule is not applicable to a class. This may be because you cannot inherit a class so you cannot have a protected class.
4. Public access modifier – this modifier make its members to be accessible everywhere.

Note: In case of overriding, the method in the sub class should not be more restrictive than the method in the parent class.

## Encapsulation

Encapsulation is a process of uniting the data and code. By this, we can make our class read only or write only. It provides more control over the data. Encapsulation is implemented by making all the variable private and using getter and setter methods to get and set data respectively. Practice one example. If you do not set a value through setter and try to print the getter then “null” will be printed.

Advantages of Encapsulation

1. Classes use getters and setters to get and set the encapsulated member of a class. Sometimes the requirement changes and we can then change the getter and setter method implementation without breaking the code that uses it. Also, we can change the data of the encapsulated method if needed. Therefore, we can maintain a code easily through encapsulation.
2. We can make the fields of a class either read only or write only as and when required.

Note: Encapsulation is used to bind the functionality (method) and data (variable).

## Object Class

The object class is the parent of all the classes in java. Coming straight to the point, we can refer any object of unknown type. This is because a parent class can refer to any object of its child class. getObject() is the key function for this functionality. Syntax: “Object obj = getObject();”. There are a lot of methods of object class. The object class gives lots of behaviors to all the objects exist in java like object cloning.

## Object Cloning

Trust me its simple! Object cloning is used to create an exact copy of an object. We can do it through new keyword but cloning method uses less processing. For using the cloning method of the object class, we have to implement java.lang.cloneable interface and also handle an exception CloneNotSupportedException. Its syntax is like this: “className obj2 = (Classname) obj1.clone();”. This looks to be similar to copy constructor.

Note: what is the difference between shallow and deep cloning and which type is supported in java?

## Java Array

* Array is a kind of data structure which is used to store one type of data. They have fixed memory size and that does not increase during the runtime. Code optimization and random access are its advantages. Code optimization means we can retrieve and sort the values easily and it allows random access of the values due to indexes. Arrays in java are index based and first element has the index zero.
* They are of two types.

1. Single dimensional array – the syntax to declare an array in java is “int[] arrayName” or “int []arrName” or “int arrName[]”.Declaring and instantiation happens like int[] arrName = new arrName[size]. Declaration, instantiation and initialization happens like “int[] arrName = new arrName[1,2,3]”. Following are the combinations allowed.
2. Separately doing the declaration, instantiation and initialization. (Or)
3. Declaring and instantiation together and initialization separately. (Or)
4. Declaring, instantiating and initializing altogether.

I tried to run a method to print an array. I figured out that, we can pass an array as an argument to a method. To send the array as argument from the main class we must make that method static.

1. Multi-dimensional array – In multi-dimensional, the index are given as row and column index like a matrix. Syntax for the declaration is “int[][] arrayName;”. Syntax for initialization is “arrayName = new int[n][n]”. Syntax for declaration and instantiation together is “int[][] arrayName = new int[1][1]”. Syntax for initialization is “arrayName[0][1] = 2”. Syntax for declaration, instantiation and initialization is “int arrayName = new arryaName{0,1}{3,3}” where 0 and 1 are in the first row and 3 and 4 are in second row. For printing the two dimensional array we will need two loops. Array out of the bound Exception occurs when the index is negative or greater than the size of the array.

* Since array is an object, so array object must belong to a class. Therefore,a proxy class is created for it whose name can be obtained on the object by getClass.getName() method. Try it. I made a program of two dimensional int array and the result I got it “[[I”.
* Copy of one array into other is also possible. First, declare and initiate first array. Then, declare and instantiate the second array. Then, use this syntax: “System.arraycopy(firstarray, copyfromindex, secondarray, indexcopystart, indexcopylength)”.
* Addition of two matrixes in java – First declare and initialize the arrays. Then, instantiate the third array. Then, with the help of two for loops, put the addition of first two arrays in the third and print it.

## Wrapper Class

* Firstly, we have to understand what is wrapper class? Everything is java is an object like a file, image, a URL etc. So, to convert the primitive data type to an object, we have wrapper class. These wrapper classes wrap a data type to make it an object. Wrapper classes also have the method of unwrapping the object get the primitive data type. Sometime, we want to store data and that can be stored in object form only. Then wrapper class becomes savior.
* Wrapper class provides the mechanism to convert the primitive to object type (Wrapper class) and vice versa. We can convert the primitive type to wrapper class explicitly and it can be automatically done for us by compiler as well. The automatic conversion of primitive to object types is called as autoboxing. The conversion of wrapper to primitive is called as unboxing. Check the wrapper class for the primitive type.
* Check the bidirectional conversion. For converting primitive to object use valueOf() and from object to primitive use intValue()

Note: I guess there is a problem with autoboxing in eclipse while boxing and unboxing is working fine.

## Call by Value and Call by reference

**Formal and actual parameters (Parameter and argument)**

We have to pass radius of the circle to calculate the area of a circle. Currently, a variable int a has the value of the radius. Its value is 5 that we know (somehow). Then, if we pass int a then it is appropriate to say that we have passed a formal parameter. Formal parameter acts as an identifier to stand for the value that is to be passed in the method. Example, methodName(int a). If we pass 5 (literally value), then it is appropriate to say that we have passed actual parameter. Example, methodName(5). Also, the formal parameter is called as parameter and actual parameter is called as argument.

**Pass by value and pass by reference**

In pass by value, a copy of the value is passed and therefore change made to the formal parameter (parameter) does not reflect on the actual parameter (argument). On the other side, pass by reference means a method gets the location of the variable that the caller provides. Thus, changes made are reflected in the actual parameter.

Java supports only pass by value. We can send a variable or a reference through the pass by value mechanism. When we send a value through call by value then the copy of a variable is sent. If any change made will be made on the formal parameter while the actual parameter will remain same. Now, when we pass a reference of an object then a copy of a reference is passed to the method. The original reference and copy of the reference both points to same object in the memory. Thus, when the change is made to that copy of the reference then the original reference also changes the value it is holding.

Note: Do not understand these terms by name. Because pass by reference does not mean we have passed the reference but we have shared the reference location of the variable or reference passed. Also, pass the value does not means that we can pass the value only but we can pass the reference as well.

**Edit:**

**Understand the difference between reference and object**

I can bet that the understanding will be better after reading this note. Firstly, we have to understand that what a reference is? We have a class student. Now, we have create a reference of student class like this “Student s”. Here, “s” is the reference but what do actually mean by this. This means that “s” being a reference is actually a memory address which is point to that location in memory which is captured by any object of Student. What you understand by the statement “Student s = new student();”? This means that a student which is actually an object is located at the memory location “s” which is a reference. I hope now you understand that a reference is the memory location at which the object lies.

**Pass by Value**  
Now, if we talk about the pass by value and pass by reference then the concept is fine that a copy of value is given in pass by value while the original value is given in call by reference. We can send a primitive or user defined object through call by value only in java. So let’s say we first send a variable with value 2 through call by value then a copy of that variable will be sent. Similarly, we have sent a reference “s” of student class (which we call object earlier) to a method through call by value. So, this reference “s” is nothing but the memory location address. So, the receiving object will receive the memory location of a particular object of student class. However, since the copy of the reference also holding the same address then the receiving class can use or change the data in that memory location.

**Pass by Reference**

Now, we will see what would have happened if java has pass by the reference. When we will try to send the primitive data like the same variable a holding a int 2 then the original copy will be sent so the changes can be seen in the sender method as well which was not the case in call by value. If we send a reference of a class like reference “s” of student class through pass by reference then the value itself of the sent reference “s” can be changed and this also means that “s” will not be pointing to the same memory location anymore. In the nutshell, if we sent a primitive value through PBR then value would be changed in the sender method too and if a reference is sent then that reference itself could be changed and will be other memory location.

## Java Strictfp Keyword

This keyword is used to provide a better arithmetic of the floating point data on different platforms. Since, different platform may have different precision so to overcome this difference we can use strictfp keyword. It can be used with class, interface and concrete method i.e. written in the signature of class, interface and concrete method. This can also give error when used at wrong places like using it with abstract classes, with variables or with constructors.

## Creating API document by javadoc tool

API document is the document contains the information about the API and project. It is needed to help the other programmer to know about the code. Javadoc tool is available for creating the API document in the HTML format from comments in the code.

## Java Command Line Argument

Command line argument are the argument that user can pass during the runtime. It can be taken as user input. This is only done through command prompt.

## Object VS Class

Class does not get memory when created but object gets the memory when created (well, I doubt on that). There are many ways of creating an object but only one way for creating the class.

Doubt: Class does not get its memory when it is created while object gets. Is it true or false and why?

Take complete understanding of Heap and Stack.

Few words on memory management in Java

Heap – Heap memory is used by Java Runtime to allocate memory to java objects and classes. Whenever we create any object then it gets the memory in the heap space. That is why garbage collection runs on the heap memory for the objects do not have any reference. Also, the heap is divided into two parts (generations) which are young (nursery) generation and old generation. Young generation contains the new object created and if they continuously referenced then they are moved to old generation to make space for other objects to be accommodated. When any of the generation gets full, then garbage collection takes place. So, this much about the object which are the object related data while the class related data like static methods (infect all the methods) and static variables are stored in the special memory of the heap called permgen (permanent generation). However, if the static variable is a reference of an object then it is stored in the normal heap memory.

Stack – This memory is comparably very smaller than heap. Stack memory used for the execution of the thread. They contain the value related to the methods which are short lived. Whenever a method is invoked then a new memory block in the stack is created to hold the local primitive values and references of other objects in the method. As the execution of the method gets over then that block of the memory gets destroyed. Also, every thread starts a new call stack.

## Overloading VS Overriding

Overloading aims to provide more readable code and Overriding aims to provide a specific implementation. Overloading is a compile time polymorphism while Overriding is runtime polymorphism. Overloading cannot be performed by changing the return type while overriding can be done by changing the return type.

**Concepts of oops –**

1. inheritance ek class k methods ko dusri class ko use karne deta hai (jaise API karte hai apan use wo sabs bada example hai inheritance ka).
2. Polymorphism matlab ek name k methodsko alag alaf tarike se use karna. Jaise ek mehtod mai same name aur different argument hone se hume same method agal alag form mai mil jata hai. Agar inter class bat kare to super class k method ko child class mai change akr skate hai. Jaise ki inheritance mai ye feature hia k super class ka use karo method lekin change karkke use karna hai to polymorphism ki help leni hai.
3. Fir hai abstraction. Abstraction possible hai abstract classes se, interfaces se aur bhi new technology se. Abstraction se app implementation hide kar dete hai. Jaise collections hai. Set interface k sare method kai class mai implement hai lekin hum unko kahi bhi use kar sakte hai. For example, add() method jisse set mai value add karte hai lekin wo add() function ki implementation thodi na show hoti hai. To humne bina implementation show kare kahi bhi kisi ko bhi wo functionality use karne di hai. Ye hum abstract class se bhi kar sakte hai bus abstract class se 100% abstraction nahi milta bal ki kam less than 100% hota hai.
4. Encapsulation - Encapsulation is about restricting the access to the implementation of the object’s components (component are methods here) and state of the components (component are variable here) as well as binding the data and methods operating the data.

Note:

1. Difference between abstraction and encapsulation – In abstraction, we just use the interfaces and abstract classes to let the user use the functionality without bothering about the implementation just from the sake of preventing complexity. In encapsulation, we restrict the user to see the implementation or state of the object’s component and bind the data and method operating the data. Now, why we gets confused in this is because few things are similar in this. Both involves the hiding of implementation and makes the code more maintainable. Let’s look at it. In abstraction, though the sole purpose is not to hide the implementation but to remove the complexity. However, still the implementation gets hide here too. In encapsulation the sole purpose is to restrict the user to see the implementation. In abstraction and encapsulation, the code become more maintainable as it is hidden from the user.

# Java String

## What is String?

Generally, String is called as sequence of character. But in java it is an object that represents a sequence of objects. String class is used to create a String object. There are two ways of creating a string.

1. By String literals – Check the syntax first “String s = “StringCreated””. Each time we create a string then JVM checks if there is any string already the value given (Stringcreated in this case). If there is no such string then it will create string s otherwise use the same string location to be pointed by the new string. String objects are stored in a special memory area known as string constant pool. Java uses the concept of literals to make itself more memory efficient.
2. By new keyword – Check the syntax first“String s = new String(“stringcreated”)”. In this case, JVM will create a new string object s in normal heap memory and the literal “Stringcreated” will be placed in spring constant pool. The variable s will refer to heap memory though.
3. I would say there is third way by converting the string from an array. Check this. Do not convert the array to string through the toString() methos (only here) but convert through new keywordand passing the array as an argument. With toString() method, the output array is shown in the bracket and values separated by comma so does not given feel of the string.

Do You Know ?

* Why String objects are immutable?
* How to create an immutable class?
* What is string constant pool?
* What code is written by the compiler if you concat any string by + (string concatenation operator)?
* What is the difference between StringBuffer and StringBuilder class?

Note:

1. When converting array to string then use new keyword but not toString method. If we use toString() method then we get hashcode.
2. When converting string to array then we have to use toCharArray() method. It works fine.

## Immutable String

* In Java, Strings are immutable which means not modifiable. But we can create a new String. In the example given, it shows that string remains the same.
* In the example, a string “Tendulkar” is tried to concat with the original string “Sachin”. But when the original string was printed it shows only sachin and a new object string was created which is “SachinTendulkar”. This also means that s1 will refer to memory in heap having just “Sachin”. Now, if we explicitly assign the original string s1 with the concatenate string (s1 = s1.concate(tendulakr)), then s1 will refer to memory in heap having “SachinTendulkar”.
* Now, why strings are not mutable? Well, there are multiple reasons for it. First, when a string is made whose value already exist in the string constant pool, then the new string is referred to the old string location. If the string is not immutable then change in a string made can do similar changes in the other string too. Second, Strings are widely used as parameter in java class. So, if strings are not immutable then the parameter could be change and could cause serious security threats. There are other reasons as well. For more jigyasa: <http://www.programcreek.com/2013/04/why-string-is-immutable-in-java/>.

Note: String can be created in two ways which are String s “Rishav” which is called as string literals and other is String s = new String(“Rishav”) which is through new keyword.

## Java String Compare

We can compare the string on the basis of content and reference. String compare is used in authentication, sorting and reference matching etc. There are three ways of comparing the string.

* String compare by equals() method – It compare the original content (values) of the strings. It also has two types. First, public Boolean equals(s.equals(s1)); this compare without ignoring the case. Second, public Boolean equals(s.equalsIgnoreCase(s1)); this compare ignoring the case. Also, we can pass the any other string like s1, s2 or directly write in the method as this: s1.equals(“ayushi”).
* String compare by == operator – the operator == compare reference but not values. This means that it will be checked whether to two string object point on the same memory block or not. This has nothing to do with the value.
* String compare by CampareTo() method – this method compare the two strings lexicographically which also mean to compare the values with their unicodes. If the strings differ through this method then either they have different character at an indexed position or they are may be have different length. The result of this method comes in integer form. 0 indicates that the strings are equal, positive indicates that the first string is greater and negative indicates that the second string is greater. This also has the option for ignore case i.e. compareToIgnoreCase().

Note: compareTo() method is showing negative vlue even if the first string is greater. Why? This may be happening because it checks the first element and gives the result if found the difference in the first element itself.

## String Concatenation

The string concatenation is used to create a new string from connecting multiple strings. There are two ways to doing it.

* By + operator – It is very simple method. Try it. However, its compiler transformation is quite interesting. The concatenation becomes possible because of String builder or String buffer class and its append methods. + operator can also concatenate the primitive data type too. All the primitives are added when used prior to the Strings and considered String when used after them. For example, “String s = 50+30+”Rishav”+40+40” results 80Rishav4040.
* By concat() method – This is also simple. Try it.

Note: when + operator is used for concatenation, then there is a little code transformation by the compiler happen which is like this – “String s = (new StringBuilder()).append(“String1”).append(“String2”).toString();”. This is because concatenation is denied in the Stringbuilder class.

## Sub-String

The sub-string is nothing but the part of a string. We can make a new string object from an already existing string. This can be done by using the startindex and endindex. While creating a substring, we need to mention the start index and end index. Start index is inclusive and end index is exclusive.

Note: “subsquence()” is a method which is similar to substring with two arguments.

## Java String Class Methods

Java.lang.String is the class that contains all the methods which are used to perform operation on String for concatenation, comparison, converting, trimming and replacing them. Well, without doubts, String is very important topics since almost all the web application, mobile application, and window based application treat everything as a string. So, here are a few important string methods.

* Java String toUpperCase() and toLowerCase() – these are used to convert the lower case values of a string to upper case and vice versa. Also, if few value out of all values in a string are already upper case and toUpperCase() is applied. Then, uppercase values will remain the same and rest will covert to uppercase too which means every value will be upper case. Also, it has got the same method with the additional parameter “locale” and its for the language which is not necessary to use as the system will pick up the default language (English in our case).
* Java String trim() method – trim removes the white spaces (Simple spaces/blanks) are removes before and after the string. Try it. However, it does not remove the white space between the String values. For ex – “Rishav Mishra”, will not remove the white space between rishav and Mishra.
* String startsWith() and endsWith() – these methods are used to check whether a string start and end with against the value checked and give Boolean output. Try it. When the string ends with a white space, then mentioning the white space, it is important. Also, if string start with “R”, then we will receive false for “r” which means it is case sensitive.
* Java String charAt() method – it is used to print the value at an indexed place. Try it. It prints the white space also.
* Java String length() method – this method is used to print the length of the string.
* Java String intern() method - this method is used to intern the string created by new keyword. If this method is used only while creating any string with new keyword then the it will first check as if any other location have the same value. If atleat one location is found then the new object will be pointed towards the same memory location containing the value or else it will get new memory location. Check example for understanding. Also, for pointing to same location, both the object must be interned either implicitly or explicitly.
* Java String valueOf method – This method covert the primitive type of data into string. When we try to add integer or any other kind of primitive data type it treats everything a string (same type of data type will be written simple and other type in double quotes).

Note: In simplest words, it just creates the string version of any primitive data type and then we can use it whichever way we want.

* Java replace Method() - this method replaces the whole string data or some part of string data with a new data.

Note: It has different version which are related to regex. Do it later when get time.

## Java StringBuffer Class

The java StringBuffer class is used to create mutable string which is synchronous as well (Synchronous means multiple thread use the StringBuffer without making program inconsistent). They are same in every other manner to a normal string. This class is thread safe (multiple threads can use StringBuffer without making program inconsistent, due to synchronization). It has three important constructors.

* First, StringBuffer() constructor creates an empty string buffer with initial capacity of 16 indexes. Second, StringBuffer(String str) which creates a string buffer with the specific string. Lastly, StringBuffer(int n) creates a string buffer with specific capacity at length. Please check the important string buffer methods.
* As we know, mutable strings can be created by the help of stringbuffer and stringbuilder classes. Below are some important methods.
  1. Stringbufferappend() method – This method is used to append the two stringbuffer. But how is this different from concat() method? In append, when one stringBuffer is appended to a sequence of values then the stringbuffer gets changed and values get added in the same stringbuffer whereas added values does not reflect in the old string object in concat() method.
  2. StringBuffer insert() Method – this method is used to insert a string into another string from a provided index. The provided index will be taken as offset argument which is excluded while inserting. Also, do not get confused with replace, this method will inert something and will keep the original data as well.
  3. StringBuffer replace() Method – This is same as the replace method in the string class. This replaces a portion of string buffer and puts the replacement from the start and end index provided. Start index is included while the end index is excluded.
  4. StringBuffer delete() method – This method delete the part of a string froma startindex to an endindex where the endindex is excluded from the deletion process.
  5. StringBuffer reverse() method – the reverse method of the StringBuilder class reverse the current string. Try it.
  6. StringBuffer Capacity() method – this method is used to print the memory capacity of any string buffer. According to the tutorial, the default capacity of a normal string buffer is 16 and when characters become more than 16. Then, the capacity increases by this formula: newCapacity = (oldCapacity\*2)+2. However, when I tried it shows that the default capacity is 16. When I add one char to the stringbuffer then capacity becomes 17 and then likewise. Capacity includes the white spaces too.
  7. StringBuffer ensureCapacity() method – this method is used to ensure the minimum capacity of the string buffer. Check tutorial if not remember. Little complicated to undertand through words. We do not print the “ensurecapacity”. The formula mentioned in the string buffer works in ensuring the String.

Note: See! This is not easy to clarify but I am trying my best. If it’s only about capacity method, then anything written in its description is correct. But when ensure capacity is used then there are two cases. First, if we try to ensure the capacity more than the current capacity of the stringBuffer. Then, {(2\*c)+2} will be applicable where c is the current capacity. Second, if the ensuring capacity is equal or lesser than the current efficiency of the stringBuffer then the ensured capacity will be same. I hope this observation is correct.

## Java StringBuilder Class

StringBuilder also use to create mutable string but they are non-synchronous. It also has three constructors which are StringBuilder(), StringBuilder(String str) and StringBuilder(int a). They are same as in StringBuffer. Please check the important buffer method. Some of the methods of the StringBuilder class are below.

1. StringBuilder append() Method – it is the same as the append method of the StringBuffer.
2. StringBuilder insert() Method – It is same as the insert method of the StringBuffer.
3. StringBuilder Replace() Method – this method is used to replace a part of the StringBuilder by some other sequence of char.
4. StringBuilder Delete() Method – this method is used to delete some part of the StringBuffer.
5. StringBuilder reverse() Method – this method is used to reverse the StringBuilder values.
6. StringBuilder capacity() Method – this method is used to print the capacity of the Stringbuffer. It is also same as StringBuffer capacity method.
7. StringBuilder ensurecapacity() method – same as Stringbuffer ensurecapacity method.

## Difference between String and Stringbuffer

These are the differences between the two –

1. The basic difference between then is that the String is mutable and StringBuffer is non-mutable.
2. Second, String is comparatively slow than the StringBuffer. This is because everytime we concat something then it calls a new instance. However, in StringBuffer, the same instance adds the append part it the existing string. Also, Stringbuffer takes less memory.
3. String class overrides the equals() method and that is why string can be compared by equals(). However, StringBuffer do not overrides the equals() and so cannot use it.

Please review later the programs for testing performance of String and StringBuffer. The difference mutable and non-mutable is also shown by printing the hashcodes for the objects. By the way, hashcodes are code which codes to find the objects in the hash table. Hashcode is made some logic.

## Difference between StringBuffer and StringBuilder

They have some differences which are as follows.

1. StringBuffer is synchronous while the StringBuilder is not Synchronous.
2. StringBuffer is thread safe and StringBuilder is not thread safe.
3. StringBuffer is less efficient and StringBuilder is more efficient.

Please check the performance test between both.

**Note: When to use String, StringBuffer and StringBuilder**

1. String – it is immutable. So, you should use it in a program where the values of the string are not required to change.
2. StringBuilder – It is mutable but it is not thread safe and synchronous. So, it should be used in a program where the values of the string are changing because of program logic or any other reason but only be used by a single thread. This is because synchronous adds extra overhead and if not used will decrease the efficiency unnecessarily.
3. StringBuffer – It is mutable, thread safe and synchronous. So, it should be used in a program where the value of the string is changing because of program logic or any other reason and used by multiple threads. Since, its being synchronized ensures that the threads will use the stringBuffer in a consistent way.

**Note: What is thread safe. All its aspect.**

Multithreading is the concept where multiple threads are executed simultaneously. Then, it may also be possible that the threads running simultaneously may access the same resource (class, method, object, or field). Now, let’s assume that we have a method A() and two threads T1 and T2. T1 and T2 tries to access the method A() and will create some problem which make the program inconsistent (Check the example given below to understand how exactly can this make program inconsistent). So, we understand that multiple threads accessing a resource simultaneously can create a problem. Now, we have to solve the problem by making the program thread safe. When we make a program thread safe then multiple threads uses a resource simultaneously (not simultaneously literally) without making the program inconsistent. This actually happens by letting one thread to use a resource at a time while letting others to wait. This is the only way of avoiding the problem that is occurring. Thread safety is implemented by synchronization.

Example - Suppose we have a file and one thread opens a file, other thread is writing in the file and another thread closing the file. This all happening simultaneously will corrupt the data of the file.

## How to create an immutable class?

If you follow few rule then can create your own immutable class. These are the few rules.

1. Make your instance variable as final.
2. Use final keyword with your class too.
3. Do not give setter method.

Immutable classes can be made to make few things unchangeable in the project which may be used sometimes.

## Java ToString() Method

This method comes in existence when we think of representing an object as a string. The sole purpose of thinking about this is to print the value of the object by writing less code.

The best way to understand this is to first make a program which prints the value passed while creating an instance. For that you need to create a constructor and use the display method (or you can also write print statement in constructor itself). In the same program, literally print the object of the class then it will print the hascode. Now, instead of representing the object with hashcode, we can represent the object as string. This is done by overriding the toString() method whose return type is String. Also, that will be involving less code to represent the clause of the object that the usual way which is its second advantage. In the nutshell, to literally print an object as string and write less code to print the values, we can use toString() method. IT’S NOT ABOUT CONVERTING THE ARRAY TO STRING.

## StringTokenizer Class in Java

* Java.util.StringTokenizer is a class that allows you to break a string into tokens. Though it does not provide a facility to differentiate between numbers, quoted strings, identifiers and etc, but it is simple way for breaking string into tokens. Example of quoted string is “Hello”.
* StringTokenizer has three constructors which are StringTokennizer(String str) i.e. string tokenizer with specified string, StringTokenizer(String str, String delim) i.e. string tokenizer with specified string and delimiter and StringTokenizerI(String str, delime, return boolean) which has got additional Boolean return type which if true then the delimiter is taken as a token to be printed otherwise not.
* Now, to use this we have to make the instance and pass the string which we want to break. Then, use its method “hasmoretokens()” in the while loop and nextToken() inside the print statement. Check example for further clarification.

Note: you can check more methods later if want to learn more on this.

## Java String Methods(left out)

1. String contain() Method – This method search the sequence of char in a string i.e one string into other. If found then returns true otherwise false. We can even search any other string in the other. If the sequence is null then it will throw the NullPointerException.

Note: we call character sequence to this: String s = “Rishav”;.

1. String Format() Method –this method is used to format the string in java by giving locale, format and argument. It can throw NullpointerException if format in null and illigalFormatException if format is illegal or impossible. Also, we have different formatting style for int, float, char etc which is out of scope right now.

Note: In case of printing the index of the searched string the index returned will be the first appeared character with searched string in the main string. Check javatpoint example for this.

1. String IndexOf() method – This method is used to find the index of character or sequence of char in a string. It can receive from Index parameter which takes the index value to start the search start inside the string.
2. String isEmpty() Method – This method is use to check if the string is empty or not.
3. String join() Method – This method is used to join multiple string but with delimiter in between. Two parameters are passed. First is the delimiter type and other is the strings. Though it will seem as in you can put one string but you can put multiple separated by comma. If we have to join an existing string then pass it as an argument with double quotes.

Note: I am not able to run this method on eclipse kepler. Also, in oracle docs it is mention 1.8 java so dose this means that it can be used from java 1.8? Also, I can see the error for this method. Please find out on this later.

1. String lastIndexOf() Method – this method is used to print the last index of the parameter given. It can be given a character to find the index of the place where it is last appeared in the string, a string can also be given accompanied by fromindex parameter which defined the start index point of the search.
2. Java String replaceAll() Method – this method is use to replace a regular expression wherever found in a string with a sequence of chars. Both are passed as the parameter. Note: this [\\s](file:///\\s) is a single white space and [\\s](file:///\\s)+ more than one white spaces.
3. String split() method – This method is used to split a string against given regular expression and return a char array. It takes two parameters which are regex which is the expression makes basis of split and int limit which is limits the number of string in an array. If it is zero than no string, one than as it is one part and 2 then in two parts. Also, we have to create an string array and put all the parts in that and print it with the help of for loop.

Note: This method returns string array and thus we cannot simply print it but we have to print it like we print an array.

1. String toCharArray() Method – it is a method used to convert a string to char array. To implement it, create a string and convert it into tocharArray() method and then print it through a for loop.

# Java Regex

(Over to java oracle docs: <https://docs.oracle.com/javase/tutorial/essential/regex/>)

Java regular expression (Regex) is used to define String Pattern that can be used for searching and manipulating a text. Java provides java.util.regex package for pattern matching with regular expression. It is very useful in defining constraint on password and email verification. This java.util.regex has mainly got one interface MatchResult interface and three classes which are Matcher, Pattern and PatternSyntaxEception class.

1. Matcher Class – This implement MatchResult interface. This is basically used to perform the match operation between the pattern and expression. Check all of its methods while Boolean Matches() being mostly used and solves the purpose of matching the regular expression with the pattern.
2. Java.util.regex.pattern Class – This class basically helps define a pattern to the regex engine. Let’s study some of it highly important methods.

* Pattern.matches(String regex, charSequence input) method – this method is a most simple way to searching a string in a text using regex. I have made a program and figured out that we call this method with its class. Pattern string and content are given as parameters. However, I still have to learn to build pattern. With this we can just search the single occurrence in the content (text) and case sensitive.
* Pattern.compile() method - For case insensitive and searching multiple occurrence, we have compile method of pattern class. While writing the code I observed that we call compile method with pattern class. The parameter passed in the compiler method are regex i.e. the pattern string and a CASE\_INSENSITIVE flag. We can use other flag for getting this done. We gets an instance of pattern class. But match making is still not complete. For this, we have to make a matcher instance.
* Pattern.matcher() method –this method is used to create an instance of matcher class. So, for that we use pattern instance with matcher method of the pattern class. Then, with the help of matches method we get the result. So, in the nutshell, to search a string (string pattern formed of regex) in the content (text) by the regex classes which are matches, compile, matcher.
* Pattern split() method – to split a text into multiple string based on delimiters (here based on regex), we can use split() method. Checkout the example. Firstly, the text and pattern were declared and initialized. Then, the pattern instance was created with the use of compiler method. Now, something new happens. An array of string is created which is feed by the spilt part of the text string. Then, we used a for loop in this way: “for(String temp : myString)” where myString is the array of spilt part and this means to get the myString value one by one till its last.

1. Java.util.regex.mather class – we have already leaned the way to create the instance of matcher class. Let’s understand some of its main methods.

* Matches() – this method is used to match the string against the text passed to pattern.matcher() method while creating matcher instance.
* lookingAt() – it is also similar to matches() but it search the String only at the beginning of the whole text.
* Find() – this is mainly used in the case of multiple occurrence search.
* Start() and end() – both these method are generally used with the find method to get the start and end index of the match being found using find() method.

**Writing Regex** –

So far, we got to know that the ways we have to match the pattern string with a text using regex. Now, we will look at the options we have to define the pattern string or regex.

1. **String literals** – String literal means string like “abc”, “123” or similar like this. If the same string is found not less or more than one time in the other string then result will be true otherwise false. Infect, if the regex contain only string then the other string should be identical. Check example “basicregexdemo”.
2. **Character String**–A character class matches a single character in the text against the multiple allowed characters in the character class. For example, Pattern.matches(“[pqr]”, “abcd”) it will give true if only one character is search against this and contained only one time. Below are all the character cl ass and its descriptions. This class search only for one character. @@@practice examples.

|  |  |  |
| --- | --- | --- |
| **No.** | **Character Class** | **Description** |
| 1 | [abc] | a, b, or c (simple class) |
| 2 | [^abc] | Any character except a, b, or c (negation) |
| 3 | [a-zA-Z] | a through z or A through Z, inclusive (range) |
| 4 | [a-d[m-p]] | a through d, or m through p: [a-dm-p] (union) |
| 5 | [a-z&&[def]] | d, e, or f (intersection) |
| 6 | [a-z&&[^bc]] | a through z, except for b and c: [ad-z] (subtraction) |
| 7 | [a-z&&[^m-p]] | a through z, and not m through p: [a-lq-z](subtraction) |

1. **Predefined character classes** – These are like a short code while writing the regex. \s is the white spaces while the[^\s] is characters except white spaces. \w is the word character which includes the alphabets, digit and one special character which is under score. \W is any character except the word character. \b is like an anchor and it matches a position that is called word boundary. Also, there is one meta character left which is a “.” (dot). This dot indicates any character. @@@practice example.

|  |  |
| --- | --- |
| **Regex** | **Description** |
| . | Any character (may or may not match terminator) |
| \d | Any digits, short of [0-9] |
| \D | Any non-digit, short for [^0-9] |
| \s | Any whitespace character, short for [\t\n\x0B\f\r] |
| \S | Any non-whitespace character, short for [^\s] |
| \w | Any word character, short for [a-zA-Z\_0-9] |
| \W | Any non-word character, short for [^\w] |
| \b | A word boundary |
| \B | A non word boundary |

1. Quantifiers – the quantifiers specifies the number of occurrence of a character.

|  |  |
| --- | --- |
| **Regex** | **Description** |
| X? | X occurs once or not at all |
| X+ | X occurs once or more times |
| X\* | X occurs zero or more times |
| X{n} | X occurs n times only |
| X{n,} | X occurs n or more times |
| X{y,z} | X occurs at least y times but less than z times (less than or equal to z times?) |

Moreover to the table above, quantifiers are of three types which are greedy (simple pattern), Reluctant ((simple patter)?) and Possessive ((Simple pattern)+). These are basically different in the approach they search the pattern. Check them in detail later if required.

**Summary**

We first have seen the way of checking a string against a regex pattern. We have found that matches(arguments) of pattern class is not the only way but the easiest way of doing the same. After this, we learned as how to make the regex pattern. For this, we have seen character class, predefined character class and quantifiers. With all these classes, we can create versatile patterns to be used against the input string. We can explore more on this at this link below.

<https://docs.oracle.com/javase/tutorial/essential/regex/>

Important link for practicing some of its examples:

[**http://www.tutorialspoint.com/javaexamples/java\_regular\_exp.htm**](http://www.tutorialspoint.com/javaexamples/java_regular_exp.htm)

# Exception Handling

## Exception Handling in Java

* The exception handling is a powerful mechanism to handle the run time errors and to maintain the normal flow of the application. Firstly, what is exception? Dictionary meaning of it is an abnormal situation while in java it is an event that disrupts the normal flow of the program. It is an object thrown at run time.
* The core advantage of exception handling is to maintain the normal flow of the application. In an application, code stop running from where an exception occurs.

Do You Know?

|  |
| --- |
| * What is the difference between checked and unchecked exceptions? * What happens behind the code int data=50/0;? * Why use multiple catch blocks? * Is there any possibility when finally block is not executed? – if system exits or the thread running try catch block kills then it will prevent final block to execute. * What is exception propagation? * What is the difference between throw and throws keyword? * What are the 4 rules for using exception handling with method overriding? |

* Well, exception has got hierarchy as well. But, this goes really out the scope as it is more for learning. Now, let’s look at different types of exceptions. The exceptions are basically of two types which are Checked and Unchecked Exceptions. The Unchecked Exception is also divided in two types which are Runtime Exception and Error.
* Checked exception – These exceptions are found at the compile time i.e. the compiler will found then while compiling the program. The super class of all checked exception is IOException. Some examples of checked exceptions are SQLException, DataAccessException, ClassNotFoundException etc.
* Unchecked exception – These are exception which are not found at compile time but will be found at runtime. These are of two categories, which are described below. All the Unchecked exceptions are the sub classes of the Runtime Exception.
  + 1. Runtime Exception – These are due the mistake of programmer. It may be because the programmer trying to access an element which does not exist or any other mistake that looks fine at compile time but will create an abnormal situation. All the runtime exception has a super class called as Runtime Exception class.
    2. Error – These are the exceptions that an application cannot deal with. It happens even when the coding is done correctly. One example of it may be when the JVM runs out of resource. The errors are catchable but typically an application will be closed until the problem is not dealt.
* Common scenarios where exception may occur – First, where a number is divided by zero causes airthmaticexception. Second, if we have null value in any variable (primitive or reference type) then NullPointerException will occur and that is because of length. Third, when wrong format of any value is used (like int in place of string). Fourth, when wrong value of index is entered (like negative or more than the size of array) then arrayOutOfTheBoundException will occur. Well, this is not it but we have other exceptions too.

**Note:**

1. Errors are of two type which are syntax error (compile time error) and runtime error (unchecked exception). The syntax error is because of typo error that we make while writing the program and hence we encounter them while compiling the program. The runtime errors are nothing but the unchecked exceptions that happens while program is running.
2. If an arithmetic Exception occurs, then we catch it by the print statement printing the “exception is caught” message. Well, this does not happen in actual application. We do write some remedial code in the catch block. So, do not think that what is the advantage we are just printing and the calculation is still not performed. This happens only in the sample program just to make us understand that the control goes over from try to catch block if exception occurs.
3. We can print exception message in three ways in java. First, by printing the “e” of exception e. Second, e.printStackTrace(). Also, do not put it inside print statement. Third, e.getMessage() can be used inside the print statement.
4. Parent class of exception and error is throwable class.
5. It is not possible to catch Errors by catch block. Error example can be OutOfMemory error or written like new Error()..
6. When exception comes then the info of exceotion then after exceptiob is occurred then finally block will be executed. Then, flow of program will stop.
7. If any variable is declared inside try block then it cannot be accessed outside the try block.
8. Default handler handles the exception when no catch block is written. This is provided by jVM.
9. If the exception comes at line written inside try block and caught by catch block then all the code written after catch (stack trace printed) or finally block will execute. Code inside the try block after the line where exception occurred and caught will not execute.
10. If the exception comes in the called method then that can be caught inside the catch block of the try block from which it is called.
11. Class not found exception occurs when class is not found when classloader tries to load it.
12. Init() cause method is used to find out chained exception code which actually caused the exception.

## Java Try Catch

* The code which is prone to give exceptions is written in try block which must be followed by catch or final block. These blocks are within the method. Check the syntax. Catch block is used to handle the exception and it must be used after the try block. We can use multiple catch block. Check the example for better understanding.
* Internal working of try catch block – It is pretty simple. When the exception is created, an object will be thrown. Now, either that will be handled or not handled. If handled, then rest of the code will execute otherwise JVM prints the exception description and will terminates.

## Multiple Catch Bloc`k

If you suspect multiple exceptions in the code written in the try block then you can use multiple catch blocks. But one exception comes at a time and handled at a time. Also, the catch blocks should be written from most specific to general.

## Nested Try Block

The concept of one try block into other is nested try block. The need arises when one part of a block may cause one exception while the entire block causes a different exception. Check the syntax. Try a program for better understanding.

## Finally block

This block is used to write the cleanup code. Now, what is cleanup code? cleanup code are the code written to properly close any file, database connection or anything else that was used for running the program but not required anymore. For example, this closes the database connection so that other users can easily make connection. For a user to make connection a sufficient number of open database connection should be closed. Finally block is always written even if the exceptions are handled or not which means if catch block is present or not. The only condition is to have a try block before a finally block. Another point of view to understand finally block is that it will definitely be executed if the try block in called. We cannot handle the exception in finally block and we cannot write it before the catch block as well.

Note: we already know that whatever happens except system.exit or JVM crash, finally block will always be executed. Also, when we have return statement in try and catch block then also finally block will be called and then return statement of try and catch will be executed. But, if finally block also have return statement then also finally block will be executed and try catch will not be executed. Though, writing return statement in finally is considered as very bad habit.

## Java Throw Keyword

At first glance, it did not make any sense to me. But yeah it did later. I was not able to understand it before because why we want to create an exception? It is abnormal condition that stops the program’s execution. Then, why do we need it? Exceptions are needed for implementing some business need related to the program. From the example given, suppose we have to build software for machine that allows or stops a person to vote according to their age. So, when you have the age below 18 then an exception should be thrown. The facility of throwing our own exception (custom exception) helps us to maintain some business requirements like age of the voter should be more than 18. If age is less than 18 then we can take some alternative actions in the catch block. In the nutshell, to implement some business requirement we need to throw self-made exception.

Now, Throw is a keyword use to explicitly throw an exception in java program which we will catch also. It is mainly to throw our own created exception called custom exception but it can also throw the predefined exceptions.

Note:

1. How to create your own exception? Well, for this, create a class giving the name you want to give to your exception. Then, extend the exception class. Then create a constructor and pass a string as parameter with the message (or any random short string). Lastly, make use of Super() function and pass the massage. @@@Practice Example.
2. Syntax for Throw keyword: “throw new AirthmaticException(“myMessage”)” or “throw new CustomExceptionName(“MyMessage”)”.
3. We can rethrow an exception by using throw keyword inside catch block.
4. Even after knowing that an exception must be catched, If we do not. Then, that’s called Ducking the exception.

## Exception Propagation

Exception Propagation is simple process where once an exception is thrown then the runtime system looks for a block containing exception handling code. The search starts from the block throws the exception to the block next (below) in the calling stack. For example, if a calls b, b calls c and c calls d. If d throws an exception then the compiler will look in the order d to c to b to a. Check for deep explanation: <http://stackoverflow.com/questions/10633664/what-is-exception-propagation>.

Note:

1. If sub classes of exception is catched after super classed then unreachable catch block error will be shown.
2. We can chain the exception by throwing one exception on another exception is already thrown.

## Throws Keyword

We have just studied throw keyword which is mainly used for throwing the custom exceptions. Though, we cannot throw the custom exceptions with “throw” keyword alone. We need to use throws keyword as well while declaring a method (not while declaring a class) which use throw keyword to throw a custom exception. Since, custom exceptions are not known by java that is why we need to use *throws* keyword for a custom Exception. If we throw a predefined exception by using throw keyword then throws is not required.

Sometimes, throws keyword is also used to indicate that the code in the method might throw certain exceptions. Therefore, if anyone use those methods can properly catch those exceptions. This is another view point for throws keyword.

Note: Also, try to understand the difference between try and catch and throw and throws. Try and catch is used to handle the exceptions with or without throw and throws. With throw and throws keyword, try and catch block will be used in the method that calls the method that throws the exception. @@@Practice Example.

## Difference between throw and throws

Below are some difference between throw and throws keyword.

* Throw is used to throw an exception while throws is used to declare an exception in any method.
* Checked exception cannot be propagated with throw *only* and it can be propagated with throws.
* Throw is followed by an instance and throws is followed by a class. (look at the syntax literally)
* Throw is used within a method while throws is used as a signature in declaring a method.
* You cannot throw multiple exceptions with throw but we can declare multiple exceptions with throws keyword.

## Difference between Final, Finally and Finalize

Below is the description exploiting the difference between each of them.

* Final – it is a keyword. Final is used to restrict a class, method or variable. Final class cannot be inherited, final method cannot be overridden and final variable cannot be changed.
* Finally – Finally is a block. It is used to write code to terminate the connection and file used to run a program.
* Finalize – Finalize is a method. Finalize method is called by Garbage collector when a garbage collector finds that there is no reference to an object.

## ExceptionHanding with methodOverridng in java

There are few rules for exception handling in case method overriding which are described below.

* Rule 1 – if the parent class does not declare an exception, subclass overridden method cannot declare a checked exception but can throw unchecked exception. I have checked it myself. It only let you throw unchecked exception.
* Rule 2 – If the superclass throws an exception (checked/unchecked) then overridden method of subclass can declare same(checked/unchecked respectively), subclass exception but cannot declare parent exception. Here parent exception is simple “exception e”. It is very simple. Try it.

## Java custom Exception

Exception created by user itself is called Custom Exception. It is very simple to create your own exception. Try it. Then you can use that exception in the same way we use the predefined exception. Throws and throw are for declaring and throwing the exceptions but we can handle them as well. Also, when we handle the exception then look at what is printed on the console. It is interesting.

Note: If we need to create a checked exception, then we need to extend Exception class while runtime class is extended to create a unchecked exception.

# Java Inner Class

## What is java Inner class?

A class inside is called as nested classes but we have different categories of nested classes. Nested classes are majorly classified into two categories which are static and non-static class. Non-Static nested class is called as inner class which is further classified into Member inner class, Anonymous inner class and local inner class. We will also study about the static nested classes.

The class that holds an inner class is called as outer class. Let’s understand the advantages of the inner classes.

* Inner class can access the outer class methods and variables even if they are private methods.
* Inner classes are used to create more readable and maintainable code because it logically group classes and interface at one place.
* It requires less code to write so it is more optimized code.

Do You Know

* What is the internal code generated by the compiler for member inner class ?
* What are the two ways to create annonymous inner class ?
* Can we access the non-final local variable inside the local inner class ?
* How to access the static nested class ?
* Can we define an interface within the class ?
* Can we define a class within the interface ?

## Member Inner class

A non-static class that is created inside a class but outside a method is called as Member inner classes. This is pretty simple. I just need to take care that the inner class used the main method of the outer class. Also, I should remember the syntax for creating the object of the inner class in the main method of the outer class to call its method. I have tried working with main method of the inner class but, firstly, you cannot create it static since your inner class in non-static and secondly, if you try to call any other method in the main method does not make much difference because ultimately main method of the outer class will be handling the program. Also, it can access even the private members of the outer class.

Java compiler creates two class files. The name of java inner class is “Outer$Inner”. The syntax for creating the object of inner class is “OuterClassName.InnerClassName in = out.new InnerClassName();” where “in” is the object of inner class and “out” is the object of outer class.

Note: We can use the members of outer class in inner class but we cannot call the member of the outer class by creating the object of inner class. @@@try Example.

## Anonymous Inner Class

Anonymous inner class does not have name. It is present inside the main method of the class to override the method of other class or interface. An object of the class whose method needs to be overridden is created inside the main() which is followed by a block which overrides the method and ends with a semicolon. That block is considered as Anonymous inner class. It is accessible only at a point where it is defined. It does not have a constructor since it does not have a name. Compiler creates an anonymous class and gives it a name for its reference. Check below for understanding. @@@try Example.

Note: We should understand that the main use of anonymous class is to override a method without extending the class containing that method.

|  |
| --- |
| Class B //class whose method needs to be overridden  {  **Public void** B1(); // method needs to be overridden. It can be abstract or concrete  } //class B ends  Class A  {  **public**static**void** main(String [] args) //main method created  {  B obj = **new** B() //object of the class whose method to be overridden  { //anonymous class begins  obj.B1(); //method overridden  }; //anonymous class ends  }  } |

## Local Inner Class

This is a type of inner class (which means non static in nature) which is declared inside a method of the outer class. Though for calling a method of the local inner class, we must only create instance of local inner class inside the method in which it is created but outside itself (local inner class). Local inner class cannot be invoked outside the method in which it created. Check the example and practice some. We cannot declare local inner class as public. @@@Try Example.

Random Note: when we try to print null from print statement then we get compile time error.

## Static Nested class

Static nested classes are the classes which are inside any other class and declared as static. It cannot access the non-static methods and variable of the outer class. Though it has the advantage of being called by the outer class object but can only access static member of the class. Also, to call the methods of the static class it should be given proper syntax. The syntax is: OuterClass.InnerClass obj = new OuterClass.InnerClass(). Use this object obj to use call the static inner class methods. Also, if the method of inner class is static as well then syntax for calling it will be “outerClass.InnerClass.Method();”.@@@Try Example.

## Java Nested Interface

An interface that is declared inside any class or other interface is called as Nested Interface. A class which implements an interface by-default implements its inner interface as well. It has to define the method of the inner interface methods but for calling those methods from the implementing class, a different syntax is given than the usual. The syntax is: “OuterInterface.InnerInterface obj = new ClassName();”. The same happens when an interface is nested inside a class. We have seen interface inside a class and an interface and class inside a class but can we nest a class inside an interface? Yes, we can. @@@Try Example.

# Java Multithreading

## What is multithreading?

Multithreading in java is the process of executing multiple threads simultaneously. Threads are the smallest unit of processing. The core advantage of multithreading is that the threads uses the same memory location and separate memory is not allocated to each thread. Context switching between thread takes place in less time than the process where context switching is switching from one thread or process to other without any conflict.

* Multitasking is a process of carrying out multiple tasks simultaneously. It can be carried out in two ways which are multiprocessing and multithreading. Multiprocessing is also a similar concept but it is simultaneous execution of multiple process rather than threads. In multi-process, each process gets its own memory space while each thread uses the same memory address in multithreading. Process is heavy weight, compared to multithreading, since it is bigger unit. Also, the cost of communication between threads is lower than processes. Lastly, the context switching is fast between multiple threads than multiple processes.
* Thread in detail – A thread is a lightweight sub process. It is the smallest unit of processing. They are independent and if error comes in one thread then other thread is not affected. It shared a common memory. So, basically, an OS may have multiple processes while a process have multiple threads. In thread safe, threads use memory one by one through synchronization.

Do You Know

* How to perform two tasks by two threads ?
* How to perform multithreading by annonymous class ?
* What is the Thread Schedular and what is the difference between preemptive scheduling and time slicing ?
* What happens if we start a thread twice ?
* What happens if we call the run() method instead of start() method ?
* What is the purpose of join method ?
* Why JVM terminates the daemon thread if there is no user threads remaining ?
* What is the shutdown hook?
* What is garbage collection ?
* What is the purpose of finalize() method ?
* What does gc() method ?
* What is synchronization and why use synchronization ?
* What is the difference between synchronized method and synchronized block ?
* What are the two ways to perform static synchronization ?
* What is deadlock and when it can occur ?
* What is interthread-communication or cooperation ?

## Life Cycle of thread

A thread can have five states in its life cycle which are called as threads states. These are New, Runnable, Running, Non-runnable and terminated. The thread will be in new state when an instance of a thread class is created but only before the start() is called. Thread is in runnable state when its start() method is invoked, but the thread scheduler has not selected the thread as the running thread. The thread is in the running state when the scheduler selects the thread. The thread comes in non-runnable state when the thread is still alive but is currently not eligible to run due to any reasons like waiting, sleeping or blocked. When the thread is terminated or become dead is known to be terminated.

## How to create a thread?

There are two ways to create a thread. It can be by extending the thread class or by implementing the runnable interface explicitly.

* By thread class – what is a thread class? A thread class contains the constructor and method to create and perform operations on a thread. The thread class extends the object class and implements the runnable interface where object class is the top most class in java and by default extended by every other class in java. Check the commonly used constructors and methods of thread class while few are thread(), thread(runnable r, String s), run(), start(), sleep(), suspend(), resume(), stop() etc. Start() method is used with the run() method of the class extending the thread class (same or the other). Also, the run() method will contain the working of the thread. Also, we create the object of the class extending the thread class and calls the method (Start()) from thread class. So, the compiler automatically converts the extending class object to thread class object through thread class constructor.
* By implementing the runnable interface – In this method, we implement the runnable interface and the entire program is same as by extending the thread class except that we have to create a thread object by casting from the implementing class object. We need to create the thread class object for sure here since we have to call its method start or other while when we extend thread class then its not mandatory since we can access member of parent class from subclass without creating parent object. Try one example for better understanding. @@@Try Example.

Note:

1. main thread always start the execution of the program and it is created by JVM.
2. Thread Class has two constructors. First, without any argument constructor while second is Runnable and String parameters constructor.
3. If we are creating thread by extending thread then also we can create class object and then pass it to thread. It will not give compilation error.
4. Thread Class implements runnable interface.
5. If a thread is printed like this “System.out.println(Thread);”, then we get output like Thread[ThreadName, Priority, TypeOfThread].
6. If run method is not present while start method is called then compilation error will come provided thread is created by implementing the runnable interface.
7. When a thread calls wait then its on sleep by giving its lock as well. Now, at this point in time, if we try to check if that thread is alive through isAlive() method then we will get exception. I guess this is because the thread is neither running nor terminated.

## Thread Scheduler in Java

Thread scheduler as the name suggest schedules that when to run which thread. It is a part of JVM and selects a runnable thread to run. Only one thread can run in a process (while multiple threads can run of different process). The thread scheduler mainly uses preemptive and scheduling or time slicing. In preemptive scheduling, the thread with the highest priority is executed first until it enters the waiting or dead state or even higher priority thread becomes runnable. In time slicing, a task executes for a certain time slice and then again another thread is selected for a slice of time on the basis of priority and other factors.

## Sleep Method in Java

Sleep() method is a thread class method which is used to sleep a thread for a specified amount of time. It has two implementation which are sleep(long miliseconds) or sleep(long miliseconds, int nanos). Try an example, the sleep method written inside the run method and then run method is called by the start() method. When a thread sleep the scheduler picks up the other thread and then runs the other thread. Also, remember to catch the exception thrown by the sleep method.

Note:

1. Yield moves a thread to state which I think is runnable state.
2. We cannot override wait method of object class.
3. We know that when sleep is called in the run method then thread currently using run method goes on sleep. What will in a situation where sleep method is called outside the run method? This will make all threads of that class to sleep for the specified time. Hence, isAlive() will return false. Since, the thread is temperorily suspended so false will be returned.
4. Difference in Yield, Join and sleep  
   **yield()** method pauses the currently executing thread temporarily for giving a chance to the remaining waiting threads of the same priority to execute. If there is no waiting thread or all the waiting threads have a lower priority then the same thread will continue its execution. The yielded thread when it will get the chance for execution is decided by the thread scheduler whose behavior is vendor dependent.

**join()** If any executing thread t1 calls join() on t2 i.e; t2.join() immediately t1 will enter into waiting state until t2 completes its execution.

**sleep()** Based on our requirement we can make a thread to be in sleeping state for a specified period of time (hope not much explanation required for our favorite method).

## Start a thread twice?

No! We cannot start a thread once it is started. If we try to do so then we will receive an illegalThreadStateException. The thread will run for the first time but will throw the exception on the second time. Though, we have to understand that the class extending the thread class and having the run() method will be one thread for one particular object created while if other object is created for the same extending class then it will considered as another thread. This can be checked by running one object twice and running two different object of an extending class.

Note: This happens because the thread once started comes in running state and cannot be started again because it is not be runnable state.

## What if we call run() method instead of start()?

Each thread will start in different call stack where the call stack is the data structure that stores the information of current active thread (program). This mean when the run() is called in the main method, then the run() method goes to the current stack instead of getting to the beginning of the new call stack. Well, recall the program where you have used the sleep() function. There we have called the start method and hence the sleep of first thread gives the active state to other thread and the other threads starts executing until the first is sleeping. Now, in the same example, if run() method was called then sleep would have worked and run() will be executed by the first object with sleep method and once it is completely done then only the run method with the other object will be executed. This means, the object will behave as the simple object, not like the thread object. Check examples for better understanding.

Note: you make object of class and call the start() method. Then we get the functioning of a thread but if you call run() method then the object of class will behave as simple object.

## Join() method

Join() method waits for a currently running thread to die so that the joined threads can be executed. This method waits until the thread on which it is called does not terminates. So, in the example given in javaTpoint tutorial, the join method called t1 by “t1.join()”. So, t1 executes first and the other threads t2 and t3 will execute only once t1 is completely executed. You can check the program. Also, we have an option to specify the time until which join should wait the t1 method for execution of t2 and t3. In the nutshell, join() method is used to wait until the thread does not completes its execution to which it is called and we can also specify the time to which the method should wait. IsAlive() method is also used for similar purpose and it returns if the method is active or not.

Note:

1. while implement this, make sure that thread which you want to execute the alone i.e. next thread thread waits for its execution is start() before it is joined. First, start() is called and the join the next thread automatically joins.
2. Join method must be placed inside a try catch block otherwise it gives InterruptionException.

## Naming a Thread

The thread class provides the methods to change or get the current name of the thread. By default the names of the thread starts from 0 to n as the number of threads increases of an extending class. We can use the “objectName.getName()” syntax to print the name of the thread. We can change the name of the string by “objectName.setName()”. Then, we can check by printing the name again.

We have a method called currentThread() in the thread class through which we can get the reference of the currently executing thread. Remember that it returns the reference of the thread but not the name. So, for the name you have to use “Thread.currentThread().getName()”.

## Thread priority

Every thread has a priority. The priority is scaled from 1 to 10. In most of the cases scheduler executes the threads according to their priority but not always. It is dependent of JVM specification that which scheduling it chooses. Three constants are defined in the thread classes which are “public static int MIN\_PRIORITY”, “public static int NORM\_PRIORITY” and “public static int MAX\_PRIORITY”. Min is 1, normal is 5 which is default priority as well and max is 10. We can get the priority of any thread by getPriority() and set the priority by setPriority(). We can set the priority by simply writing the number from 1 to 10 or by “Thread.MAX/MIN\_PRIORITY”. By putting value of priority less than 1 and more than 10 gives exception.

## Daemon Thread

* This thread is used to provide service to the user threads and only fulfill the background tasks. JVM terminates all the daemon thread as soon as all the user class dies. It is of lower priority thread. We have to understand the way to check if the thread is daemon thread and to set a thread as deamon thread. For checking if a thread is daemon or not, we have method called isDaemon(). For setting a thread as daemon, we have a method called setDaemon(). Try an example for better understanding.
* If we want to make user thread as daemon thread. Then, we can only do it if the thread is not yet started. We will receive an exception even if we try to do so. See, if a thread is not daemon then it will definitely be user thread which means a simple thread. So, when to call a user thread by an object and then try to change it to daemon thread. We will receive an exception. Exception will be illigalThreadStateExcepion.

Note: Main thread and deamon threads are created by JVM.

## Java Thread Pool

Thread pool contains the equal sized threads which are waiting for the job and reused many times. A thread is assigned a work by the service provider and then after work completion it again comes back to the thread pool to get a new job. It saves time since avoid it requirement of creating a new thread. One example of it is its use in the servlets and JSP. Servlet and JSP container creates a thread pool to handle the request.

To demonstrate that each thread of the pool handles a single task and then, if all the threads are busy in some task then the new task will be taken when any of the thread becomes free executing the previous task given. Then check: <http://www.journaldev.com/1069/java-thread-pool-example-using-executors-and-threadpoolexecutor>. In this, a class extending thread class has run method which has print statement printing the current thread name and a message (message is actually indicating the current task which was given a parameter to one of the method and used in the run method). Then, a method is made to take a parameter of current task and that is used in the run() method. Another method is made to provide sleep of some time. This method is also used in run() method. A thread pool class is made wh0ere a thread pool of 5 five thread was created (syntax: ExecutorService executor = Executors.newFixedThreadPool(5)). Now then, we will create a for loop that runs 10 times and we will execute the current task (syntax: executor.execute(worker) where worker is the object of thread class). Also, while printing the output of the thread class each time, we will print the task number also. This way, we will get to know the thread number which performs which task and in which order. Also, the concept of thread pool will be clear.Try the program of thread later when get time. @@@Try Example.

## Thread Group in Java

The thread group is the set of threads. Java.lang.ThreadGroup includes many groups of threads where each group has a parent except the initial thread group. Check the constructors and methods useful in this class. Now, let’s see the way to create a thread group. Syntax: “ThreadGroup groupName = new ThreadGroup(“Group A”);” “Thread t1 = new Thread(grouypName, new className, “threadName”)”. This way we can first create a group of threads and then we can, kind of, insert the thread in the group. The advantage now is that we can apply some commands in one shot to all the threads in the group by applying it on the group. For example, we can use the getName() method on the thread group to know the names of each thread.

Note: “System.out.println()” this gives an extra line. @@@Try Example.

## Shutdown Hook

Shutdown hook is used to clean up the resources before the JVM is closed. When we exit from the command prompt pressing cntl c then user logs off and JVM starts to shut down. So, before shutdown of the JVM, shutdown hook works to clean up the necessary resource. Also, we can add some code that we would like to execute just before JVM shutsdown. For this, we have to create a thread which contains the code we want to execute. Then, add that thread as a shutdown hook through addShutDownHook(runnable r) method. This method is of runtime class. So, we have to create a runtime class instance and then use this method. Also, we can halt the execution of the added shutdown hook. Check the syntax. We understood the concept of shutdown hook and just check the actual program later when you get time.

## Performing Single/multiple Task(s) through Multiple Threads

This section deals with performing same task with multiple threads and multiple tasks with multiple methods. For this, we must have to understand a concept that each thread runs on a separate callstack. Now, suppose we have a class with a run method. We have created two thread instances and called the start() method from both the object. Then, both threads will work on the same method. That will happen one by one (I think when the class is thread safe). Now, we have to run multiple tasks by multiple threads. So, we have created two classes having a run method. We have created third class and created threads relating to both the class and called the start method. The same is shown through the anonymous classes but that is definitely out of scope now.

Note: This is actually an example of multitasking using multithreading. @@@Try Example

## Garbage collection

An object which is not reference currently unnecessarily takes the space in the memory and reduces the efficiency. So, all unreferenced object must be removed or destroyed. An object can be unreferenced by nulling, assigning new reference to it, anonymous object (new Employee()) etc. In C language we have free() method and we have Delete() method in C++ language. However, in java we have even better feature of Garbage Collector which collects all the unreferenced objects. It is performed by Daemon threads. These daemon thread calls the finalize() method every time before collecting the object. Therefore GC has two main advantages which highly efficient and automatic. But GC is only applicable on collecting those unused objects which are created through new keyword. We have got GC() method of system class which is used to invoke the garbage collector explicitly. But there is no guarantee that garbage collection occurs on request and we cannot force it.

Note: Object in java can be created in different ways. I have checked some 5-6 ways of creating object. Though, many of the ways use new keyword. So, the other ways does not come under java garbage collector.

## Java Runtime Class

This class is used to communicate with the java runtime environment. This class has got some important method like invoking GC, get total and free memory etc. There is only one instance of this class available from one class. Check the example later when get time. Currently, it is out of scope.

Note:

1. ThreadLocal is a keyword we can use with thread so the each thread accesing the resource should have their own copy of the variable. Thus, giving synchronization at variable level.
2. Thread leak happens when an application does not releases the reference of a thread and those thread were not garbage collected.

# Java Synchronization

## Synchronization in Java

Synchronization in java is capability to control the access of multiple threads to any shared resource. Actually, when multiple threads try to access a shared resource then the problem of thread interference and inconsistency occurs. So, we have thread safety which is done through synchronization and letting only one thread to use the shared resource at a time. Synchronization is of two types which are process sync and thread sync. We will discuss thread sync here. Thread sync is further classified into Mutual Exclusive and Cooperation. In examples, we have given an idea when multiple threads works on a single object (object of task) then the inconsistency occurs. So, we have used the keyword *Synchronized* in the method signature with the class whose object was worked by threads. Except this, everything else looks same as the normal program. Check the examples in details later when have time.

**Concept of lock** – Every object has a lock or monitor. A thread that needs consistent access of the object will require having the lock of the object. Once the thread is done then it should release the lock.

Mutual Exclusive - This helps to prevent the thread interference when multiple threads try to use a shared resource. This can be done in three ways which are synchronized block, synchronized method and by static synchronization.

**Synchronization Method** – Suppose you have a class which performs some task. Now, you have created an object of that class. Then, you have created multiple threads which try to access the same object. So, for maintaining the consistency we will use the keyword *Synchronized* with the method whose object is shared as the shared resource. @@@Try Example

Note: The synchronization done through putting synchronyzed keyword with method signature, actually behind the scene capture the monitor of a particular resource. Now, when other multiple method of a class is using synchronization then the monitor used will be same. The basis is same in all the synchronized method of a class, so if a thread occupies the lock of one monitor then other thread cannot even access the other synchronized method of the same class. Obviously, non synchronized method can be called by any number of threads any time.

## Synchronized Block

This is similar to synchronized method. If you want synchronization for some part of the code in the class then you can create a block like this “Synchronized(this) { code to be synchronized}”. It will work same as synchronized method. Can we have a block inside a method? Yes. Infect, in the example itself the block is inside the method. Synchronized keyword was used with block instead. There is some difference in rules for static and non static block. Why not try and check. @@@Try Example

Note: the synchronization block is syntax is synchronization(this){code}. This can be any object while generally this keyword.

## Static Synchronization

Lets say we have two classes, class A and class B. Now, class a has a method named methodA and class B has a method named methodB. We have two threads, T1 and T2, of class A which try to access methodA and threads T3 and T4 which try to access method. Now, thread safety among T1 and T2 for method and T3 and T4 for method can be done through using synchronized eyword with method signature of methodA and methodB. Imagine a situation, methodB class method. Then, either of thread T3 or T4 will try to access methodA. In this scenario, we do have thread safety for methodA. Now, for this if we do static synchronization then we can thread safe this situation as well and thread T3 or T4 has to wait for T1 or T2 to finish. The lock in static synchronization is on class level, not on object level. The syntax of static synchronization: public static synchronized void add(int value){method body}.

Note: How actually synchronization makes a piece of code thread safe? Actually, when a thread tries to access any ynchornized block then it checks in the heap to give the most updated code to the currently accessing thread. This is one aspect.

## Deadlock in Java

It is a situation where we have two threads A and B. A is waiting for the object which is acquired by B and B is waiting for the object acquired by A. This situation is called as Deadlock. Check the example.

Note:

1. Starvation is a situation when a thread not able to access a target resource for a long time.
2. Thread.joijn could definitely be a potential reason for thread deadlock situation.

## Inter Thread Communication or Co-operation in Java

Inter thread communication means when synchronized threads talk to each other. There are some situations where threads exchange information. This is similar to process communication. The thread pauses in some stage of execution of task. It releases the lock as well. Then, some other thread gets the lock and begins its task on the same object. The first thread will wait until it is not notified by second thread or to a specified time. Let’s see the important method of the object class related to co-operation. Also, these methods are of object class, not of thread class this is because it acts on lock which is a part of a lock.

1. Wait() – It takes the current working thread to a wait state. It will be in wait state until other thread does not notify using notif() and notifyAll() method.
2. Notify() – This method is used to notify the first thread by the second thread that it is done with the object. If multiple threads are waiting to be notified then only one thread will be notified and its selection will be random.
3. NotifyAll() – This wakes up all the threads which are waiting for the object.

**Difference between wait() and Sleep() –**

1. Wait() releases the lock while sleep() does not.
2. Wait relates to object class while sleep relates to thread class.
3. Wait is a non-static method while sleep is a static method.

Check the example carefully later.

## Interrupting a thread

If a thread is in wait or sleep state then calling the interrupt method will force the thread to come out of sleep or wait. It will throw an exception InterruptedException. The thread will stop working if exception is not handled though will work soon as the exception is handled. If a thread is not in wait or sleep state then the thread will behave normally the interrupted flag will be set true. We have three methods in the thread class related to interruption which are interrupt(), interrupted() and IsInterrupt(). Interrupt() is to interrupt a thread. Interrupted() method is used to set the interrupt flag true which can be checked by IsInterrupt() method.

## Reentrant Monitor in Java

In java, Java monitor (lock) is reentrant which means it can use the same monitor for different synchronization method if method is called from the method. This is the situation when one method is called from another method (synchronized method), but the lock used will be same as if it is called directly. So, let’s say a method is called directly and through another method then thread will become unsafe if the same lock is not used. But lock are reentrant and hence we are safe. This eliminates the chances of single deadlock.

# Java I/O

## Input and Output

Java I/O is used for processing the input and producing the output. Java uses the concept of stream to make I/O operation faster. Java.io package contains all the classes which are required for input and output operation.

* Concept of stream – Stream is a sequence of data and the data is in the form of byte. They are of three types which are system.out (output stream), system.in (input stream) and system.err (standard error stream). For example, we use very frequently, system.out.println(“message”) is used to print the output. Similarly, system.err.println(“error message”) is for printing error message at the console. Also, for taking an input from the console we use input stream with syntax: “System.in.println((input data type)variable)”. For example, “System.in.println((char)i)” where char is the data type which is to be taken and input will become value for variable i.
* Output Stream – As the name suggest, it is a output stream of data which is used to place the data in the output destinations like a file, an array, peripheral, device or socket. While the input stream is to take the data from such sources. The process of taking the data from the source to the java application and taking data from java application to destinations is done through input and output streams respectively.
* Output stream class – it is an abstract class which is extended by each class using the output stream. It uses some methods like write(int), write(byte[]), flush(), close() [check meaning for all the before mentioned methods]. Also, there are different types of output streams which are fileOutputStream, ByteArrayOutputStream and etc. check all the types in later if required.
* Input Stream class – it is also an abstract class which is super class to all the classes using the input stream. Also, these have different methods used for different features. Check later if needed. Similar to output stream, it is also of different kind. Check later if needed.

## File InputStream and OutputStream

In simple words, reading data or writing data in a file is done through the Input and output file streams. It is also called as file handling.

* File output stream – This is used for writing byte code in any destination file. Firstly, we create an object of fileOutputStream. Then, we create a string and convert it into byte code to write it into file through write() method (since we cannot write directly from string and check what all type of data type we can write through write method). Then close the FileOutputStream by close().
* File Input Stream – it is used to take input from a file and print it on the console as output. Firstly, create the object for the FileInputStream. Now, to read the data (byte code for example) and then, print the char value of the code. The last entry of stream is -1 which is by default so we have used this logic to print till the last value of the text file. As we know that, these streams contain the stream of byte code. So whatever contained in your file will be received in byte code. That is why we have to convert it into char while printing which is done as the syntax in the print statement. If you get error that file not found than to save file at location and file path while declaring the object. Also, make sure that the file is saved in the format declared. Well, try the example for better understanding.
* Reading the data from one file and writing it in another file

In this, instead of using the print statement while reading the data from one file we write the content two another file (create the output object beforehand). Also, you can then print the data of the file to which the data is written. Try example for better understanding.

## ByteArrayOutputStream

ByteArrayOutput Streams are used to write the data into the byte array buffer and then used to write into multiple file. Its size increases automatically as required but initially it is 32 bytes which is changeable. Close() method does not make any affect over ByteArrayOutputStream. Here, we use write() method towrite into ByteArrayOutputStream and writeTo() is used to write into FileOutputStream(). Check the example for better understanding.

## SequenceInputStream

This class is used to read the data from multiple files. It has got two constructors. First, SequenceInputStream(InputStream1, InputStream2) and second, SequenceInputStream(Enumeration e). First is used for reading data from two files simultaneously and other is used to read the data of an enumeration. Try the example for better understanding. Also, practice for reading the data from two files and instead of printing write it in some third file.

**Must Read** - I have used the sequence stream for reading the data from two files and wrote it in a third file. So for writing data into third file, I had made an object of fileoutputstream pointing third file. Therefore, this way I wrote data. Now, when I have to print the data of the third file, I make another object which is of file input stream to read data. I made that object and tried reading the data but got a message at console that file not found. Now, is it because the object that I made for reading the third file points at another location? But when we simply write something is newly created file and then it is printed. Then, it prints the message even when the different objects used for writing and reading (you may have to use two different object for writing and reading). Well, as of now I moving ahead. But later if required then study the topic in detail. I think I must be doing some mistake that time but it should print the msg. verify this.

## Java BufferedOutput/InputStream

Java bufferoutputStream uses an internal buffer rather than storing the data directly in the stream. It increases the efficiency and hence the performance of the system becomes faster. Buffers are output as well as input type. What I have to do it just to connect the buffer with the respective stream. Else they are all same as file output and input streams. We can use buffer with the byteArray to read multiple array and sequenceinputstream for reading data from multiple file.

Note: Make sure to close the object of these classes otherwise program won’t print the values every time.

## Java FileWriter and FileReader (file handling in java)

These are for writing and reading data from a text file. They can read and write the character oriented data (unlike byte oriented data in other previous streams).

* Java FileWriter class – These are used to write the character oriented data into a file. These are simple, as we do not even have to create string and convert it into byte. But we have to directly insert string data by using write() method. Check example for better understanding.
* Java FileReader class – Java is used to read the character oriented data from a file. This is exactly same as streams for reading the data because the data gets internally converted from byte to character. So the difference cannot be made by codes. Check the example for better understanding.

Note:

1. While trying an example randomly, I have given the name of the class as “FileWriter” which itself a class and will be treated as the keyword. So, I was not able to import the FileWriter class. Therefore, we should not use the keyword as the class name.
2. Make sure to close the object of these classes otherwise program won’t print the values every time.

## CharArrayWriter Class

This class is used to write the character oriented data into multiple files. This class implements the appendable interface. Its buffer size increases automatically whenever required. Close() method does not have any affect. This is pretty simple and similar to ByteArrayWriter class but just handles the character oriented data instead of byte oriented data.

## Reading data from keyboard

We know how to display the data but never looked at how to take data entered at the keyboard. There are few methods for that which are described below.

* **InputStreamReader** – This class is used to read data from the keyboard. It first connects to the keyboard and then converts the byte code to characters. BufferReader class is another class which reads the data line by line by readline() method where buffered reader is nothing but a buffer which we connect with the inputStreamreader. Check another example of it which stops taking the input until the user writes stop. Here, do check the while loop condition. It is not working dude!Both logic and syntax is fine but do not know why it is not working.
* **Java Console Class**– This class is used to read the input from the console. Java.io.console class is attached with the system internally. It is also used very much to read the password and have different method for that which is readPassword() which is not displayed at the console. Also, the instance or object of the console class is created differently. Its example is giving NullpointerException. It is an eclipse bugs.
* **Java Scanner class**– java scanner class break the input through delimiters that is whitespace by default. Java scanner class extends object class and implements iterator and closeable interfaces. Check the methods of the scanner class which all the related to the tokens (parts of string). That method we have used during cognizant training. Also, look at the to use the delimiter by useDelimeter.

## PrintStream Class

PrintStream adds functionality to an outputStream which is to print representation of various data values conveniently. Also, it can be used to print the data at output console if “System.out” is mentioned as the parameter in printStream. It adds to more functionality. First, it does not throw the IOException (most of its methods), unlike outputStreams. Second, it provides an automatic flush. The code is converted into byte internally according to the OS dependent encoding schemes. Thus, it becomes little OS dependent. I have tried an example, where we have to create an output stream and connect with the printStream. Then, we can print the primitive data type values as text through the methods of the printStream class. Also, check the methods of the printStream class. Errors can be checked by calling checkError() method.

**PrintStream VS PrintWriter**

From java 1.0, we have FileOutputStream included in java which is super class of printStream. While we have FileWriter which is included in java1.2, which is super class of PrintWriter. Both these classes are used to print the data into outputstreams and on console as well. However, both are little different from other. First, PrintStream converts the data internally into byte through OS encoding scheme and thus makes itself a bit OS dependent. It can trouble while shifting the application to other platform. On the other hand, printWriter does not do any sort of conversion and prints the data directly as a sequence of characters. We can specify the encoding scheme if needed.

## Compressed and Uncompressed File

DeflaterOutputStream and InflaterInputStream are used to compress and un-compress a file in the deflate compression format. Let’s see how we do it.

* DeflaterOutputStream – This class is used to compress a file into deflate compression format. Suppose, we have a simple file. We can use .java or .txt file to deflate and inflate. We first read the data from it (so have to create a fileinputstream). Then, we will create a write object (which means fileoutputstream object) which then be connected with the deflaterOutputStream Object. So, we will use deflateroutputstream the same way we use the buffer with the fileoutputStream.
* While deflate, we need file (.txt or .java) which need to be deflated. For this, create a file in the workspace location of the class. You know where it is saved.
* InflaterInputstream – this is used to uncompress a compressed file. We also use this in the same way we use buffer with the fileinputstream. Firstly, read the content of the compressed file and create an object to write something into the new file. Then, create an inflaterInputStream object and connect it with the fileoutputstream object. Then, write the content of compressed file into uncompressed file using the object of inflaterInputStream. Also, then flush the inflater stream.

## PipedInputStream and PipedOutputStream Class

These streams are used to read and write the data simultaneously. This is done by connecting both the streams. Threads in used here. Firstly, create the object for both and connect them. Then, create thread t1 to write the data through the object of the pipedoutputstream. Then, create another thread and read the content through the object of pipedinputstream. Then, start both the thread. Check the example for better understanding. We have used anonymous classes for writing the threads. I have tried the example where we create the pipedinput and output streams and connected them. Then, we have created two threads and inserted the numbers through output thread and read the numbers from the input thread which are printed. Though, I was getting IOexception and some unknow special characters were shown.

# Java Networking

## Concept of Networking

Networking is a concept of connecting multiple computing devices to share resources. In java, java socket programming is used to provide facility to share resources between computing devices. The typical advantage of networking is the ability to share resources and centralize software management (centralize as a hub controls all the nodes in a network. Also, can check more on this) Few important terminologies in networking are given below.

* 1. IP Address – It is a unique number to identify a computing device in a network. It is a logical address which may differ with the physical address. Logical address is the address which is generated by CPU where a physical address is the one that it actually available on the memory.
  2. Protocol – It is a set of rules which are responsible for setting up the communication between computing devices.
  3. Port Number – This is used to identify the application in a computing device to which the message needs to be sent.
  4. MAC Address – this is a unique identifier to NIC (Network Interface Controller). A computing device may have multiple NICs that connect it to multiple networks but each NIC have unique MAC Address. PS: It is media access control address. This is used in Ethernet type of networks. Ethernet are LAN type of network and comparatively very small than internet.
  5. Connection oriented and connection less protocol – Connection oriented is where the notification is sent by the receiver and hence it is slow but reliable. On the other side, connection less is opposite to it.
  6. Socket – It is the end point between the communicating computing devices.

Note – Check what physical and virtual (logical) memories are. This is an operating system tutorial topic.

## Socket Programming

The basic idea of socket programming is to connect two computers in a network through java programming. Client creates a socket and tries to connect it with the server (it knows the host name and the port number of the server), in term to make server connection. On the server side, server listens to the socket for any request by the client for connection. Once the connection is made, server creates an object of socket and can share resources. This is the concept of socket programming. Java.net.socket is a class that is used to create socket and java.net.ServerSocket gives a mechanism to listen to client socket and establish a connection. Let’s look at the steps involved in the resource sharing.

1. The server instantiate the serverSocket class object, denoting which port number to which the communication occurs.
2. Server invokes the server accept() method of the server socket class. This method waits until the client socket is not connected. A server object is created by invoking the accept() method because it returns the object of server.
3. At the client side, client instantiate the socket class object specifying the server name and the port number at which the communication will occur.
4. The constructor of the socket class tries to connect the client with the server at the specified port number. Then, the client has a socket object that can communicate with the server.
5. At the server side, server sends a socket reference to the client (accept method of server actually).

I have seen the example programs in the tutorial. The steps are same. Though I cannot practice it since port number and all is not known. However, few other things must to mention. First, we have read about many different types of IO streams but here we have used none of the types but a new one called “DataOutput/InputStream”. These streams have methods like writeUTF() and readUTF(). This is again for reading and writing but it supports the UTF-8 standard character encoding scheme. We should also use the flush() and close() methods wherever required. Also, if you want try this program later, open two cmd, one for server and other for client. For now, it is out of scope.

## Java URL

URL stands for Uniform Resource Locator. URL points to a certain location on World Wide Web.

An URL contains much information such as

1. Protocol (generally “**http**//www.javapoint.com:80/javatutorial”).
2. IP address/server name/Host name (http//**www.Javapoint.com:**80/javatutorial).
3. Port Number (http//www.javapoint.com:**80**/javatutorial). If no port number is shown in the example, then -1 is returned by default.
4. File name or directory name (http//www.javapoint.com:80/**javatutorial**).

Java.net.URL class gives many methods. Check the example. In the example, we create the object of the URL class and pass an URL as the argument. Then, we have different methods like getHost(), getFile() etc to find certain details. The program was working fine.

## Java URL Connection

Suppose the application you are creating require you to connect the application with a URL. There we use URLConnection class. From the specified URL, your application can read and write data. To make this actually happen, first create a URL object to specify the URL we want to deal with. Then, create an object of the URLConnection class but that is done using the openConnection() method of URL class (which is very important to understand). This needs to be done because openConnection() method is present only in URL class and it returns the object of the URLConnection class as a return type. Actually, a method URLConnection of URL class returns the object of the URLConnection class. So, its object is made using this method. Also, the URLConnection class does not have its constructor. Now, if we have to read something from URL referred, then create an input stream with the use of URLConnection class object and its method getInputStream(). Then, do usual coding for displaying the inputstream. The example, I have tried has given an error of “refused connection”. So, try and figure out the possible reasons for such error. By now, it is going out of scope. Try the example.

## Java HttpURLConnection Class

HttpURLConnection class is used to get the information like header information, status code, response code, etc. It is not about creating a connection but to get the information of an http URL. The HttpURLConnection class has many methods that give such information. Try an example. In the example, the object of HttpURLConnection class is created which needs to be carefully understood. Creation of object of HttpURLConnection class is interesting. The openConnection() method of the URL class return the object of the URLConnection class and HttpURLConnection class is a sub class of URLConnection class. So, typecasting can be done to convert the object of the URLConnection class as the URL class.

## Java InetAddress Class

This class is used to get the IP address for any host name such as [www.google.com](http://www.google.com) or any other host. Check the methods of this class. In the example, the object of the InetAddress class is created first and by using the getByName() method, we pass the host name. Then, we print the host name and host address. Check this class in detail later if needed.

Doubt(s):

1. While trying this, I have tried creating a simple object using new keyword with no parameter. This has given as error as this type of the constructor is not present in the class. However, I also struck as then why not the default empty constructor is not used? An answer to this could be that once you create any constructor of the class (default or parameterized), the default empty constructor by the compiler is not generated. Also, in this case, getByName() method is used which is purposely made to return the object of this class.
2. How does it return the IP address of a website like I have got the IP address for [www.Google.com](http://www.Google.com) and [www.javaTpoint.com](http://www.javaTpoint.com). This happens as the eclipse is connected with the internet and if internet connection is removed then it will not work. Though, this answer is not verified.

## Java DatagramSocket and DatagramPacket

DatagramSocket and DatagramPacket classes are used for connectionless socket programming. DatagramSocket class is used to create a connection less socket for sending and receiving the data. Datagram is some content and there is not guarantee of its arrival at the receiver. DatagramPacket class is used to create a packet (packet of message) that can be sent or received. If we send multiple packets then it can be received in any order. In the example, we have first created an object of the DatagramSocket class to create a connectionless socket. Then, we will create a message string. Then, we will get the IP address by name through InetAddress class. After this, we will create an object for DatagramPacket which needs information like byte version of the message string, length of the string, ip address and port number. Lastly, using send() method of DatagramSocket class, we can send a packet. At the receiver side, create a DatagramSocket object. Then, create a byte array. After this, create a DatagramPacket object and pass the byte array and the port number into it. Then, with the help of receive() method of the DatagramSocket, we can received the data. Lastly, we have to print the data. So, create a new string and pass the necessary parameters and print it.

# Java AWT (Takenover by Tutorials point)

## AWT Home

Java AWT (Abstract Window Toolkit) is an API to develop GUI (Graphical User Interface) or window based application in java. Here, GUI is the interface by which user actually interacts with the application and window based application are those that run by the OS but not run through the browser. In short, by java AWT you can develop GUIs.

## AWT Overview

GUI is Graphical User Interface which is nothing but a user interface through which a user can interact with the application. In my definition, I would like to call it a face of a website. A face has got eyes, ears, nose, mouth and other things. These are all its components. Similarly, GUI has also got its components as window, frame, panel, button, textfield and many more (Where Window, panel, frame, dialog, applet are container type components). GUI works in the form of events. For example, checking a checkbox, Clicking a button, opening a window or closing a window is form of events. Now, let’s understand the basic terminologies.

1. Component – a component is an object that has some graphical representation and can interact with user. Examples are button, checkbox, list etc.
2. Container – Container object contains multiple components of GUI and other components as well which are ordered according to the index decided at the time of adding to the container. If index is mentioned then the newest component will come at last.
3. Panel (kind of a container) – Panel provides a space in which application can attach any other component or any other panel. This is just for grouping up the component and does not stand alone.
4. Window (Kind of a container) – It is rectangular space which is displayed on the screen. In different window, we display different programs and display different data. Window provides us with multitasking environment. A window must have frame, dialog or other window as its owner when it is constructed. It will not be focused until its owner is showing on the screen. It can stand alone.
5. Frame (kind of a container) – it is a top level window with a title, a border and closing, minimizes and maximize buttons. It has resizable corners.
6. Dialog (kind of a container) – It a special type of window. This is mainly used as a pop up window for saving and selecting the file.
7. Applets (kind of a container) – It is a kind of panel.
8. Canvas – It is a blank rectangular area of the screen onto which an application can be drawn.

Note: definitely do check the hierarchy of AWT components.

Some example of GUI applications are ATM, Mobile application etc. But do check the non-GUI application examples. We have CUI also which is Command-line User interface or character user interface. It was the earliest technique of interacting with an application through commands. Though, its use has reduced because of the ease of use and familiarity of the GUI applications. Let’s look on the advantages of GUI over CUI.

1. Obviously, GUI provides graphical icon which are its components that makes the user interaction with the application far better.
2. It provides the facility of clicking and execution environment (IDE’s and all).
3. Using GUI is much easier and interesting at the same time than CUI.
4. CUI supports only one task at a time while GUI can support multiple.

## AWT Environment

This section is all about installing java in the machine.

## AWT Controls

There are three main aspect of any user interface which are discussed below.

1. UI elements – UI elements are nothing but the components of the AWT. These may be Label, text field, text area, scroll bar etc.
2. Layouts – They define how the UI elements (AWT components) are designed on the screen and this defines the final look of the GUI.
3. Behavior – These are the event that occurs when the user interacts with the UI element. In other words, it will define the behavior of the element when user interacts with them.

AWT Component class – This is an abstract super class for GUI components (controls). Below are the most frequently used components.

1. Label – Label displays a single line of read only text. The text can be changed by the programmer but cannot be changed by the user in anyway. The label class extends the component class and implements accessible interface. The label class has a lot of methods and constructor. Study them later if required. It is opposite to Textfield and Textarea since it is read only.
2. Button – this class crates a labeled buttons. Labeled here means that something is written over it. The looks of the buttons changes when cursor passes over it or when it clicked. It creates an event when the button is clicked or released.
3. Check box – A check box can be in ON or OFF state.
4. Check box group – This class is set for using a group of check boxes.
5. List – it presents a user a scrolling list of items. (may or may not have scroll)
6. Text Field – A textfield object allows a text component that allows an editing of single line text.
7. Text Area – It allows the editing of multiple line of text.
8. Choice – A choice control is used to show pop up menu of choices. Selected choice is shown at the top of the menu. (This may also be called a drop down menu).
9. ScrollBar – These are for scrolling horizontally and vertically.
10. Canvas – this class represent user a rectangular area to draw something or receive user input.
11. Image – This class is superclass of all the classes representing the images.
12. Dialog – it is a top level window with a title and border to take some kind of input from the user. It comes in the form of a pop up window.
13. File dialog – it is for the purpose to let user select a file. Let’s say we have to attach a file and after clicking browse, we get a pop up window for selecting the file we want to attach. This window is the file dialog.

Note:

1. Every control/component extends the component class and implements the accessible interface.

2. Refer tutorialspoint for studying components of AWT.

## Event handling

Firstly, we have to understand an event. Whenever any user interacts with the components then an event triggers specific to the kind of interaction. There could be foreground and Background events. Foregrounded events require direct interaction of the user like clicking a button while the background events require interaction of end users like operating system interrupts, hardware or software failure, time expires etc.

**Event handling** – It is a mechanism which decides what to do when a particular events gets triggered. Java uses delegation Event Model for event handling. It has two key parts. First is source which is an object on which the event occurred. It also informs the handler about the event triggered and provides information like what happened and to whom it happened. Second is listener which is known as the event handler. It is responsible for generating the response to an event on the basis of the information provided by the source. The benefit of this approach is that the user interface logic is completely separate from the logic that handles the event. That is why the listener must be registered with the source object so as to receive the notification of the event generated.

Steps involved in event handling –

1. The user interacts with the UI element and an event gets generated.
2. Then the object related to that event gets automatically created and information about the source and the event gets stored (populated) in the same event object.
3. Then, the event object is forwarded to the method of the registered listener class.

The listener class must implement some interfaces which have some forecast callback methods. These methods must be implemented by the listener class for the class to become listener class. These callback methods are nothing but the event methods and in response to the event these callback methods are fired. So, a listener must implement interfaces having callback methods and source class must register the listener to let listener to respond to the events.

Note: I have checked the program and it seems like we need to do it a lot more depth because I am not able to understand that program.

Note: The above information where it is told that event handling has two important things which are source and handler is incomplete and the explanation is also not so clear. Here is another attempt to explain the event handling. Actually, it involves three things which are user interacted with whom, how the user interacted and who should be the processing the response of the GUI. So, the component that user has interacted with is called as source. The way the user interacted is called as event like clicking a button. The response is process by something called as handler. So, an object of the component is created which is nothing but the source. An event is added to that source (component) and then listener is configured for that event (state). A small tip here is that try to think an event as a condition or state and you just need to clarify the UI action on every possible event. Now, since you understood the three key things of event handling then it’s the time to understand the sync between the three to handle the event. Firstly, when we create a component then the reference referencing to that component is nothing but the source. Then, we add events to that component by addActionCommand() method (its just an example for adding action type event but the method will change according to the type of event). We also pass the name of the event so that the different events can be identified among different events. Then, we add listener to that component by addActionListner() method (again I am taking an specific example of listener). In this method, we pass the object of the listener class that we created. Actually, we create the listener class but make use of listener interfaces. Then, in the listener classes that we created in the program we handle the different events. Try the example from tutorialspoint when get time for this.

## Event Class

(bahut der mai samagh aya tha bhai. Huuh!)

Event occurs when a user interacts with an UI element. Every event is referenced with an object which is called as source. The super class of all the event class is the EventObject class which is having two methods. First is the getSource() method which returns the object (i.e. the source and clarifies that which UI element interacted). Second, int getId() which returns an Integer value which identified which type of event occurred on the UI element. Now the event classes are sub classes for the EventObject class which are categorized in semantic and low level. The high level event classes example are ActionEvent (an action is performed either by clicking a button, checking a checkbox, etc), AdjustmentEvent (when the scroll is adjusted) and other. However, low level classes are those which are generated for a high level event. Example of low level events are focusEvent (when the mouse is placed on any check then it glows. Therefore when a high level event occurred like Actionevent of checking a checkbox then along with it a low level event like focusEvent also occurred when the cursor is taken over checkbox), WindowEvent and other. One more thing is important, when comes to event classes then they are not about a particular event on which UI element but it is an event which may be applicable for many UI element. For example, the actionEvent can be done over many UI elements. Also, check other event classes.

## AWT Listener

AWT event listener represents the interfaces responsible to handle events. Java has many event listener classes. Every method of the java listener classes has one argument which is the object of the sub-class of the EventObject class. For example, mouse event listener will accept the instance from the mouse event class which is derived from the eventObject class.

EventListener Interface – It is a marker interface which is implemented by all the classes in the java.util.package. We have some other listener interfaces which are shown below.

1. ActionListener – this interface is implemented by the classes which receives the action event.
2. ComponentListener – This is used to receive the component listener.

Check the rest of the listener interfaces.

Note: Generally, we have listener interfaces for corresponding event classes.

## AWT Adapters

Adapters are the abstract classes which are used to receive the events. Adapters make the event handling easier. In listener, implementing class must override all the methods but adapters can override only those methods of which they need to receive the events. So, the advantage is simple and clear. Check the AWT adapter classes.

## AWT Layouts

Layout means the arrangement of the UI element within a container. The way they are designed gives the final look of the GUI. We also have a layout manager which can do it for us. Let’s see what layout manager it?

**Layout Manager** – layout manager simply arranges the UI element for us and makes our life easier. Though, it can be handled manually but layout manager is preferred because of two key reasons. First, it is very tedious to arrange UI element in large quantity. Second, the size information of the UI element is not shown while arranging them manually. We have different kind of layout manager and the arrangement changes according to the size of the applet viewer and window size.

Check the layout managers and layout manager classes.

## AWT Container

Containers are also AWT component but have the ability to add other components and containers in it. The sub class of a container is components like frame, panel, window etc. Each container has a default layout which can be overridden using the setLayout method. The examples of the some containers are frame, window, panel etc.

## AWT Menu Classes

Almost every high level window has a Menu bar. These menu bars have options available to the end user (those who will actually use the application). These menu bars are also has some options which are in the form of the drop down menu. We have MenuControl class which is the super class for all the Menu and Menu items. Check the hierarchy and the menu classes.

## AWT Graphics

Graphic controls allow the application to draw onto the component or image. Check the graphical control classes.

# Java Swing

## Basics of Swing

Java Foundation classes are a set of GUI components which simplify the development of desktop application. Java swing is a part of the JFC which is used to create window based application. AWT is machine dependent and so uses the machine code also. However, swing is purely java written code for the development of WBA. Swing uses AWT though limited. It uses majorly that part of AWT which is in Java, not it machine language.

History - It was originally named Java foundation classes which later named as Swing (from wiki). Until java1.2, swing was distributed as separate downloadable file.

Difference between Swing and AWT

1. AWT is platform dependent unlike Swing.
2. AWT component are heavyweight as compared to Swing component.
3. AWT window application will look according to the OS but swing gives an option to either look likFe native OS or to have “Java look and feel”. This also means that the java look and feel gives an option of same look on different OS.
4. Swing has more components.
5. AWT does not support MVC unlike swing. MVC is model view control is the architectural pattern for implementing the user interface. Check more on MVC.

Check the java swing class hierarchy. It is same as AWT’s hierarchy.

We already know about what component is. We have a component class which is super class for all the components available and we will look at the most common method of the component class of java swing. These are given below.

1. Public void add(Component obj) – to add the component to another component.
2. Public void setSize(int width, int height) – set size of the component.
3. Public void setLayout(LayoutManager obj) – set the layout manager for the component.
4. Public void setVisible(boalean obj) – set the visibility of the component. It is by default false.

Now, we will see the way to create a frame and add a component to it. Frame can be made by two ways which are by creating its object and by extending the frame class. I have made the first swing program creating a frame and then adding the button to the frame. It’s all about creating object and then using the methods.

Note: local value may not have been initialized error (Compilation error) comes when the local variable is not initialized. However, by rule it must be initialized before it is accessed.

## JButton Class

The JButton class is used to create the buttons which are platform independent (better not to mention, because I know I am studying swing). Well, the use is same which means we can create its object and then use the functionality of the methods. We have to check with the type of constructors and the methods. Also, we have to check with the types of buttons or most commonly used buttons. The button classes are sub classes of the abstractButton Class and uses its methods. Also, for using these methods, we do not need to extend it explicitly. I have understood the example. Check the way of inserting the image. Also, my image was not visible. Check it what’s wrong later when you get time.

## JRadioButton Class

JRadioButton class is used to create the Radio Button. These are used to select one option out of multiple choices. It should be added in the groupButton class to make sure only one option is selected out of many. The, group button should be added to frame. I have checked the example and it’s very easy. Also, check how a constructor is called in the main method in its own class.

## Executable Jar file

JAR stands for Java Archive Resource. It is a package file format which is used to aggregate many java class file and associated metadata and resources like images, text etc. The purpose of putting it together is to distribute an application on a platform. They are archive files and has .jar format. They can be executed by just a double click. We can show our application on other’s computer without even jdk but JRE is needed. They can be extracted using JAR command that comes with JDK or by zip tool. We get a JAR utility in JDK.

Now, we can make the jar file executable and that is called as executable jar file. We can do this by a Jar tool (jar.exe) provided by jdk to create the executable jar file. This tells the runnable about the main method. To do that, we need to create a manifest file (.mf) file. There is a particular syntax for creating manifest files which is “Main-Class: first” where first is the class name. Also, in mf file, new line is just after the class name. Now, java tool has got something called as Switches. For example,“-c” is for creating new jar file. Check the other switches available. Syntax for creating the jar file with the manifest file is “jar –cvmf myfile.mf myfile.jar first.class”. This way an executable jar file will be created and is runnable after the double click. Check how does an executable java file look like? I am not able to find out as why the manifest file is necessary for creating executable Jar file.

Check this link for wonderful explanation on this topic - <http://www.dreamincode.net/forums/topic/243325-how-to-create-a-java-executable-jar-file/>

## JTextArea Class

JtextArea class is used to create a text area that can contain multiline plain text only. Though we will try to create a simple text area but still check the important constructors and the methods of this class for adding more features. One thing to note here is that setVisibility and setLayout is used for frame but not for lower level components. Check if the same setsize, setvisibility, setlatout are to be necessarily written at the end. Nothing is mandatory. But you can set these properties along with setting the background and foreground color.

## JComboBox Class

This class is used to create a ComboBox i.e. the drop down list from which only one option can be selected. Check the example and try more methods for the more features. We have to create an array of string that we need in the drop down box and pass that string as the parameter to the constructor of the JComboBox class.

## JTable Class

This class is used to create a table which can display the data two dimensionally. I have tried an example. Firstly, I have created an object of the frame. Then, create two arrays of string (I am not talking about string here, but array of string) i.e. one for columns and another for data. Then, create the object of table and pass the data and column as arguments. Then, create a JScrollPane object. It is not mandatory to create the scrollPane. Lastly, I have passed the table as argument in the JScrollPane and add it in the frame.

Note: Just understand Pane as a lower level container which can fit inside the higher level container like frame. These are really helpful when the GUI is really complicated.

## JColorChooser

This class is used to create or show the color chooser dialog box through which a user can select a color. Here, i have tried the example. Container is used. Container in swing is a space where a component can be placed and container can add any component into it. We have created a container object and used getcontentpane() method and setLayout() for setting the flowout as layout. I have to understand the purpose for having container and using these methods. Now, in the previous examples, we were having buttons but not adding any event to them. Now, we want to add event to the button so use the “addActionListener()” method. We have to pass “this” keyword as the argument of the addActionListener method. ”this” keyword is used to refer to itself only. Then, we have to create a ActionPerfomred() method passing “Action e” as argument. Check the code of this class later since unable to understand except one thing that “jColorChosse.showDialog()” is used to show the color dialog box. Check the code of the last line in the main method.

## JProgressBar

This is used to indicate the progress of the task. I have tried the example. In this, we create JProgressBar object and then used setBound(), SetValue() and SetStringPaint(). JProgressBar object was made by the constructor with arguments as minimum and maximum value. Then, since we have extended the JFrame class so we simply used the add() and setSize() methods for adding the Progress bar and setting the size of the frame without making its object. Then, we created iterated method where we are actually creating task of printing value from 0 to 2000 with adding 20 everytime with a thread.Sleep(150). Now, we have connected a JProgressBar and a task. Now, with the help of setValue() method we can connect the loop with the progressBar. In the main class, you have to create an object of the class and call the iterate method.

## JSlider Class

This class is used to create Slider. Using this class, user can select a value from a range of values. I have tried an example. I have created a class extending the Jframe. Then created a constructor and created a JSlider object and specified the orientation, min, max and initial value. Then, using the setMinor/MajorTickSpacing(2/10) placed the major and minor ticks. This does not complete here but we have to setPainttick/label() as true. Then, I have created a panel and added the slider to it (though, even without using panel and simply, directly adding the component to the frame also works the same way). Later this, I have added the panel to frame. Then made the main method and made the constructor visible and used pack() method which ensures that the size of frame is bigger than its component. In the example, pack method was used but I have made it without pack method and it is called by the constructor.

## Digital Watch

In this section, we will learn how to create a digital watch. I have tried the example. It is simple. I have created the class method and created an object of frame and a button (don’t know why botton) which I have to that frame. Also, created a thread and called the start() method. Then, created the run() and done the coding for making the data string. I have understood much of this except the part where we have created the time string and used classes and methods related to date and calendar. Once we will be done with date, then we will do it.

## Graphics in Swing

Java.awt.Graphics class provided a lot of methods for graphics programming. In the example, a method was made using the pre defined methods of the graphics class. Also, the canvas class was extended. Check why canvas class was extended. Lastly, the frame created and the class object was added in the frame. Check why the program is not running despite the fact that I have made it correct.

Note: we can only extend one class at a time but we can implement multiple interfaces.

## Displaying Image

For displaying an image in swing, the drawImage() method of graphic class is used. In the example, the canvas class is extended.Then, what is this toolkit and getdefaulttoolkit() metod is? Check the entire program again.

## Edit Menu of Notepad

Do it later if you have time.

## Open Dialog Box

Do it later if you have time.

Note: These topics are not studied in detail. We have just studied them to have a brief idea about all and build concept.

## Java Swing App

This is definitely out of the scope at the time I am writing this. Though, if needed later I can revise the basics and try to understand this section too.

## Layout Manager

Layout is the design in which the components are arranged in Swing. In this section, we will look at the different types of layout managers available where the layout managers are the interfaces which are implemented by the classes to get the layout. There are many types of layout managers though we will study few.

**Border Layout**

It is used to arrange the components into five regions which are north, east, west, south and center. The border manager provide five constant i.e. one for each region. Check the constants and constructors. I have tried an example. Only different we have here is to define the region of the button while creating it. Secondly, while adding also we have to mention the region as well.

## Grid Layout

It arranges the component in the form of rectangular grid. One component is displayed in one grid rectangle. I have tried the example. In this, you just need to mention the number of the rectangle block of the grid the component belongs to at the time of creating the component but not at the time of adding it in the frame. Also, we have to set the layout here as gridlayout (check the way of adding the layout) which is not required in the border layout.

## Flow Layout

This layout arranges the component in a series which are numbered. It is same as grid layout while writing it. We just need to setlayout as flow layout except that everything else is same. Also, we have to initiate the series from left or from right. By default it will be in the center.

## Box Layout

This layout arranges the component into boxes in horizontal, vertical or other orientations. I have tried an example and found that this layout does only have a constructor which has parameters container and int axis. Check the example in the tutorials once again because that I have written was giving error.

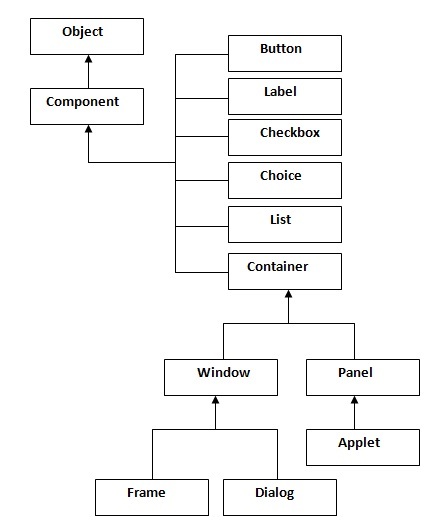
## Card Layout

This layout shows only one component at a time. It treats each component as a card. Check its example later when you have time. Currently it is getting out of scope.

# Java Applet

**Must Read – understanding of graphics in computer**

Anything that you see on your computer screen is graphics. There are a lot of APIs in java which are dedicated to graphics. The top level class, just after object class, we have Component class which is super class for all the GUI component classes. After Component Classes we have classes of the component class like button, Label, checkBox, canvas, choices etc and container class. Container class is also a sub class of the Component class (a special one) along with the classes of component. Container is a kind of component which can container other components or other containers.



Now, the container class has two sub classes which are window and panel. Panel is used only for grouping of components which does not exist alone (when I talk about component then include container as well because it is also a kind of component). Window is for creating a group of component and handling them as well. One example of panel is Applet which does not exist independently, but a web browser or applet viewer is needed to view them. It does not stand on its own. However, a window is represented as independent and is not inside any other component. We have two kinds of windows which are dialog windows and Ordinary windows. Dialog windows are for opening and saving the files which are implemented using dialog class. Other category is the ordinary window which is implemented using the frame class.

Useful link for: <https://staff.fnwi.uva.nl/a.j.p.heck/Courses/JAVAcourse/ch5/s1.html>

**Concept of Graphic class**

The above writing covers the components that we have for producing the graphics. Now, we must understand that for any drawing you want to do, you require a “graphic context”. This graphic context is an object belonging to a class Graphics. Now, the need of the Graphic class object is because it has the methods for drawing shape, text, images etc. With one graphics object, you can draw only at one location i.e. any one of the GUI components. Also, the Graphics class is an abstract class so it is impossible to create an object directly. That is why, you have two indirect ways to use Graphics class object. First, You can use the object of the Graphics class as the parameter to the paint() methods. (PS: We will study paint() later in detail). Second, with the help getGraphics() method. This getGraphics() method is present in every component class. First, create the object of any component like button, canvas, applet etc and then creating the object of the graphics class with the object created of the component like this.

Example: line1 – Button b = new Button();

line2 – Graphics g = b.getGraphics();

Also, we should dispose the object of the graphic class if we have created the object using the getGraphics() method.

**AWT SWING AND APPLET connection**

So far what I have figured out is that AWT was made for GUI application (does not specified for which type of application i.e. for window based application or web based application, so I consider it for both). Now, the difference between AWT and swing I know very well. Just to tell you in short, Swing was for same purpose as AWT but with few improvements which you can check in differences. Swing and AWT is used for designing the GUI of window application while Applet is used for web based application. Explore this more later when you get time. Also, GUI development does not only mean to display things but to make them functional together properly.

**Where to use Applets**

1. Applet code written in java can be added in an HTML file. When we use a browser to view the page then the applet code get transferred to the system and executed by the browsers JVM.
2. I also want to add about the use of Applet. One of its main use is client side validations. It can make the web pages more users interactive. Think of a situation where you are filling a form and the moment you filed a field incorrect, it becomes red. Other situation could be that you filed all the fields and pressed enter. The page reloaded and the same form loads with a message for the correction needed in some fields. So, in the first situation the page become more interactive and that is only because of client side validations done through applet. However, these days, applets are rarely seen as they are replaced by other better options, mainly javascript for validations. If you only want to use java for validation, then applet becomes the option. Also, if you are thinking that how validation becomes a part of applet that you studied then think this way. The applets can create text field and other things. To restrict user to fill them in appropriately we have validations.
3. Another internal use of applet is to get the data of the users’ machine which could typically be done once the user signs the signer’s certificate which is seen very rare to be done by the users.

Note: What applet can do and cannot do: <https://docs.oracle.com/javase/tutorial/deployment/applet/security.html>

1. Paint repaint and update

## Applet Basics

A Java Applet is a java program that runs in a web browser. It can use all the Java APIs and resulting it as a complete java application in itself. Let’s check out the difference between java applet and java application.

1. A java applet class will extends java.applet.Applet class.
2. Unlike traditional java application, java applet does not invoke the main() method.
3. Applets are designed to be embedded inside an HTML webpage.
4. Applet gets automatically downloaded to the user’s machine once the user view the HTML web page.
5. The JVM is required to view the web page. In some cases, the JVM is a browser plugin or can be the JVM of the machine.
6. Then, the JVM instantiate the applet class and invokes the methods defined within the applet class.

**Some advantages of Applets**

1. It runs on the client side and thus takes less time.
2. Secured. Though we have read a little about the Jwrapper which have a significant advantge over Applets due to security reasons.
3. It can be executed by a browser running under many popular OS like windows, linux, Mac OS etc.

It also have disadvantage as the plugins are required at the client side. Check the applet hierarchy.

**Lifecycle of Applets**

1. Applet is initialized – This is done through “init()” method which is needed for the initialization of the applets. It is called after the param tags, inside the applet tags, is processed.
2. Applet is started – This method is automatically called when the init method is called. This method also called when the user returns to that page after visiting some other page. Initialization occurs only once.
3. Applet is painted – This method invokes immediately after the start() method. This method will also be called whenever an applet needs to be painted. Here, painted means to use the methods of the Graphics class to be used for drawing oval, rectangle, arc etc.
4. Applet is stopped – This method is called whenever the user moves off the page applet embedded to. Also, in the case when the browser is minimized.
5. Applet is destroyed – This method is only called when the browser is closed. This is for the basic reason as the applets lives in the HTML page. So, when the browser is shut down then the applets are also destroyed.

Note:

1. The java,applet.Applet class gives init(), start(), stop() and destroy() methods while java.awt.paint is gives paint() method.
2. Java plug-in software is responsible for the management of its life cycle.

Now, let’s look at the way to run an Applet.

There are two ways. First, by HTML file i.e. java enable browser and second, by appletViewer tool. I have made an applet example. In this, first I need to create a .java file which uses java.applet.Applet and java.awt.Graphics. Using Graphics class methods, we can create a .java file. Then, under the applet tag we can attach the .java file and run it on browser. Google chrome and opera does not support the applets while internet explorer, Mozilla, safari do.

## Displaying Graphics in Applets

We have seen Graphics class in AWT as well but unable to understand much about it. Let’s us understand this now. Graphics class let the developers to draw geometric shapes, print text and position images within the borders of a component. In short, all the graphical operations within AWT are performed Graphics class. Please check the common methods of the Graphics class. I understood the example. The graphics that we need to create are written in the Paint(Graphics g) method having argument as the instance of the Graphics class. Then, you can use the methods of the graphics class. We need the graphics context for doing any type of painting. So, just understand the methods of this class well.

## Displaying Images in Applet

Applets are mostly used in games and animation. Therefore, displaying image is very common use. This can be done by java.awt.Graphics class drawImage() method. Other methods used are getImage(), getDocumentBase() and getCodeBase(). I have checked the example. We first have to create a class variable of Image datatype. Then, create an init() method and use getImage() method which has getDocumentBase() and image name as parameter. getDocumentBase() gives the URL of the document that has the image. Then, create the paint() method and use the drawImage() method which has Image type variable, int x, int y and ImageObserver as the parameter where the ImageObserver is an object. Component class implements ImageObserver interface and but applet class extends the component class so current class can also be treated as ImageObserver. Therefore, use keyword ”this” as the 4th argument.

## Animation is Applet

This is a very vast topic though a simple example given to make us understand a bit of it. In the example, an image is displayed 500 times with a sleep of 100 milliseconds.

## Event Handling in Applet

The way we have done the event handling in AWT and swing, the same we can do in applet as well. In this, We have to implement the interface ActionListener and use the method ActionPerformed(ActionEvent e).

## Japplet Class in Applet

We prefer Swing over AWT. We can also use Japplet class that extends the Applet class. In Japplet we use Jbutton instead of button else the program is same as when we extend the Applet class.

## Painting in Applet

We can perform painting operation in Applet by the mouseDragged() method of MouseMotionListener.

Check the example later if get time.

## Digital Clock in Applet

Check it later when you have time.

## Analog Clock in Applet

Check it later when you have time.

## Parameter in Applet

We can get any information from the HTML file as a parameter. For this, we have getParameter() method in Applet. Check the example later when you have time.

## Applet Communication

Java.applet.AppletContext class provides the facility of communication between two Applets. Check more on this when you have time.

# Java Reflection

## Reflection API

Java Reflection API is the process of examining or modifying the runtime behavior of a class at run time. This topic is comparatively an advanced topic.

**Java.lang.Class class** – this class is used to provide two main tasks. First, provide to get the metadata of the class at the run time. Second, provides the methods to examine and change the run time behavior of the class.

We can create an instance of a Class class in three ways which are below.

1. forName() method of the Class class – this method should be used to dynamically load the class and when full name of the class to be loaded is known. Also, this cannot be used for primitive data type.
2. getClass() method of the Object class – This returns an instance of Class class. It should be used when you know the type. It can be used with primitives. Check the example.
3. The .class syntax – do it later. Check the example.

**Determining the class object** - It means to check that the object of the class hold a class, an interface or an array. This can be checked by the help of three methods which are public boolean isInterface(), public boolean isArray(), public boolean isPrimitive(). It will return either true or false. Check the example.

Note:  **Java Reflection:**

Reflection is an API of java which is used to inspect and modify the methods, classes and interfaces at runtime. Java.lang and Java.lang.reflect are the packages that provide the important classes for implementing reflection property in java. Java.lang.Class class is one of the most important classes which has the methods used to implement Reflection. Some of the methods of Class class are forName(), getClass(), which returns the instance of the class Class.

Reflection can be used to get the information about class, constructors or methods. The methods such as getClass(), getConstructors() and getMethods() gets the class, constructor or method to which a particular object belongs.

We can typically use reflection in inspecting the elements of a class like name of class, constructors and method. We can invoke constructors and methods. Infect, we can invoke the private members of a class as well which can only be done through reflection. We do setAccessible(true) for accessing private variables of a method.

## newInstance() Method

The newInstance() method of the Class class and Constructor class is used to create a new instance of the class. The newInstance() method of class Class can invoke method with zero argument while the newInstance() method of the Constructor class can invoke method with multiple arguments. So, constructor class is preferred. This is used to create an instance of the class then what is so special of it than the usual method?

## Understanding Javap Tool

The Javap command is used to disassemble a class file. The javap command used to display the information about the fields, constructor and methods present in the file.

Try an example and understand more of it later as it is out of scope as of now.

## Creating Javap Tool

This is out of scope as of now. So, understand it later when done with the fundamentals.

## Creating your own Appletviewer

Appletviewer creates a frame and displays its result in the frame. We can also create our own frame and display the result in it. Check more this later as it is out of scope for now.

## Call Private Method

We will look at the way of calling a private method from another class by changing the runtime behavior. Check this later when you have time.

# Java Date

## Java Date API

The java.util, java.sql and java.text package contain classes for representing data and time. There are four days of printing the current date in java.

1. Creating object date of the of the class Date of the util package. In the example, first create the object of the date class and then simple print it.
2. We can pass the long type object while creating the date object. We have to use the currentTimeMillis() method for creating the object. “long millis = system.currentTimeMillis”. Else the method is same. The result is same as above way.
3. Same way we can create using the Date() method of the java.sql sql package. Here, the date method will need an argument (madetory). This is because it do not have a constructor without any argument. So, create a long type variable and then pass it as argument. It prints only date.
4. This time we will use the java.util.Calendar package and methods getInstance() and getTime().

## Java.util.Date class

This class represents the date and time in java. It provides constructors and methods that deals with date and time. It implements serializable, cloneable and Comparable<Date> interface. It is also inherited by some classes. However, after the calendar class was made then many of Date class methods were deprecated. Check its methods. I have tired the example. We have simply create an object and print that object. We can pass argument through that object as well.

## Java.util.SQL class

This class represents only dates in java. It inherits the java.util.Date class. It is higly used in the JDBC because it can be used to store the date in the database. Some of its methods were later deprecated. I have tried an example and we can also pass a string containing date with the help of valueOf() method of the data. Check the example.

Doubt: when I tried to import java.util.Date and java.sql.Date in the same class, I received an error that “java.sql.Date collide with other import statement”.

## Java Date format

In java, date formatting can be done by java.text.DateFormat class and simpleDateFormat class. The java.text.Dateformat class is an abstract class which extends java.text.Format class and extended by java.text.simpleDateFormat class. Also, converting date into string is called as the Formatting andvice versa is called as Parsing. Check the methods of the DateFormat class.

I have checked an example. First we have to create an object of current date through Date class. Then, we have to create a string and use the format() method of the class DateFormat to format the class. Try second example later if get time. Also, to convert the string date into date, the entire program is same except we use parse() method instead of format() method.

## SimpleDateFormat class

This class is used to format and parse the date and time in java. It is a concrete class and extends the DateFormat class. Though this method we can present the date and time in which ever format that we write while create the instance. I have checked the example to format date to string. We have to create a current date object. Then, we have to create an instance of SimpleDateFormat class and pass the format we want to print the date in. Also, dd/MM/YYYY shows the format where “d” should be small, “M” should be capital and “y” could be capital or small. Try other methods of this class. Also, I have checked parsing as well.

## Get Current Date

We have three ways for getting the current date. We can get it by java.util.Date class, java.sql.Date class or java.util.Calender class. Though, java.util.Calender class is not recommended. Check the example.

# Java Conversions

## Java String to int

We can convert an String to int in java using parseInt() method of integer class. Whenever we get data from the text field or text area, entered data is received as String. If entered data is String then we have to convert the String to int. This method returns int and accept String as argument. I have tried the example and it is very easy.

## Java int to String

This we can do by the help of valueOf method of String class and toString() method of Integer class. It is used when we have to show the number as a string because everything in a form shown in String. I have tried the example and they are very easy. Just keep in mind that valueOf() method is of String class and toString() is of Integer class.

## Java String to long

Long is nothing but the int with more range. So the requirement of conversion between String and long is for the same purpose as the conversion of String and int. we use parseLong() method of the class Long.

## Java Long to String

This we can do by valueOf() method of String and toString() method of Long class. The purpose is same as we have to show the long values as String. The example is simple.

## String to Float

To convert the String to float, we have parseFloat() method of the Float class. It is because of the same requirement of extracting the float data received in the string.

## String to Double

It is same as above.

## String to Date

This we can do by parse method of the DateFormat class. Check the example. It is easy.

## Extra

**Important concepts**

1. **Difference in object and instance** –A class contains the behavior and property of a group of object. A class contains lots of object. While if we instantiate a class i.e. try to talk about of a particular object which is an instance. For example, we have a Student class which is nothing but made of lots of students and contains properties and behaviors of the students. But when we instantiate the student class and create a instance. Like Student rishav = new student(). Here rishav is instance as we are talking about a spcific student of a class. Now, some people say that object and instance are same. I would say they are correct in a way but not completely. In the example, I said “rishav” is an instance but it is an object as well as it is a student at the same time. In the nutshell, an object is any element of a group of elements of which your class is made and instance is a specific element we are talking about.
2. **Difference in argument and parameter**– This is interesting. Parameters are also called as formal parameters. These are the variable which is passed into the caller method that represents the actual parameter while the argument is called as the actual parameter which is actually passed into the caller method. Example, I = 8; If a method calls I. Then, we can variable I or value 8. Here, I is the parameter (formal parameter) while numeric value 8 is called as the argument (actual parameter).
3. **What exactly static and public keyword mean** – If something (variable, block, method or nested class) is made static gives a general idea that it is related to the class, not to any references. Check static section.Anything declared public means can be used by anyone without any restriction.

**Important terminologies**

For understanding the difference between package and library, try to understand the intentions creating these concepts.

**Package** – it is a namespace which is made mainly on two ideas. To identify the class location like java.packageName.className. We can trace the location of a file with its package name and class name. Also, it is used for scoping purpose. The classes in the same package have different ability to share its resources.

**Library** – it is also an organizational concept. The intention behind creating libraries is to group some similar classes which can be called to give some feature or functionality. It is like you have thousands of program that can be used to give a specific functionality, whenever needed, are placed in library. It is not of a particular project or application but these are general and can be used in any project if needed the functionality it serves.

**API** – It is a part of a Library. There are different opinions that people have about API. Some says, API is a part of the library containing those methods which a program can access while libraries have some private member that cannot be accessed. Other says, as a communication needs an interface like for user and application communication we need graphical user interface, we need Abstract Programming Interface (API) for the communication between two codes (ex. Communication between libraries and framework). These two point of view looks reasonable. Also, APIs were originally the part of libraries but then it became the part of other things as well like framework and other. This means you can encounter API as something else as well but the definition above also holds true.

**Framework** – Your application uses the libraries for some functions already implemented in it for code reusability. A framework runs your code or application and decides what order things must happen or execute. The main difference between library and framework is of Inversion of control which means the control is at the code while using the libraries while control is at framework when it calls the application. Framework provides a skeleton for the application into which the application code is inserted to give it a structure.

# Java Collections

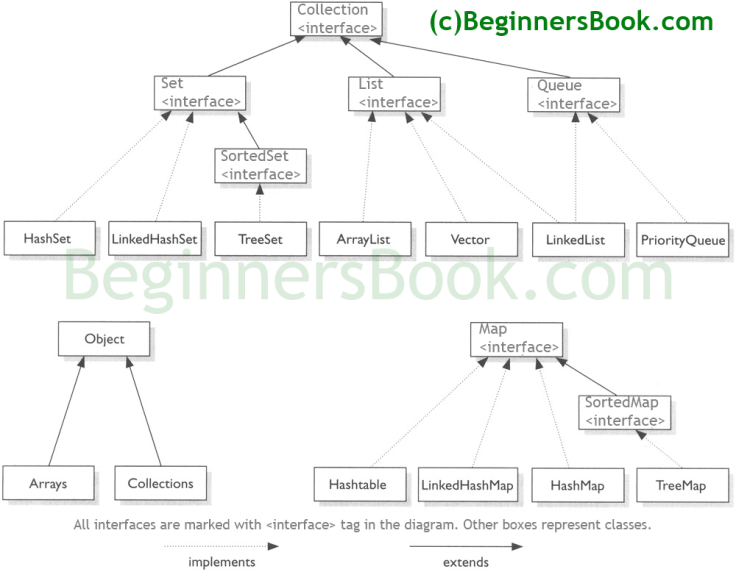
## Collection Basics

Collections in java Wiki – “The Java Collection Framework is a set of classes and interfaces that implement commonly reusable collection data structure. Although it is referred as framework but it works in a manner of a library. The JFC provides both the interface which define the collection and classes that implement the methods of the collection.”

Collection is simply an object that groups multiple elements in a group. A collection framework is a unified architecture for representing and manipulating the collections. Each collection framework has the following.

1. Interfaces –The collection interface is the foundation on which the collection framework is built. The collection interface declares all the methods that a collection will have.
2. Implementation (classes) – It is the implementation of the methods in the interfaces i.e. the implementation of the method that a collection interfaces have. In essence, they are reusable data structures (quoted in oracle docs).This means that these classes will have the methods which will in accordance with the data structure of collection.
3. Algorithms – These are the static methods that perform useful implementations such as searching and sorting on the objects implementing the collection interfaces. Algorithms are known to be polymorphic as the same method can be used on many different implementations of the appropriate collection interfaces. In essence, algorithms are reusable functionality. This means that these functions are simple called from the collection classes whenever required instead of reinventing the wheel again.

Check the hierarchy of the Collection Framework



Note: the above shown hierarchy missed the iterator interface which is extended by Collection interface.

**Iterator Interface** – This interface provide the facility to iterate the elements of a collection. It can iterate only in one direction and once the iteration is complete in one direction then new iterator needs to be created for further iteration. It has three methods which are Public boolean hasnext(), public object next() and public void remove(). Once the iterator method is created for collection, an exception is thrown if the collection is modified. Iterator object is not created by “new” keyword. It is created specific to any collection.

**Collection interface** – This interface extends the iterator interface and extended and implemented by the all other classes and interfaces in the collection framework hierarchy. Check its methods later, if get time.

Begginners Book link to collection tutorials: <http://beginnersbook.com/java-collections-tutorials/>

Note: Collection allows only object to store (String, Wrapper classes and user defined objects). Wrapper classes are actually the replacement for every primitive type data. For example, int can be stored through the wrapper class Integer. Boxing is done to convert the primitive to wrapper class type and vice versa. Some think that it is design fault in java as allowing the primitive data type indirectly through wrapper classes puts an unnecessary load for boxing. But this is how java is designed.

## ArrayList Class

**List Interface**- List interface is an ordered collection (sometimes called as sequence). Lists may contain duplicate elements. Since, it is ordered collection then the position of elements matters. Its elements can be inserted or accessed by their position in the list i.e. index. Index starts from zero.

ArrayList class implements the List interface, so it must be ordered collection and can contain duplicate values. It can contain null elements as well. It is non-synchronous. Here, manipulation is slow as any element is removed then lot of shifting needs to be done. Mostly, arrayList is used in place of array mainly because array is fixed size and arraylist is resizable. Also, if any element is removed from array then memory consumption would be the same as it does not shrinks. Moreover, we can use predefined methods in arraylist that make our task easy.

I have practiced the example. We need to create an object of arraylist first and then we can use the methods of class arraylist to add, add to a specific index position, remove, removing element from specified index position. When we add element at any index where other element is already present then the arraylist expands and shrinks if an element is removed. So far we have just checked the way to create, add and remove element from an arraylist. However, check all the tabs to of the below mentioned links and practice them.

Must to check: <http://beginnersbook.com/2014/08/arraylist-in-java/>

Also check:

1. Simply printing the object of collection shows all the elements of the collection while the Iteration is printing all the elements one by one. We can do the iteration by Iterator() and by for loop.
2. You can use user defined class objects to store in an arraylist. Suppose you have made a class where you have passed the values in the constructor by creating objects in the some other class. Then, we can use those objects as the element of arraylist.
3. We can add element of one array into other in single shot by using addAll() method. Try this.
4. We can also remove the elements of one array which are common to another array in single shot by removeAll(). Try this.
5. Java Collection Framework was non-generic before java1.5. From java1.5 java collection framework became generic which means we can have only one type of element. Here, types mean data type and that we have to mention while creating an object. We mention the type in the angular braces “<String>”. We will get compile time error if we try to insert other type of element.

**Where to use** – It should be used where the elements are more searched then inserted or deleted.

How arraylist works internally? How add() method works internally?

The basic data structure used by arraylist is nothing but array. Now, array list has given three constructors. First, we can create an arraylist like this “List<String> l = new ArrayList<String>”. This uses the default constructor of array list and results in the creation of a arraylist with default size of 10. Second, we can also create an arraylist like this “List<String> l = new arrayList<String>(5)”. This uses the parameterized constructor of arraylist class which results in creation of arraylist with a sixe of 5. Third constructor is little strange so lets leave it now which actually is used to create an array from some other collection.

Now, when an element is needs to be added in an arraylist then we make use of arraylist() method. We a check mechanism in add mehtjod of array list. It first check the capacity of the arraylist and how much an array currently occupied. If arraylist has the scope to add an element then it simply adds but if the array is already full then it will increase the capacity by 50%. for a current capacity of 10, new capacity will be of 15. But how the size of arraylist increases dynamically? It actually creates a whole new arraylist with the new capacity and transfers all the data from old arraylist to new arraylist.

How to implement a list? <http://www.java2novice.com/java-interview-programs/arraylist-implementation/>

## LinkedList Class

LinkedList Class is doubly linked list which implements the List interface and Deque interface. Since it implements List interface so it is ordered and can contain duplicate elements. It can contain null values as well. It is non-synchronous like ArrayList class. Manipulation is faster in LinkedList because no shifting is needed. The concept of doubly linked list is that the elements of the linkedList are treated as nodes and connected by double linked with preceding and succeeding element. That is why named as doubly named, not because it implements two interfaces.

Detail on Doubly Linked list, check: <http://java2novice.com/data-structures-in-java/linked-list/doubly-linked-list/>

I have tried creating the object of the LinkedList. Then, I tried adding, printing and removing the elements from the linked list. It is very much similar to the ArrayList. Many of its methods are same as methods of arrayList. Check other all the tabs of the below mentioned link.

Must to check: <http://beginnersbook.com/2014/08/java-linkedlist-class/>

**Where to use** – It should be used where the insertion and removal of elements is more frequent than searching an elements.

## Comparison between ArrayList and LinkedList

1. Search operation arrayList is much faster than of linkedList. This is because arrayList maintains indexed based system as it uses array data structure implicitly but linkedList uses doubly linked list data structure which requires traverse through all the elements for searching an element. Also, array data structure stores the elements in the linear sequence and every element associated to an index.
2. Insertion and removal of an element in a linkedList is much faster than in arrayList. This is because elements in linkedList are connected through double links with other elements and can be added and removed from the list without shrinking of expanding. While in array list, whenever an element is added or removed than a lot shifting of element occurs.
3. Memory Overhead – since linkedList maintains the data as well as two links to connect two elements while array maintains data and index. Therefore, arrayList uses lesser memory than linkedList.
4. Both are non-synchronous and can be synchronized explicitly.

Note: Doubly linked list is a kind of data structure which is followed by linkedList. Similarly, we have other data structures.

## Vector Class

Vector implements the List interface and it is similar to arraylist collection as it also an ordered collection which stores data on the basis on index. It is synchronized. Since, it is synchronized, it is thread safe as well. It can contain null values as any as user wants.

I have practiced an example. It is similar to arrayList and linkedList in creating, adding, removing and printing the elements.

**Where to use** – It should be used in the multithread environment as it is synchronous. We must avoid using it in single thread case because it has less efficiency and arraylist should be used in single thread cases.

## ListIterator Interface

List interface extends the collection interface and declares methods to insert and delete the element in the List. It is also a factory of ListIterator interface. We have studied the Iterator interface which is extended by collection interface and use to iterate the elements of collection. listIterator interface is specific to List collection. It has few methods like hasnext(), next(), hasPrevious() and Previous(). First two are same as the methods of Iterator interface but other two are to provide the iteration in opposite (backward) direction.

I have checked the example. We have to create a list collection (any type). Then, create an object of listIterator interface for list collection object. Carefully, check the syntax for do this. Then, apply the iteration using listIterator method.

Note: There are mainly three ways to iterate a list. First, to iterate with the simple for loop where we make use of list.size method. Second, we use advance for loop which is only possible with generics where we know the type of the list element. Third, we can make use of iterator where we first make iterator object and then make use of its methods.

## HashSet Class

**Set interface** – Set is a collection that cannot contain the duplicate elements. Some of the Set implementations are ordered and some are not. Set allows only one null element. There is no such thing as listIterator for Set but we use Iterator for it as well. It extends the collection interface.

HashSet class implements the Set interface and so it cannot hold duplicate elements and can have only one null element. It is not ordered so can iterate elements in any order. It is non-synchronous. It stores its element in hashTable. (Hashtable is also a data structure which accommodates the value by mapping it to a key. It is also called as associative array.)

I have tried an example and found that it is similar to List interface. First we need to create its object and then with the help of methods like add(), remove() we can add and remove the elements. Iterator is used for iteration of the elementsCheck all the tabs of the link below. Also, if two null values were inserted then only one will be printed and no exception or error will be given. Same with duplicate values, it will print one value without giving any exception.

Note(s):

1. How to insert null value? The correct way of inserting the null value is to write “null” with double quotes. When we write nothing in the double quotes then it is means inserting the empty String.
2. Hashset is based internally on Hashmap by keeping element of hashset as key with null as value.
3. SortedSet class implements navigableSet interface.

Must to read: <http://beginnersbook.com/2013/12/hashset-class-in-java-with-example/>

How Hashset works internally?

Hashset uses Hashmap internally. Whenever we try to add any value into a hashset then the hashcode of the element is found out using the hashfunction. Then, that element is treated as the key for the hashmap on which hashset is based. A bucket in the hashmap is filled based on the hashcode with element inserted in hashset as the key and value is by default “PRESENT”. Likewise, when we try to remove the value from hashset, the key with the same hashcode is searched and removed from the hashmap.

## LinkedHashSet

LinkedHashSet implements the Set interface and so it cannot contain duplicate elements and can contain only one null element. LinkedHashSet maintains the order of insertion i.e. it will iterate the elements in the order it is inserted. I have found nothing on whether they are synchronous or not. I feel that they are also non-synchronous. It is same as HashSet except the order of the elements iteration.

I have checked an example and everything is same. Though check all the tabs of the link given below.

Must to read: <http://beginnersbook.com/2013/12/linkedhashset-class-in-java-with-example/>

## TreeSet Class

TreeSet class implements the Set interface and so it cannot contain the duplicate value. Unlike other sets, treeSet cannot contain any null value and if tried then nullPointerExcetion will generate. Actually, adding null element in treeSet was possible until java 6 but later it declared as bug. TreeSet maintains the element iteration in ascending order which means from latest inserted to the older. It is non-synchronized. It is similar to hashSet except the order of element iteration.

I have checked the example and everything is fine. Just check the tab in the link below.

Must to read: <http://beginnersbook.com/2013/12/treeset-class-in-java-with-example/>

Note - We will do a deep analysis of where these are taken in use but I would like to make one point here. The selection of collection for use is decided on the requirements and we have to see as which collection completes our requirement.

Note: check time complexity and try to relate them with the different type of collections.

## PriorityQueue

Queue Interface – It extends the collection interface and represents a group of object just like list but the data is intended to be inserted at the end of the queue. The elements are ordered in First in First out manner generally, but not necessarily. It has two implementations which are TreeSet Class and PriorityQueue class. Ordering is must.

Priority queue implements the Queue interface and but it does not follow FIFO (link: <https://docs.oracle.com/javase/7/docs/api/java/util/Queue.html>). It follows LIFO which is last in first out. It can contain the duplicate elements (Checked myself). It cannot contain even a single null value. Do not know if it is synchronous or non-synchronous.

I have checked the example. It is same as conventional collection. It has a Poll function that removes the head element. Head element is the first element of the Queue.

Note:

1. Blocking Queue is tread safe which has two threads which are commonly known as producer and consumer threads. Producer thread keeps on putting the object inside the queue and it is blocked as the queue is completely occupied. The consumer thread then start taking object from queue and until consumer thread takes an object out of queue, producer thread remains blocked.
2. Queue and stack are in interfaces with classes implementing them present inside concurrent package and are FIFO and LIFO mostly. So, they cannot be accessed randomly. In such cases, we can use them.

## HashMap Class

Map interface – It extends the collection interface. It takes the element with respect to a key as a combination of key and value. The pair of key and value is called as Entry. Entry is a sub-interface of Map. We can access it by Map. It provide key to provide methods to get key and value.

HashMap is a class that implements the Map interface and hence it contains a pair of key and value and cannot contain duplicate key. Elements are not ordered in this class. It permits onenull keys and multiple null values. It is not synchronized. The difference between HashSet and HeshMap is that heshSet contains only values while HeshMap contains keys and values both.

I have tried an example. Firstly, I create an object where we need to specify the key and value types. Then, to add the values to a Map, we have used put() method instead of add() method. We can print the object of Map. Now, if we have to iterate the entries (pair of keys and values) then we have to create an object of iterator as usual. However, to create object of iterator we have to first create a Set view of a Map. For doing so, the syntax is: “Set obj = MapObj.entrySet()”. But why do we need to create the set view? Check note point below. Then, we can iterate the keys and values using Set view object. Though I have understood the basic methods still check all the tabs and methods of the below link.

Must to read: <http://beginnersbook.com/2014/08/java-hashmap-class/>

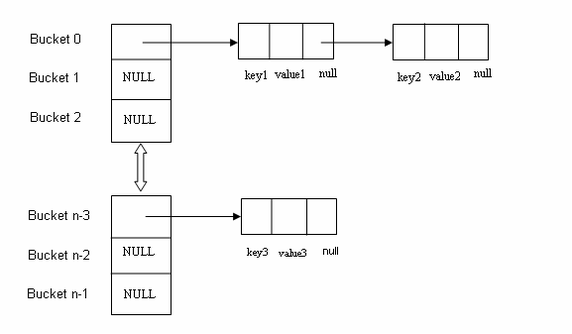
Note: We can understand the requirement of the set view as the HashMap class does not have the iterator method while set interface have. So, we create the object of set class. Since it is an interface thus we cannot create its object directly and hence we do it through the entrys et class that returns an instance of the set interface. I am not sure about this by the reasoning looks fine.

How Hash Map works internally or how they are implemented? Also, how put() and get() method works?

Before getting into how an hash map works, we must understand certain things in the below mentioned points.

1. HashCode function – This function is used to get the hash code for a given value. It takes a key as an input and returns its hashcode which is an integer value. The hash code is also called as hash value.
2. The entry of a key and value pair is stored in bucket. These buckets are actually implemented as linked list. So, we have multiple buckets in a hashmap and each bucket is nothing but a linked list. Though, from java 8, hash map has started using balanced tree set after a certain threshold is reached for values in linked list.
3. Equals method is used to compare objects. So, in a situation where a bucket (linked list) stores multiple entries then equals method is used to find the entry actually referred.

Now, let’s see how actually a hashmap works. When we try to add an entry to a map then we make use of put(key, value) method then hashcode method is used to implement to get the hash code of the key. On the basis of the hash code (hash value) of the key, the bucket in which the entry needs to be stored is decided. Similarly, when an entry is retrieved from a hash map then get(key) is used. This also calls the hashcode method to get the hash value of the key which then be used to find the bucket where the value is stored.



Now, since the entry is stored in bucket on the basis of the hash value of the key. Then what will hapeen in a situation where two different keys has the same hash value? There multiple entries will be stored in a single bucket. Now, while retrieving the value from the hashmap will keys with the same hashvalue, how will be resolve that key entry is called. Then, we make use of equal method to identify which entry is exactly called.

Note: In the hashmap of length 8, a maximum of 8 buckets are possible. Each bucket is an instance of a hash map.

## LinkedHashMap Class

LinkedHashMap implements Map interface and extends the HadhMap class. It has all the properties of HashMap class but ordered in the order of insertion. Also, LinkedHashMap is HashTable and LinkedLink implementation of Map interface.

I have tried an example and it is the same as hashMap and seen the iteration in the order of insertion. Check the below link carefully.

Must to read: <http://beginnersbook.com/2013/12/linkedhashmap-in-java/>

Note: Map implementations have the same order of iteration of elements as SET implementations have.

## TreeMap Class

TreeMap implements the Navigational Map interface and extends AbstractMap methods class. It also contains value with a key. It also cannot contain duplicate elements. It can have null values but cannot have null keys. It is same as HashMap but it has the ordering of the elements in ascending order. TreeMap is Red Black Tree based Navigational interface implementation.

I have checked a program and It is same as other Maps. Check the link given below.

Must to read: <http://beginnersbook.com/2013/12/treemap-in-java-with-example/>

## HashTable

HashTable is an array of list. Each list is known as bucket. It implements Map interface and extends Dictionary class. Actually, HashTable started implementing the Map interface from java 2 and became the part of collection framework. It is similar to hashMap but is synchronized and cannot insert null key. It is similar in the sense that it also stores entries as key and value pair. It can contain duplicate value will print only a unique key. Explore more for better understanding.

I have checked an example. It is same other Map collections. Though explore more for better understanding.

Note:

1. Hashtable is fail safe, indifferent to Hashmap.
2. We have three collection views of Map. Since, map is not counted as Collection then we can have collection view of map which Keyset, entrySet and Values. Keyset contains the set of keys contained in a Map. EntrySet is a method of map interface that provides the set of key and value [air in a Map. Likewise of keyset and entryset, we have value method which provides the set of values present inside a map.
3. Java..util.properties extends java.utils.hashtable.

## Differences between HashMap and HashTable

1. HashMap is not synchronized (can explicitly be synchronized) while HashTable is synchronized.
2. HashMap can contain null values (one key and multiple values) but HashSet cannot contain any null values.
3. HashMap introduced in Java 1.2 and HashTable is a legacy class. This means HashTable was later implemented the Map interface and became a collection.
4. HashMap is faster than HashTable. (apply sync. and non-sync funda).
5. HM can be iterated by iterator but HT can be by Iterator and enumerator.
6. Iterator that HM has is Fail-Fast while Enumerator in HT is not Fail-Fast.
7. HM extends AbstractMap class while HT extends Dictionary class.

Note (s):

1. Enumerator – Iterator methods have shorter names than enumerator method names. Iterator has an additional remove() method.
2. Fail Fast – it means that when we try to iterate and mean while try to modify the content then it will give an exception “ConcurrentModificationException”.

## Sorting

In collections, we generally use String, Wrapper class objects and user defined objects. We can sort the element of String object, Wrapper class object and User defined class object. Also, these elements should be comparable as well. String and Wrapper classes implement the comparable interface implicitly.

Collections class provides the static methods for sorting the elements of the collection of some specific types. Also, the sort() method of the collections class is used only for sorting the list collection elements. It does not sort the other types.

Note: Please do not get confused with the words: Collection interface – It is an interface which extends the iterator class and extended by all the lower level interfaces. Collections class – it contains the static methods for sorting the collections, concurrent collection, synchronized collection, readonly collection and many more having certain types of elements. Collections (neither interface nor class) meant as a plural version of the word collection. ;-)

I have checked an example. First, create any type of List collection and insert elements in it. Then, simply by using the sort() method, we can sort the elements. String type is sorted in alphabetic order and Integer (Wrapper) is sorted in increasing order.

Note: collections sort uses Arrays sort method so has same performance. But takes sometime to convert the collection to array.

## Comparable Interface

Comparable interface is used to order (sort) the objects of user defined class. This is because the String and Wrapper class are already comparable and can be compared by Collections class method sort(). It contains only one method called compareTo(Object). You can sort the elements based on the single datamember like rollno, name, age etc.

Doubt: I have tried the program but have to major doubts. First, why does the downcasting is done if the same class is object is sent as parameter. Second, logic of the “if else” conditional statements is fine but how different objects will be compared. Take someone’s help.

## Comparator Interface

Comparator interface is also used to sort the object of the user defined class on the basis of multiple datamembers. This interface contains two methods for sorting. First, compare(Object o1, Object o2). This helps in comparing the object on the basis of multiple datamembers.

Doubts:

1. Why cannot the comparable class be used for comparing by multiple datamembers of the object by using the multiple iterator and multiple times using the compareTo() method. What special we have done in comparator interface.
2. I understood that we have created a class whose objects are added in the list. Then, made all the classes for using the compare method for each datamember we want the sorting for. Then, made a main method. I have understood the connection setup among all classes also but how they are comparing and sorting the values are what is did not understand.
3. It has equals() method then why is it using the compareTo() method for comparing string. Infect, compareTo() can just compare the object location but not the content.

## Differences between Comparable and Comparator interface

1. Comparable provides single sequence sorting (on the basis of single datamember) while comparator can provide for multiple.
2. Comparable affects the original class while comparator does not.
3. Comparable has one method compareTo() while Comparator has two methods which are compare() and equals().
4. 1st has lang package and 2nd has util package.
5. They have different parameter of their sort method.

## Properties Class

This class is used store the data in the pair of key and value in the form of String. It is sub class of HashTable. We can add and access the properties values through the properties keys from the properties file. The main advantage of properties file is the lowest maintenance. Once the pair is saved and then we do not need to execute it again to update the value. Properties class has the facility to specify default value when no value is defined for certain key.

There are few examples of properties files. First is to create a properties file. Second, getting the information from the properties file and third, is to get the properties of the system.Check later when get time.

Need to be done

1. Data structure of all the collection and then do the other important data structures.
2. Try to complete the Data structure and algorithms tutorials. <http://www.tutorialspoint.com/data_structures_algorithms/>

# Java JDBC

## JDBC Introduction

Java JDBC (Java Database Connectivity) is a java API which connects and executes query with the database. JDBC API has JDBC driver which is specifically used to connect the java application with the database. Now, why to use JDBC? Before JDBC, we had ODBC whose driver was written in C language which is platform dependent and unsecure. There, we have java API JDBC which has its driver written in java that makes it platform independent and much more secure.

API – Java API is a set of classes included in the development environment (this refers to packages and libraries). These classes are written using java language and run on the JVM. So, basically it contains all the classes of the java that forms its basis. Also, JDK has two main things which is the compiler, debugger and other classes and the JRE. Now, JRE contains the JVM and API (libraries and packages).

## JDBC Driver

The JDBC driver is responsible for the connecting the java application with the database. The JDBC drivers are of 4 types.

1. JDBC-ODBC bridge driver – This converts the JDBC function calls into ODBC calls. This needs to be installed in the client machine. Though this was easy to connect and use.
2. Native API driver – This instead of converting the JDBC function calls into ODBC converts the function calls into Native Database API library. It is not entirely written in java. Its performance is better than JDBC-ODBC.
3. Network Protocol Driver – This include a middle ware in the form of application server. This driver uses the middleware to directly or indirectly convert JDBC calls to vendor specific database protocols. It is fully written in java. However, network support is becomes necessary for the client machine. Also, needs the database specific coding needs to be done in the database.
4. Thin Driver – This driver itself converts the JDBC function calls to vendor specific database protocols. It is also known as Thin driver. It is fully written in Java. It has the best performance among all the types but we need to choose it according to the database type.

Note: Actually driver is mainly needed to convert the JDBC function calls to vendor specific (Oracle, sql, Mysql etc) protocols.

## 5 steps to Connect to the Database in java

To connect the java application with the database we need to follow 5 steps. These steps are

1. Register the driver class – This means we need to register the driver class with the application. This can be done by forName() method of the Class class.
2. Create connection - We can create a connection using the getConnection() method of DriverManager classes that we just got after registering the driver.
3. Create statement – we can do this by createStatement() method of the connection interface.
4. Execute a query – This we can do by executeQuery() method of the Statement interface. We print the result set using while loop.
5. Close connection – this we do by close connection of connection interface.

Note: Check the coding of all the above steps.

## Connecting the Oracle Database

Well, we already know the following steps to connect the database. However, while connecting to a database we need to have some database specific information. Let’s try for oracle 10g. Driver class for it is “oracle.jdbc.driver.OracleDriver”. So, we have to register this class to our java application. Then, the URL needed while making connection. It is “jdbc:oracle:thin:@localhost:1521:xe” where jdbc is the API, oracle is database, thin is a type of driver, local host is the server name, we may use the IP address, 1521 is the port name and xe is the oracle service name. We need pass username and password while making connection. Username is System and password is given by us.

I have tried to make connection and made entire program correct. Though, it was giving one error because the ojdbc jar file was not loaded. So, what is this ojdbc14.jar file and how we load it?

Ojdbc14.jar File – This file need to be loaded for connecting the java application with oracle database. OJDBC is oracle java database connectivity.

The ojdbc14.jar file can be downloaded from internet and then pasted specifically in JRE/Lib/ext path and then the path should be set. Path can be set either by command prompt or by the control panel. While setting with command prompt the syntax is “Set ClassPath = c:\folderpath\ojdbc14.jar;.;”

## Connecting the MySQL Databse

Try this later if you get time.

## Database connectivity with Access with and without DSN

Access is a Microsoft based DBMS just like oracle, MySQL. In this section, we are describing about the procedure of connecting the java application with the access database with and without DSN. DSN is Database Source Name which contains all the information about the database to which the ODBC is to be connected.

Try this later if get time because java applications are mostly connected with Oracle, MySQL and SQL.

## Driver Manager Class

This class acts as an interface between the java application and the driver. It keeps track of the drivers available for creating the connection. This driver manager class has a lot of driver and it maintains a list of drivers which are connected with any java application. Check the methods of the DriverManager class. It has methods of registering, deregistering a driver and creating connections.

## Connection Interface

It is used to create statement object. By this only, we can execute the queries. Also, this interface is used for providing many important SQl trasactions methods like rollBack(), Commit() and AutoCommit(). Also, we can close the connection with Close() method.

## Statement Interface

This interface is used to execute the SQL statement and creates the resultSet. It has different methods for different category of commands. Like for select statement we have executeQuery() method while for all update related SQL statement (create, drop, insert, update, delete etc) we have executeUpdate() method. Also, for returning multiple resultset we have “public boolean execute(String sql)”. We can execute batch Cmmands as well by public int[] executeBatch() method.

Note: Batch file in SQL contains a set of commands (batch) which are always written together so that you can simply enter the batch name rather than writing the set of command every time you to use.

## ResultSet Interface

The object of resultSet contains the resultSet which we get after executing a SQL query. Its object maintains a cursor at a particular row of data. Initially, cursor points towards the first row. By default, resultSet object can move further but cannot be updated. However, we can make this object updatable by passing “ResultSet.TYPE\_SCROLL\_INSENSITIVE, ResultSet.TYPE\_SCROLL\_UPDATABLE”. Also, check the methods that it has for moving cursor.

## PreparedStatement Interface

It is a Sub-Interface of Statement Interface. It is used to execute parameterized query. We pass “?” as a parameter. The value will be set the setter method of the prepared statement.

Why to use PreparedStatements – The concept of prepared statement is that these statements are compiled once and can be executed multiple times. So, it increases the performance as the statement is not compiled every time we call it.

I have checked few examples. We have to register the driver and then create the connection. Once, it is done then we have to create a prepared statement object and pass the sql query as an argument which is different from simple statement. Since we pass the “?” as the argument of SQL query so have to define these arguments with the help of the methods of the preparedStatement interface. Also, check the syntax for creating the preparedStatement. Then, we have to execute the statement using execute method depending on the type of query. Then, we can print the result that we have taken in an integer variable. Practice few examples.

Note: The prepared statement are compiled and saved within the DATABASE. Though, remember that not all the databases support the prepared statement.

## ResultSetMetaData Interface

MetaData is the data about the data i.e. the further information that we get from the data. We can get the meta data of the result of the executed SQL query. This can be done by the methods of the ResultSetMetaData Interface. It has methods like getColumnCount(), getColumnName() etc. I have checked an example and we can simply call the methods of the resultSetMetaData interface by the objects of the resultSet Interface. However, the object of the ResultMetaData can be created with the help of the getMetaData() method of the ResultSet Interface.

## DatabaseMetaData Interface

We can get the MetaData of the Database by the help of the methods of the DataBaseMetaData Interface. It has got the methods like getDriverName(), getDriverVersion() etc. we can simple create an object of DatabaseMetaData interface and use its methods. However, to create the object of the DatabaseMetaData class we have use the getMetaData() method of connection interface. Syntax is: “DatabaseMetaData md = con.getMetaData()”. This will return an object md of DataBaseMetaData class. Check its method and try the examples.

## Storing the image in the Database

We can store an image in the database with the help of preparedStatement interface method called setBinaryStream(). This method is used to set the binary information into the parameterized index. In its example, the driver was registered and connection was made. Then, we have connected a prepared statement and passed an sql query which updates an images with its name in the name column with string data type and photo column with BLOB data type. BLOB (Binary Large Object) is used as the data type in the database to store an image. Then, we have to use the setString() of prepared statement to set the name and SetBinaryStream() method for setting the image in the photo column after creating a fileInputStream which refer an image. Then, we have to execute the update query and get the result set. Try an example.

## Retrieving image from the Oracle Database

Even after storing the image in the database, when the table is seen all the data will be shown except the image. To show the image, we have to retrieve the image from the database. To do so, we use getblob() method of the prepared statement which is used to get the binary information and getbytes() method which is called on the blob object to get the array of the bytes of the binary information that can be written into the image file. In the example, we create the prepared statement. Then, if statement we get the blob object (which means blob is data type as well as a class like string) that gets the binary information of the image through getblob() method. Then we create an array of byte on the object of blob. Then, we write the array of byte on the image using the FileOutputStream. Check the example for better understanding. Check why retrieving an image is important in Database.

## Store file in Database

This is same as storing an image in a database, though the data type of a file for database is CLOB (character Large Object). We make use of the setBinaryStream() method to store the file. Check the example.

## Retrieve file in Database

This is same as retrieving the image from the database though the method for retrieving changes from getBlob() to getClob(). We use fileWriter in it. Check example.

## Callable Statement

To call the stored procedure and the functions, collableStatement interface is used. Stored procedures are the SQL queries which are given a name and stored in the database in the precompiled form and hence can improve the efficiency. Also, suppose you need the age of an employee by giving a name so you have created a function that takes the name and returns the age. This is a function.

**Difference between stored procedure and function**

1. SP is used to perform business logic while F is used for calculation.
2. SP must not have a return type while F must have a returntype.
3. SP may return zero or more values while it returns only one value.
4. We can call F from SP but cannot call SP from F.
5. SP support input and output parameter while F supports only input parameter.
6. Try catch can be used in SP but not in functions.

To create an instance of the callable statement, we should use prepareCall() method of the connection interface. The syntax is: “CallableStatement cs = con.prepareCall(“Call procedureName(?,?,?....)”)”.

Practice a detailed example after having JDBC and database.

Note: Outer parameters are those which can either return data or the cursor variable where this cursor variable is the result retrieved one row at a time.

## Transaction Management in JDBC

A transaction represents a single unit of work. ACID describes the transaction management very well. It stands for Atomicity, Consistency, Isolation and durability. Atomicity means either all successful or none which means that a set of SQL statements either all or none gets executed because partial change is dangerous. Consistency means bringing database from one consistent state to other. Isolation means one transaction is isolated from other. Durability means once a transaction has been committed it will remain so even in the events of error, power loss etc.

The main advantage of maintaining a transaction is that it increases the speed. The collection interface consists of methods of transaction like commit, rollback etc. In the example, we have just used the commit() method called by the object of connection interface. Check the example in detail.

## Batch Processing

Instead of executing single query, we try to execute many queries together to improve the efficiency. It makes the performance faster. In this, we use the addBatch() method of the statement class to add a query in the batch. Then, we use executeBatch() method of execute the queries added in the batch together. Try the example.

# Generics in Java

Compile time errors are easier to deal as compared to run time errors. This is because the compile time errors can be seen through the compiler log/error messages whereas the runtime environment cannot be tracked easily. Now, why I am telling you this here is because generics deals with making more of your bugs detectable in the runtime.

## Why use Generics?

In this section, we will understand the advantages we get using generics in our code. Generics has three main advantages which are described below.

1. Type Safety – Here, “Type” signifies the type of the object. In absence of generics, we create collection without specifying the type of object. When we will be using the add() method to join the collection. Then, we will be asked for the type of the add() method based on the data type we want to insert. Though, we can add different type of data as well.

# Missing topics

Jdbc rowset

Java New features

RMI

# Implementations

I have created this section for the sole purpose to speak about the implementation of what we have learned in this tutorial. We must use as many practical examples as we can related to the real world.

**1. Core Java Programming Introduction of Java**

Introduction to Java; features of Java

Comparison with C and C++

Download and install JDK/JRE (Environment variables set up)

The JDK Directory Structure

First Java Program through command prompt

First Java Program through Eclipse

**2. Data types and Operators**

Primitive Datatypes, Declarations, Ranges

Variable Names Conventions

Numeric Literals, Character Literals

String Literals

Arrays(One dimensional; two- dimensional)

Array of Object References

Accessing arrays, manipulating arrays

Enumerated Data Types

Non-Primitive Datatypes

Defining a class, variable and method in Java

Method Signature; method calls

Expressions in Java; introduction to various operators

Assignment Operator

Arithmetic Operators

Relational Operators

Logical Operators

Conditional Operators

Operator Precedence

Implicit Type Conversions

Upcasting and downcasting

Strict typing

Type conversion

**3. Control Flow statements**

Statements and it's various categories in Java

if, if-else, if-else-if

switch case

for statement (both flavours traditional and enhanced for)

while and do-while loops

The continue Statement; labelled continue statement

The break Statement; labelled break statement

return statement

**4. OOPS and its application in Java**

Classes and Objects

Defining a class;Defining instance variables and methods

Creating objects out of a class

Method calls via object references

Abstraction

Interfaces and Abstract classes

Abstract and non-abstract methods

Inheritance

extends and implements keywords in Java

Super class and Sub class

this keyword, super keyword in Java for inheritance

Concrete classes in Java

Polymorphism

Compile time polymorphism -- Overloading of methods

Run time polymorphism -- Overriding of methods

Method  Overriding rules and method overloading rules

Introduction to Object class and it's methods

Encapsulation

Protection of data

Java Bean, POJO

Getters/Setters

Memory management in Java

Heap

Stack

**5. Packages**

Need for packages

What are packages; package declaration in Java

Import statement in Java

How do packages resolve name clashes?

**6. Miscellaneous**

Var-Args

Reference variables, local variables, instance variables

Memory allocations to variables

Double equals operator(==) operator for primitives and objects

toString() method on an object

**7. Statics**

Static variables and methods

Static imports

Static initialization blocks; instance intialization blocks

Static concept in inheritance

**8. Constructors**

What are Constructors?

Properties of Constructors

Default and Parameterized Constructors

Rules for constructor implementation

Constructor Chaining

this call; super call for constructors

Constructors for Enumerated Data Types

Constructors concept for Abstract classes and interfaces

**9. Exceptions in Java**

What are Exceptions?

Need for exceptions

How can Exceptions be coded in Java?

API heirarchy for Exceptions

Types of Exceptions

Keywords in Exception API: try, catch, finally, throw, throws

Rules for coding Exceptions

Declaring Exceptions

Defining and Throwing Exceptions

Errors and Runtime Exceptions

Custom Exception

Assertions

What are Assertions?

Enabling and disabling assertions in development environment

**10. Strings in Java**

What are Strings?

String heap memory and Constant Pool memory

Immutability in Strings

String creation on heap and constant pool

Method APIs on String; operations on Strings

Mutability of String Objects - StringBuilder and StringBuffer

Splitting of Strings and StringTokenizer class

**11. Collection Framework in Java**

The Collections Framework

The Set Interface

Set Implementation Classes

The List Interface

List Implementation Classes

The Map Interface

Map Implementation Classes

Queue Interface

Queue Implmentation classes

Utility classes

Sorting collections using utility methods

equals() and hashCode contract in Java collections

overriding equals and hashCode methods in Java

New Collections added in Java 1.6

Primitive wrapper classes and all its method APIs

**12. Generics**

Generics for Collections

Generics for class

Generics for methods

**13. Input-Output in Java**

What is a stream?

Overview of Streams

Bytes vs. Characters

Overview of the entire Java IO API

Reading a file; writing to a file usinf various APIs

Reading User input from console

PrintWriter Class

**14. Serialization**

Object Serialization

Serializable Interface

Serialization API

ObjectInputStream and ObjectOutput

Transient Fields

readObject and writeObject

**15. Inner Classes**

Inner Classes

Member Classes

Local Classes

Anonymous Classes

Static Nested Classes

**16. Threads in Java**

Non-Threaded Applications

Threaded Applications

Process based multitasking Vs Thread based multitasking

Thread API in Java

Creating Threads

States of a Thread

Sychronization for threads; static and non-static synchronized methods; blocks; concept of object and class locks

Coordination between threads - wait, notify and notifyAll methods for inter-thread communication

**17. Applets**

What are applets?

Need for Applets

Different ways of running an applet program

Applet API heirarchy

Life Cycle of an applet

Even Handlers for applets, mouse events, click events

**18. Swing GUI**

Introduction to AWT

Introduction to Swing GUI

Advantages of Swing over AWT

Swing API

Swing GUI Components

Event Handlers for Swing

Sample Calculator application using Swing GUI and Swing Event handling

**19. JDBC**

What is JDBC; introduction

JDBC features

JDBC Drivers

Setting up a database annd cretaing a schema

Writing JDBC code to connect to DB

CRUD Operations with JDBC

Statement types in JDBC

Types of Rowset, ResultSet in JDBC

**20. Access Modifers in Java**

What are access modifiers?

Default

Protected

Private

Public

**21. Debugging of Java Programs in Eclipse.**

**ADVANCED JAVA Syllabus:**

**Servlets and JSPs**

**1. Basics of a Web application**

What is a web application?

What is a web client and web server?

How do client and server communicate?

HTTP protocol basics

HTML language basics

What is a TCP/IP port, URL?

Need for a Web Container

**2. Web Container and Web Application Project Set up**

To set up Tomcat Container on a machine

To set up a Servlets JSP project in Eclipse

To configure dependency of Servlet JSP APIs

Web application project structure

**3. Servlets**

What are Servlets?

What can they do? Why are they needed?

How do Servlets look in code?

HTTP Methods; GET, POST, PUT, DELETE, TRACE, OPTIONS

GET/POST request; differences between the two

Servlet Lifecycle

Servlet Context and Servlet Config

Forwarding and Redirection of requests

**4. Session Management**

What is a session?

Why is it required?

How to get a session?

Session information passing between client and server

Sesssion information passing mechanisms - Cookies, Rewriting

How to destroy a session

**5. JSPs**

Introduction to JSP an dneed for JSPs

Basic HTML tags

JSP Lifecycle

**6. JSP Elements**

Scriptlets

Expressions

Declarations

Significance of above elements and fitment into the JSP Lifecycle

What are Directives in JSP?

Page Directive

Include Driective

Taglib Directive

**7. JSP Tag library**

JSP Standard Actions

Expression Language

JSTL basics and it's usage

Need for Custom Tag Library

Custom Tag Library implementation

**Struts Framework (version 2.x)**

**1. Basics of MVC**

What is MVC?

MVC Type1 and Type2 architecture

Why Struts framework?

Struts 1 overview

Struts 1 and Struts 2 comparison

**2. Struts 2 Architectutre**

Architecture Diagram explanation of following components:

Components of Model, Views and Controller in Struts Framework

Interceptors

Model/Action classes

Value Stack

OGNL

Introduction to configurations; framework and application architecture

Declarative and Annotations configuration approaches

**3. Struts 2 set up and first Action class**

Download JAR files

Struts 2 project build up and Configuration files

To build Action class

To intercept an HTTP request via Struts2 framework using Action class

Defining data and business logic in Action class

Preparing and Forwarding control to Views

**4. Struts 2 Interceptors**

What are Interceptors

Responsibilities of an Interceptor

Mechanism of Interceptor calling in Struts 2

Defining Interceptors

Defining Interceptor stacks

Defining Custom Interceptors

**5. Struts 2 Tag Library**

Introduction to tag library of Struts 2 and it's usage

**6. Struts 2 Validations**

Validations using Validateable interface

Workflow interceptor mechanism for validations

Validations using Validateable interface

Validation Framework introduction and architecture

Validating user input with above two mechanisms

**7. Struts 2 Tiles Frameworks**

Introduction to Tiles in a page

Struts2 Tiles framework introduction

Defining tiles.xml file

Configuring pages for tiles

A complete Tiles example with Struts2

**Hibernate Framework (version 3.x)**

**1. Introduction**

What is ORM principle?

Why ORM?

ORM implementations

**2. Hibernate Architecture**

Introduction to Hibernate

Hibernate Architecture

What are Peristent classes?

**3. Hibernate CRUD**

Setting up Hibernate project

Configuring all JARs and XML files

Setting up connection to DB using Hibernate

Performing basic CRUD operations using Hibernate API

Object Identity; Generator type classes

Using SQL with Hibernate

Using HQL

Using Criteria queries

**4. Mapping Collections and Associations**

To define sets, mas, lists in Hibernate

Association Mappings: 1. One to one

2 One to many

3 Many to one

4 Many to many

**5. Hibernate Caching**

What is caching?

What are the types of caching in Hibernate?

Explanation of various caching mechanisms in Hibernate

**6. Using Hibernate Annotations (if time permits)**

Sample example of using Hibernate Annotations

**Spring Framework (version 3.x)**

**1. Introduction to spring**

What is Spring?

Spring Architecture explanation and all it's components

**2. Introduction to all modules of spring**

Spring Bean Factory

Spring Application Context

Spring DI

Spring AOP

Spring Integration; Spring messaging, Spring JMS

Spring MVC

Spring DAO

**3. Setting up spring**

Setting up of Spring framework

Download JARs

Configure XML files

**4. Dependency Injection**

What is Dependency Injection?

How is it implemented using Spring Framework?

Bean Wiring mechanisms in Spring

**5. Spring AOP**

What is Spring AOP?

Implementation of Spring AOP