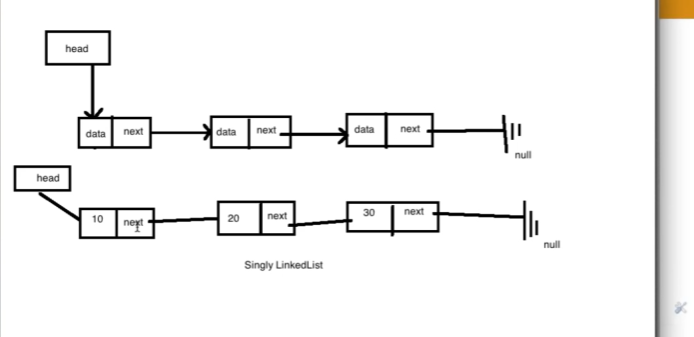


Singly linked list :



Final, finally and finalize:

Final is a keyword used for the variables, methods and class, where as the finally is a block of code which is executed always along with any exceptions, finalize is the method which is called automatically before the garbage collector method.

final is a keyword, finally is a block and finalize i a method.

here the finalize method is called without any reference to the main method coz

finalize will be called every time before the garbage collector system.gc method

The finalize method is called to clean up the process,just before the garbage collector method .or clean up the empty memory which is not used any any reference

Constructors:

**Constructor** is a block of code that initializes the newly created object. A **constructor** resembles an instance method in java but it's not a method as it doesn't have a return type. ... People often refer **constructor** as special type of method in Java.

Constructor is the class entity which is used to define some class feature while creating the object. It looks like function but its not a function. Contructor name is a same class name with no retun type.

Types of construct:

Default constructor.

For

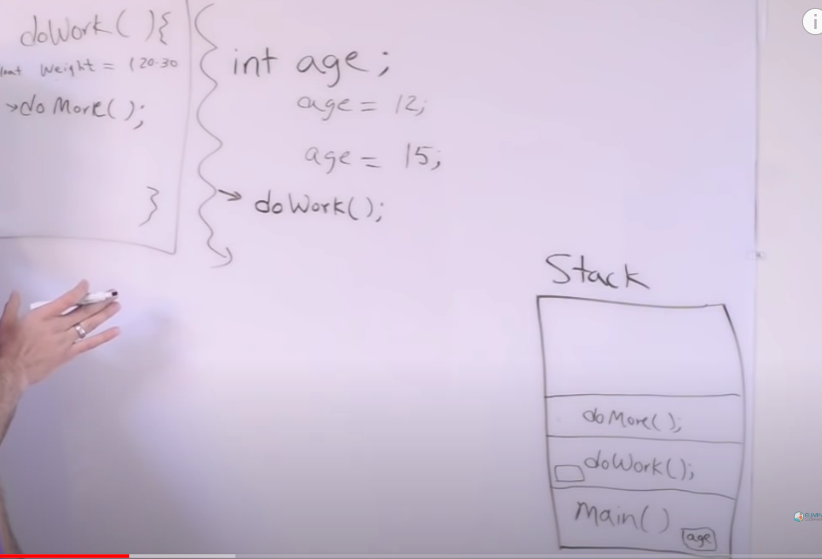
This variable is used to initialize the class variables in the constructor.

In inheritance, if there is a parent class and child class constructors, then while creating the object of the child class, the parent class is called default.

Super keyword should be the first statement in the constructor of the child constructor.

WE cannot have two super keywords inside one block of child constructor.

Heap and stack memory : <https://www.youtube.com/watch?v=UcPuWY0wn3w>

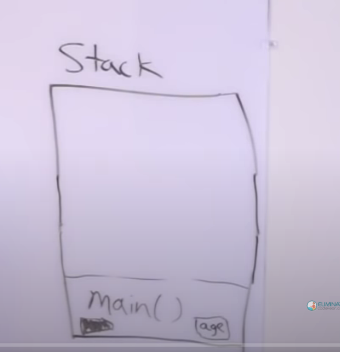


Stack concept is that it maintains the method invocations(methos inside of methods) compiling on top of each others .

Here when the doMore is done , its erased from the stack and then the control goes to the DoWork().

And once the DoWork is done , the memory inside the stack of DoWork is erased and moved to the next line of variables or methods or whatever is there after that DoWork ().

When that internal method is complete, its gonna pop of the stack.and when the application is done running , we are done with the main method too, and main is also gone from the stack and all the memory is clear for us to run other programs.

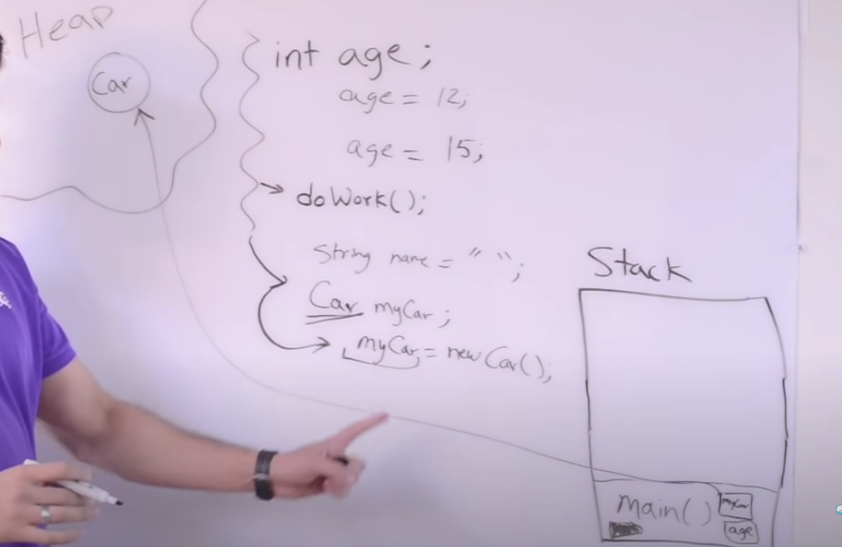


Now moving to the next program main method which has a variable car.This variable is not the primitive data type but it’s a complex data type which is defined as

Car myCar (at this point we don’t have any object)

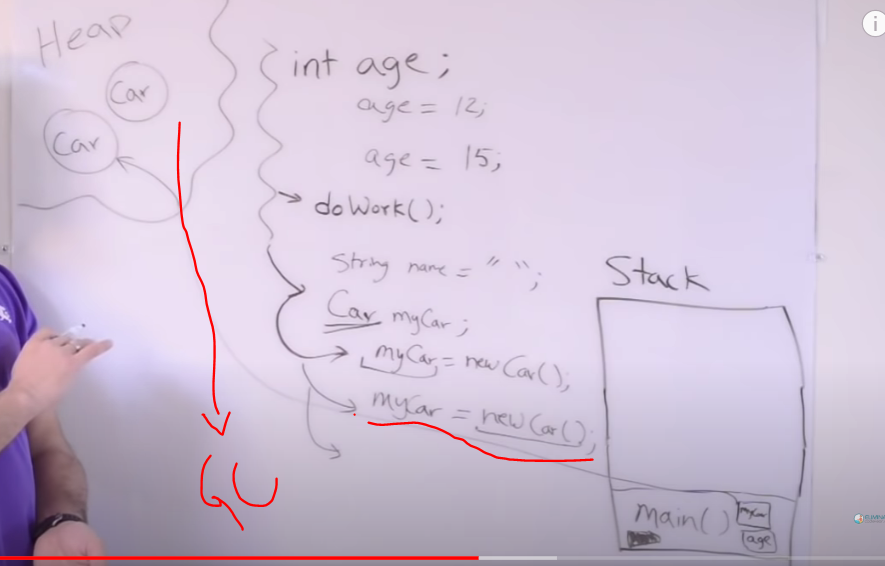
myCar = new Car();   
9at this pount we have object which is created in heap

Here myCar is not the object but its called the reference variable which is created inside the stack of the main method. And the new Car(); is the object which is created in other space called heap



Now when the control moves to the next line where myCar is pointing to the

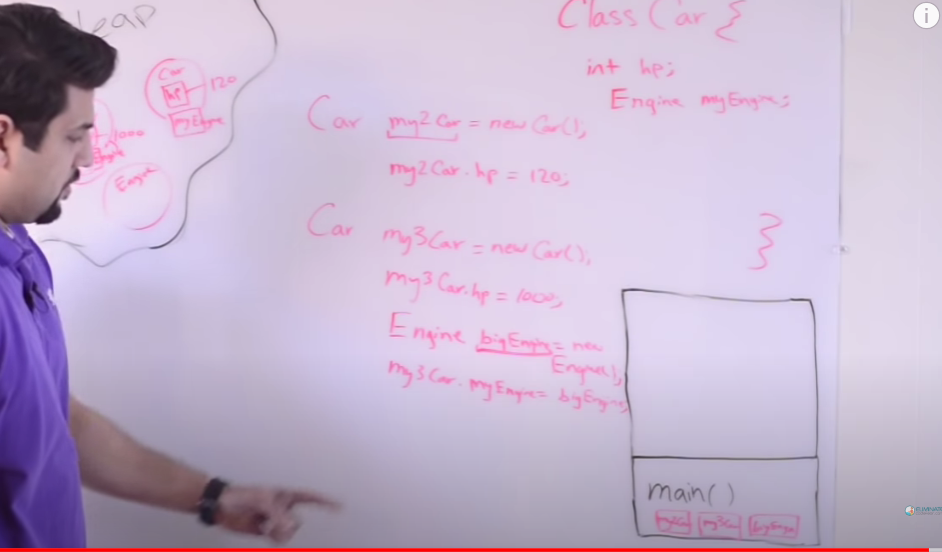
next object new Car() , the earlier object is lost control to the new object Car() . Now here comes the concept of the garbage collection



So the old car control is completely lost connection with it . So here garbage collection comes in picture.

GC is a process which runs in heap and every so often it comes and looks for the object that don’t have any relation to them or any variable pointing to them and clears the space in the memory.The whole process of GC is that we can save memory for the future variables.

Now we have seen the primitive data types and the object reference variables till now ,Now we are going to see about the instance variable ( variable that belongs to some instance)



Naveen concept of heap and stack :

<https://www.youtube.com/watch?v=aAjkJW08BGQ&list=PLFGoYjJG_fqqyIj2ht0aHMx_HnGX3ZFEx&index=21>

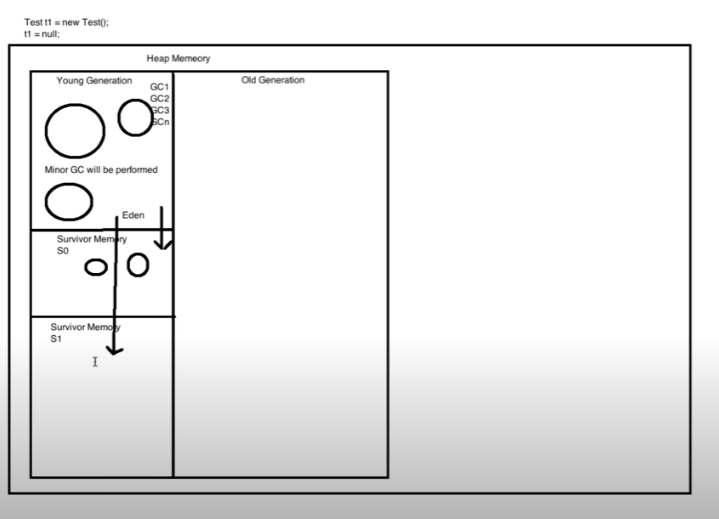
Here the heap is divide into two parts : Young generation and old generation;

Inside young generation we have the 3 parts : Eden , Survivor memory S0 and survivor memory S1. Whenever the objects are created they move to the eden-Heap memory.

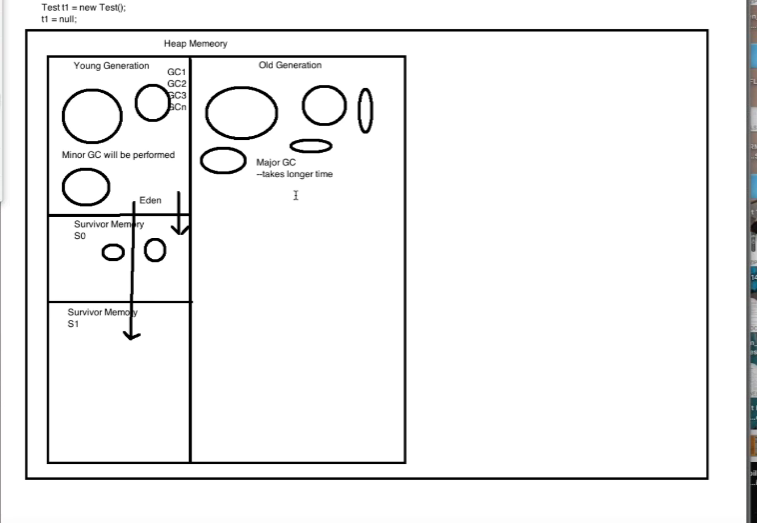
In young generation , all the class variables will be stored .After its filled , then the survivor memory is filled and then the S1.

This hEAP MEMORY IS ONLY AND ONLY for the class objects.

This eden memory will have its own space and once the eden is full, one mechanism called minor GC will be called automatically and this GC will shift some objects from eden memory to the S0 and S1.Now when S0 and S1 are also filled .This is called the basic fundamental of the young generation .



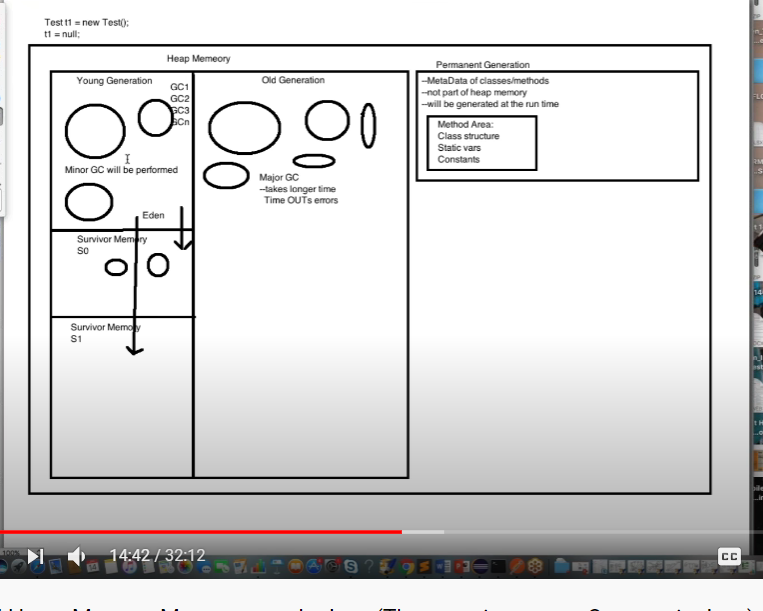
Now in the next concept we have the old generation. Here when the S0 and S1 are also filled, java will shift the new into the old generation .When the old generation memory is also filled up , then the major GC is called which takes longer time and is very slow. Major GC will degrade the application performance. It should be minimized. It will give the timeout errors



Minor GC will take less time when compared to the major GC (to destroy the objects)

So major GC activity should be avoided. Because of this it will give the timeout errors.

There is another part of heap memory called memory pool. This is used to store the immutable objects called string class or string pools. This will be defined at the runtime by JVM `memory managers.

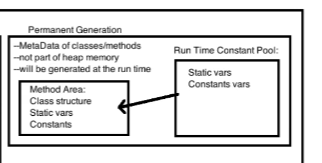


. Now there is another memory called permanent generation where the metadata (data of classes and methods is saved). This is not the part of the heap memory and is generated at the run time.

There is also another area called method area which is part of the permanent generation.

The complete class structure will be stored here. All static/constant variables.

There is another concept called runtime constant pool which stores the part of the method variables like the constant variables, this is the part of the method area.



So java heap memory stores:

1. All the class objects,
2. all JRE classes.
3. The GC runs on heap memory only to free some memory by destroying some objects which don’t have any references

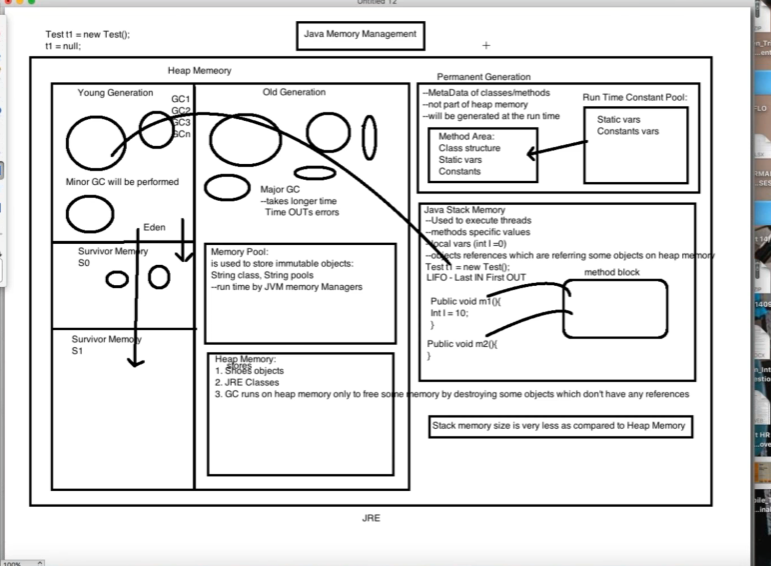
Java stack memory:

1. Used to execute the threads.
2. Method specific values
3. Local variables.
4. Object References which are referring to the objects in the heap memory.

Ex: Test t1= new Test(). SO here t1 will be store in the stack memory and the object(new Test()) will be store in the heap memory.

There is also a method block which is given to the methods executed one by one in the stack. Permanent generation is also a pert of the stack memory

Stack memory size is very less when compared to the heap memory



In short :

Stack memory is always used for method specific values, local/static/constant variables. Complete class structure is store in the stack memory.

Heap memory is always used to store the immutable or class objects.

Singleton concept in java :

<https://www.youtube.com/watch?v=1mEWYPasgUc&list=PLFGoYjJG_fqqyIj2ht0aHMx_HnGX3ZFEx&index=22>

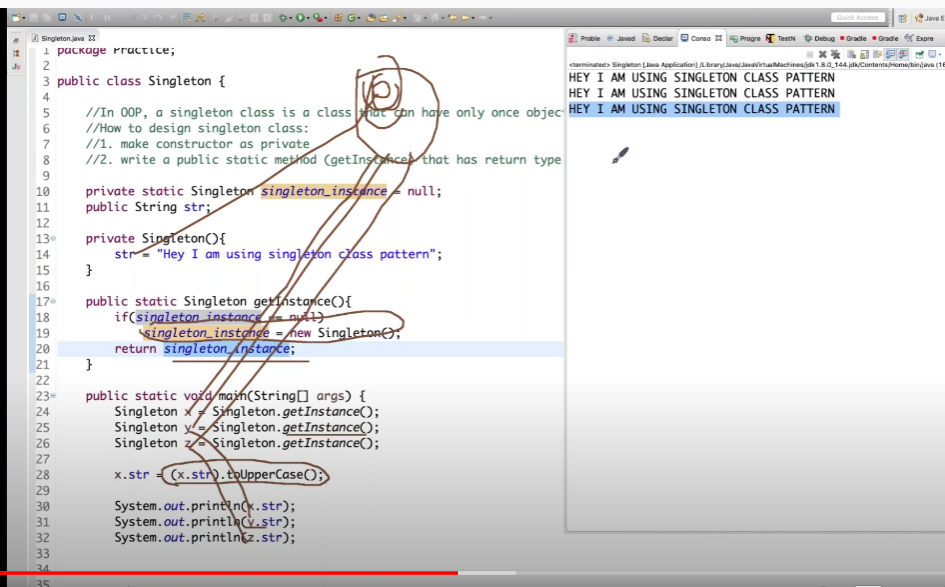
Differenece between Normal class and singleton class:

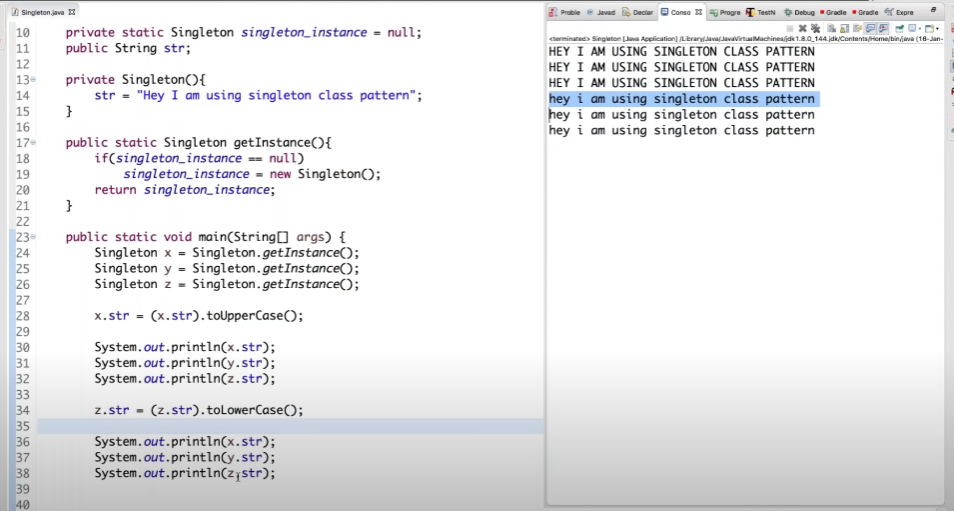
Normal class we use the constructor and we create no of constructors and many objects,where as for the singleton class we use getInstance method for instantiation.

Singleton class is the special class where we can have only one and one object at a time and if we try to create another object it will point to the same object.

Designing a singletom class:

1. Make the constructor private.
2. Write a public static method that has return type of object of singleton class(lazy initialization)





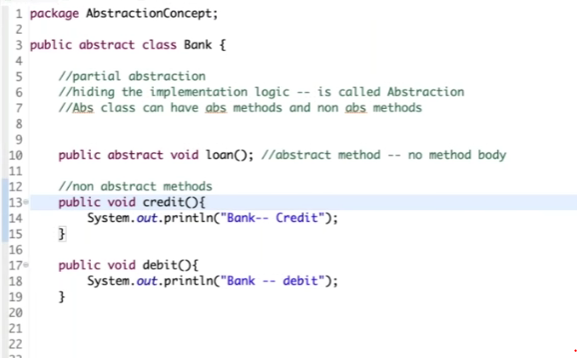
Functions :

Functions cant be created inside a function . Finctions are independent to each other.Functions are parallel to each other.But a function can be called inside a function.

Method overloading : When in same class , the function names are same but have different parameters

Absttract class: This class can have multiple abstract methods and multiple non-abstract methods.

But it have atleast one abstract method.cannot cretate object of the absatact class.



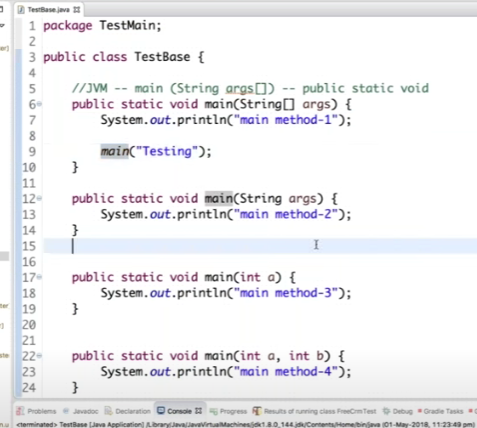
Interfaces:

Interfaces always define only the abstract methods.Abstarct methods just define function but no body.!00%abstraction is achieved in the interfaces. Cannot create the object of the interface. In interface the variables will be default final and static.

Whenver we want ot use the partial abstraction , we can use the abstfract class where as when we want to achieve 100% abstraction , then we use the interface.

We can override the main method in java :

JVM will try to search with the exact parameters public static void (String[] args)



Why is string immutable and final in java;

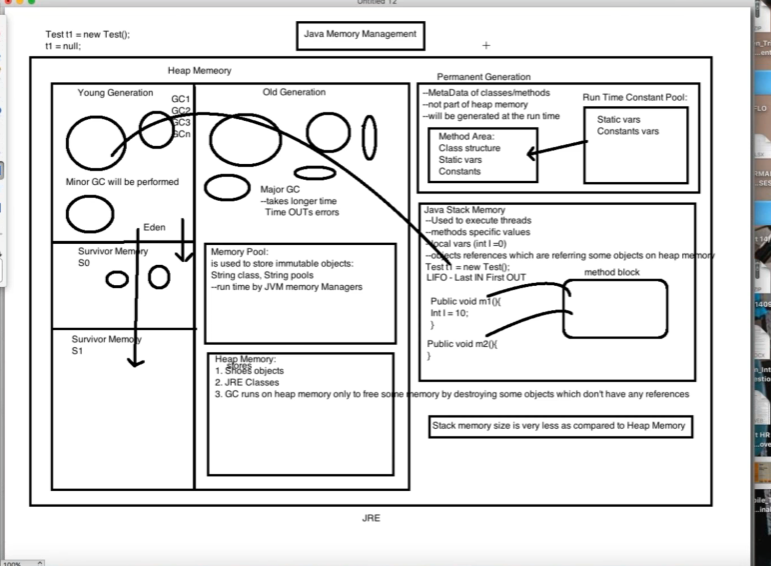
String is very powerful and frequent data type in java whih is available in the memory pool.

1) Imagine String pool facility without making string immutable , its not possible at all because in case of string pool one string object/literal e.g. "Test" has referenced by many reference variables, so if any one of them change the value others will be automatically gets affected i.e. lets say

2) String has been widely used as parameter for many Java classes e.g. for opening network connection, you can pass hostname and port number as string, you can pass database URL as a string for opening database connection, you can [open any file in Java](http://javarevisited.blogspot.sg/2012/07/read-file-line-by-line-java-example-scanner.html) by passing the name of the file as argument to File I/O classes.

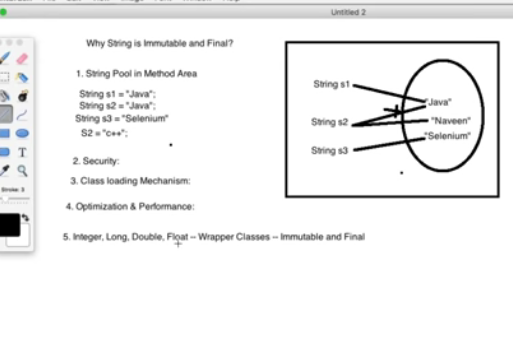
3)Since String is immutable it can safely share between many threads which is very important for multithreaded programming and to avoid any synchronization issues in Java, Immutability also makes String instance thread-safe in Java, means you don't need to synchronize String operation externally.

4)Optimisationa and performance: Another reason of **Why String is immutable in Java** is to allow String to cache its hashcode, being immutable String in Java caches its hashcode, and do not calculate every time we call hashcode method of String, which makes it very fast as hashmap key to be used in hashmap in Java.



The absolutely most important reason that String is immutable is that it is used by the [class loading mechanism](http://javarevisited.blogspot.sg/2012/07/when-class-loading-initialization-java-example.html), and thus have profound and fundamental security aspects.

Class loading mechanism : In some classes if we want to read some info as filename , it can be changed by any hacker and changed if the strings are not finlal



Can static class be overridden ?

Static class can be overloaded by cant be override.

What is method hiding in Java and how to use it? When super class and sub class contains same method including **parameters** and if they are static. The method in the super class will be hidden by the one that is in the sub class. This mechanism is known as method hiding

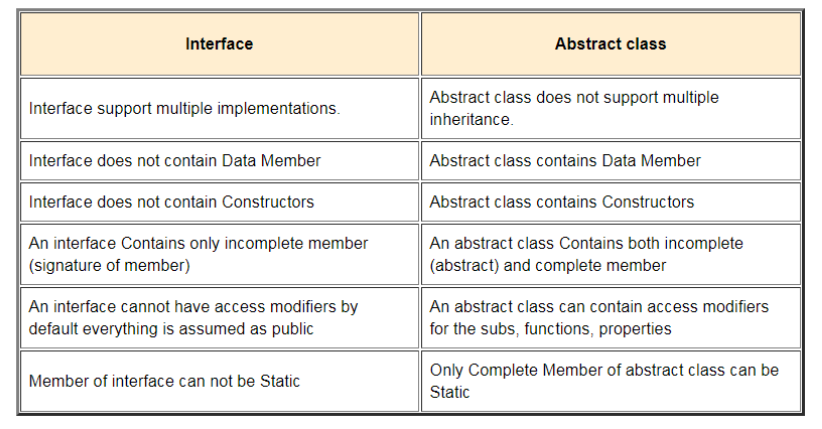
Interfaces vs abstract classes:

A particular class can have multiple parent interfaces but it cant have multiple parent classes.

**An interface is an empty shell**. There are only the signatures of the methods, which implies that the methods do not have a body. The interface can't do anything. It's just a pattern.

Exception :

Any unwanted beahviours of the program which terminates the code



//diff btwn : String and StringBuffer : String is immutable and doesnt have reverse

//function whereas StringBuffer is mutable and have reverse function

String can be mutable by using the string buffer and string builder.

StringBuffer is a class which provides the string mutablilty