**Important Points**

**Note-1:**

**Instance variable** : A variable which is created inside the class but outside the method is known as an instance variable**.** Instance variable doesn't get memory at compile time. It gets memory at runtime when an object or instance is created. That is why it is known as an instance variable.

**Note-2:**

The **new keyword** is used to allocate memory at runtime. All objects get memory in Heap memory area.

**Note-3:**

**Method** is like a function which is used to expose the behavior of an object.

Advantage of Method: Code Reusability, Code Optimization

**Note-4:**

There are 3 ways to initialize object in Java.

1. By reference variable {Student s= new Student() ; s.id=10; s.name=”VIcky”;}
2. By method { s.setValue(10,”vicky”) }
3. By constructor { Student s= new Student(1,”qwerty”) ;}

**Note-5:**

There are **many ways to create an object** in java.

1. By new keyword
2. By newInstance() method of Class class.
3. By newInstance() method of Contructor class.
4. By clone() method
5. By deserialization
6. By factory method etc.

**Note-6:**



As you can see in the above figure, **object gets the memory in heap memory area**. The **reference variable** refers to the object allocated in the heap memory area. Here, **s1 and s2 both are reference variables** that refer to the objects allocated in memory.

**Note-5:**

Each thread run in a separate call stack.

**Note-6:**

All objects in Java inherit a default implementation of hashCode() function defined in Objectclass. This function produces hash code by typically converting the internal address of the object into an integer, thus producing different hash codes for all different objects.

**Note-7:**

**Anonymous Object** simply means nameless. An object which has no reference is known as an anonymous object. It can be used at the time of object creation only.

If you have to use an object only once, an anonymous object is a good approach. For example:

**new** Calculation();//anonymous object

**Note-8:** The **javax.persistence**package contains the JPA classes and interfaces.

**Note-9:** The Hibernate architecture is categorized in four layers.

* Java application layer
* Hibernate framework layer
* Backhand api layer
* Database layer

**Note-10:**

**Comparable -**

1. public int compareTo(Object obj)
2. java.lang package
3. sort : String Object, Wrapper Class Object , User Defined Class
4. It provides a single sorting sequence only you can sort elements only single data member.

Example: It may be roll no, name, age or anything.

**Comparator –**

1. public boolean equals(Object obj), public int compare(Object ob1, Object ob2)
2. java.utils package
3. Sort: only User defined object
4. It provides multiple sorting sequences i.e. you can sort the elements on the basis of any data member

Example: it will be roll no, age , name etc.

**Note-11:**

**Collections Class:**

1. Collections class provide static method of sorting the element of collections.
2. If collection elements are of set or map type then we can use TreeSet or TreeMap.
3. However we cannot sort of the elemnts of list. Then Collection class provides method for sorting the element of list type elements.
4. Public void sort(List lst) : List must be comparable type.
5. String class and Wrapper class implements comparable by default. So if you store objects of String or wrapper class in a list, set, map. It will be comparable by default.

**Note-13:**

Scanner Class found in java.utils.scanner package

**Note-14:**

project POM (Project Object Model) file.

**Ducat Notes Rules:**

**Rule 1:** Runtime memory Allocation of class portion is Called Dynamic memory allocation and only one time its happen.

**Rule 2:** The portion of a class which is getting memory at class loading time that can get a memory once in a life cycle of a class this concept is known as static memory Allocation.

**Rule 3:** All the function are getting memory at class loading time and in which area these functions are getting memory are called method area.

**Rule 4:** **new Keyword:**  this operator allocate memory to class data member or instace variables at runtime in a heap.

**Rule 5:** The memory which has been allocated at runtime that has no name.

**Rule 6: Always** keep those behavior of an object as a instance function of a class which have to be performed a different task for particular object.

Note: String s1=”Hello”;

String s2=new String(“Hello”); there are 2 object create in memory one in heap and one in constant pool and one reference created in stack

s1 and s2 hashcode are same

(s1==s2) compare the object reference so result comes false

(s1.equals(s2)) compare the hashcode so result comes true

(s1.hashCode()==s2.hashCode()) true

s1 == s2.intern(); // s2.intern return the ref-id so result comes true

String s3=new String(“Ass”);

String s4=new String(“red”);

String s5=s4

If(s3==s4)// false

If(s4 == s5) true // because two ref ponting to shme object

If(s3.equals(s4)) //false

If(s4.equals(s5)) TRUE

}

}

# Java String intern()

The **java string intern()** method returns the interned string. It returns the canonical representation of string.

It can be used to return string from memory, if it is created by new keyword. It creates exact copy of heap string object in string constant pool

# Java Garbage Collection

In java, garbage means unreferenced objects.

Garbage Collection is process of reclaiming the runtime unused memory automatically. In other words, it is a way to destroy the unused objects.

To do so, we were using free() function in C language and delete() in C++. But, in java it is performed automatically. So, java provides better memory management.

### Advantage of Garbage Collection

* It makes java **memory efficient** because garbage collector removes the unreferenced objects from heap memory.
* It is **automatically done** by the garbage collector(a part of JVM) so we don't need to make extra efforts.

## How can an object be unreferenced?

There are many ways:

* By nulling the reference
* By assigning a reference to another
* By anonymous object etc.

**Note-7:** Every class in java by default extends the object class.

**Note 8**: In object class have 11 methods

**Note-9:** hashcode method declare in object class. Hashcode() is a native method. That’s gives integer value.

Each object hash unique no.

**Note:10:** Native Methods: Native methods are the method that doesn’t have implementation and written in other languages.

Public native in hashcode(); Never force to implement because semicolumn are their.

**Note-11: hashcode() return value is not a memory address.**

**Hashcode() use memory address to calculate the unique no. but it is not exactly the memory address.**

**But Its being used memory address to calculate unique address.**

Note:12- In Java, Method Overloading is not possible by changing the return type of the method only.

### Instance variable in Java

A variable which is created inside the class but outside the method is known as an instance variable. Instance variable doesn't get memory at compile time. It gets memory at runtime when an object or instance is created. That is why it is known as an instance variable.

### Method in Java

In Java, a method is like a function which is used to expose the behavior of an object.

#### Advantage of Method

* Code Reusability
* Code Optimization

### new keyword in Java

The new keyword is used to allocate memory at runtime. All objects get memory in Heap memory area.

=======================

### Why String class is Final in Java?

The reason behind the String class being final is because no one can override the methods of the String class. So that it can provide the same features to the new String objects as well as to the old ones.

# Can we start a thread twice

No. After starting a thread, it can never be started again. If you does so, an IllegalThreadStateException is thrown. In such case, thread will run once but for second time, it will throw exception.

* Each thread starts in a separate call stack.
* Invoking the run() method from main thread, the run() method goes onto the current call stack rather than at the beginning of a new call stack.

class TestCallRun1 extends Thread{

public void run(){

System.out.println("running...");

}

public static void main(String args[]){

TestCallRun1 t1=new TestCallRun1();

t1.run();//fine, but does not start a separate call stack

}

}

Output : running…

#### Note: If you want to make a user thread as Daemon, it must not be started otherwise it will throw IllegalThreadStateException.