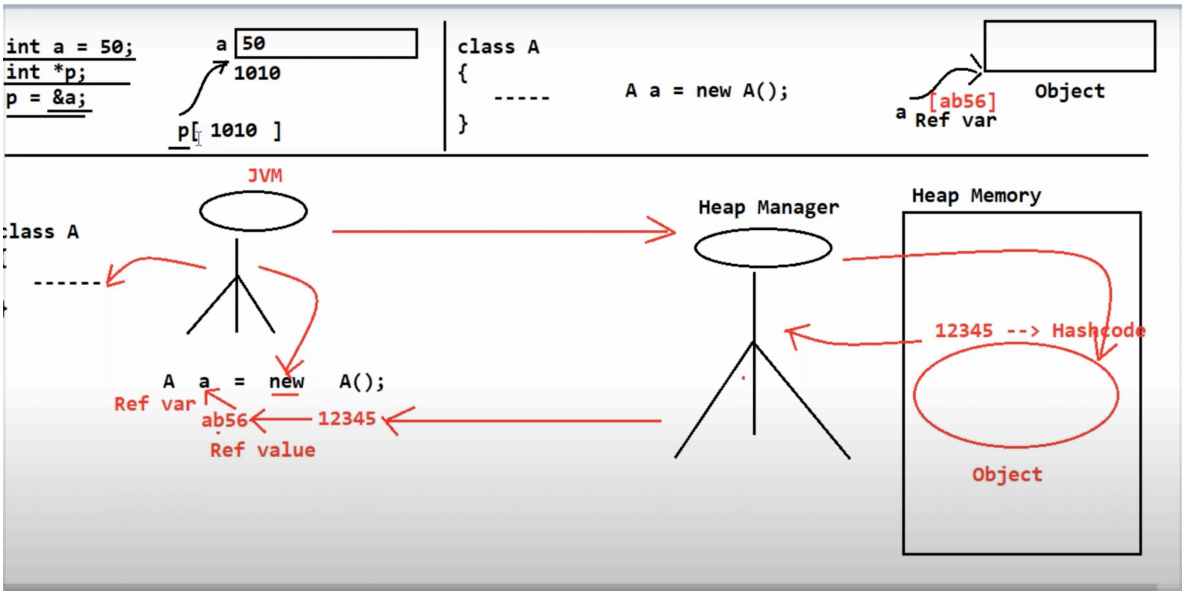
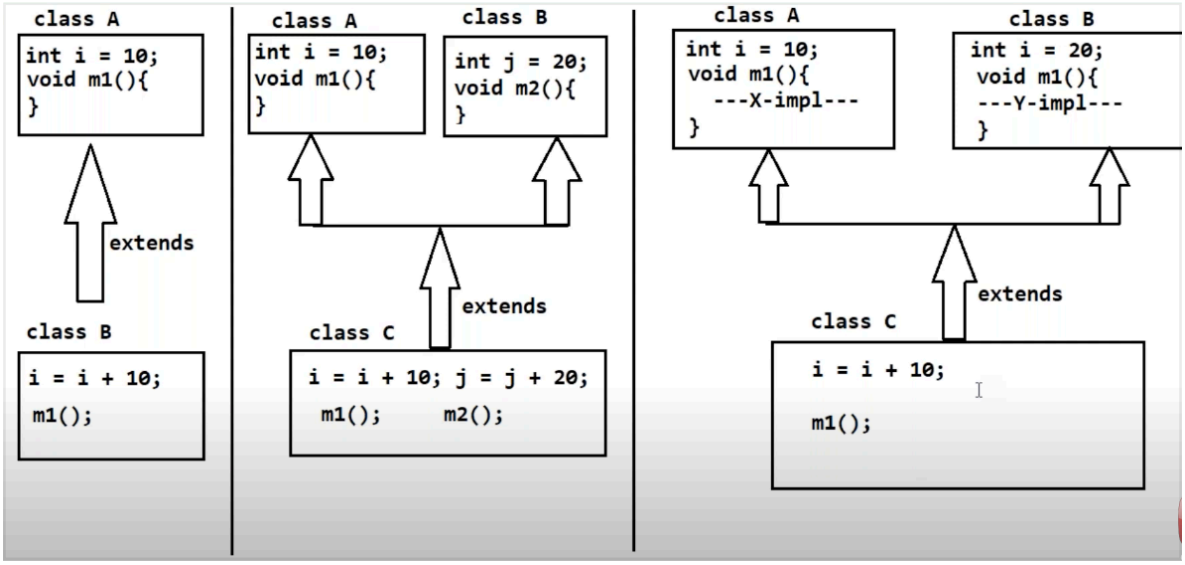
Inheritance OOPs

```
Sugg:
-----
1. In Java applications, it is suggestible to provide identifiers with a particular
meaning.
String xxx = "abc123"; ---> Not Suggestible
String accNo = "abc123";---> Suggestible

2. In Java applications, no length restriction for the identifiers , we can write
identifiers with any length, but, it is suggestible to provide identifiers with
around 10 symbols
String temporaryemployeeaddress = "Hyd"; --> Not Suggestible
String tempEmpAddr = "Hyd"; ----> Suggestible

3. If we have muiltiple words in single identifier then it is suggestible to
seperate multiple words wit hspecial notations like '_' symbols.
EX:
String tempEmpAddr = "Hyd"; -----> Not Suggestible
String temp_Emp_Addr = "Hyd";----> Suggestible
```

```
1. Pointer variables are existed in C and C++.
   Reference variables are existed in JAVA.

2. Pointrer variables are able to refer a block of memory by storing address
   locations.
   Reference Variables are able to refer a block of memory by storing objecxt
   reference value, where Object reference value is not an addres location, it is
   hexa decimal form of Hashcode, where Hashcode is an unique identity provided by
   heap manager in the form of an integer.

3. Pointer variables are recognized and initialized at compilation time.
   Reference Variables are recognized and initialized at runtime.
```




```
class Employee{
  eno, ename, esal, eaddr
  void setEmpDetails(){   }
  void getEmpDetails(){  }
}

class Manager extends Employee{
   --Use Employee class members---
}

class Accountent extends Employee{
  --Use Employee class members---
}

class Engineer extends Employee{
  ---Use Employee class members---
}
```




```
5. Multiple Inheritance is not possible in JAVA:
-------------------------------------------------
Object Oriented Features.

1. Class
2. Object
3. Encapsulation
4. Abstraction
5. Inheritanc
6. Polymorphism
7. Message Passing

Inheritance: It is a relation between classes, it will bring variables and methods
from one class[Super class ] to another class[ Sub class].

class Employee{// Super
  eno, ename, esal, eaddr
  void setEmpDetails(){   }
```

```
The main adv of Inheritance is "Code Reusability".


1. Single Inheritance
2. Multiple Inheritance

1. Single Inheritance:
-----------------------
--> It is a relation between classes, it will bring variables and methods from only
one super class to one or more no of sub classes.

2. Multiple Inheritance
------------------------
--> It is a relation between classes, it will bring variables and methods from more
than one super class to one or more no of sub classes.

class A{
}
class B{
```

Inheritance: It is a relation between classes, it will bring variables and methods from one class[Super class] to another class[Sub class].

```
class Employee{// Super
  eno, ename, esal, eaddr
  void setEmpDetails(){   }
  void getEmpDetails(){  }
}

class Manager extends Employee{// Sub Class
   --Use Employee class members---
}

class Accountent extends Employee{// Sub Class
  --Use Employee class members---
```

In c and C++ , memory allocation for primitive data types is not fixed, it is variable depending on the Operating System which we used.

In Java, memory allocation for the primitive data types is fixed irrespective oif the OS which we used.

```
byte ------------> 1 byte
short -----------> 2 bytes
int -------------> 4 bytes
long ------------> 8 bytes
float -----------> 4 bytes
double ----------> 8 bytes
char ------------> 2 bytes
boolean ---------> 1 bit
```

In c and C++, if we pass pointer variable as parameter to methods / Functions then the parameter passing mechanism is "Call By Reference" only, why because, in C and C++ pointer variables are able to store address locations.

In Java, even if we pass reference variable as parameter to the methods then the parameter passing mechanism is Call by Value only, not call by reference, why because, in jAVA , REFERENCE VARIABLES are not storing address locations, reference variables are storing Object reference values, where Object reference value is hexa decimal fomr of hashcode, where Hashcode is an unique identity for the objects provided by Heap manager in the form of an integer value.

JAVA

---

1. J2SE / JAVA SE
--> Java 2 Standard Edition
--> It will provide only Fundamentals of JAVA programming Language.
--> Standalone Applications.
--> If we design and execute any application with out using client-Server arch or with out distributing application logic over multiple machines then that applications are called as Standalone Applications.

8. C and C++ are following Call By Value and Call By Reference Parameter Passing mechanisms, but, JAVA is following Call By Value Parameter Passing mechanism:

In any PL, if we pass primitive data like byte, short, int, long, float, double, boolean, char as parameters to the methods then the parameter passing mechanism i call by Value parameter passing mechanism

IN any PL, if we pass address locationsd as parameters to the methods then the parameter passing mechanism is "Call By Reference".

JAVA

---

1. J2SE / JAVA SE
--> Java 2 Standard Edition
--> It will provide only Fundamentals of JAVA programming Language.
--> Standalone Applications.
--> If we design and execute any application with out using client-Server arch or with out distributing application logic over multiple machines then that applications are called as Standalone Applications.

2. J2EE / JAVA EE:
--> Java 2 Enterprise Edition.
--> It will provide Server side programming
--> Enterprise Applications.
--> If we design and execute any application on basis of client-Server arch or by distributing application logic over multiple machines then that applications are called as Enterprise Applications.
--> 95% of the applications are Enterprise Applications

3. Java Features:

To describe the nature of Java programming Language, JAVA has provided the following features.

1. Simple
2. Object Oriented
3. Platform Independent
4. Arch Nuetral
5. Portable
6. Robust
7. Secure
8. Dynamic
9. Distributed
10. Multi Threadded
11. interpretive
12. High Performance

2. Java has not allowed all the confusion oriented features l
   Inheritance,.....
3. Java is getting all the simplified syntaxes from C aned C+

2. Object Oriented :

File Edit Format View Help

1. Simple :
---
1. Less memory and Less Execution Time
2. Java has not allowed all the confusion oriented features like Pointers, Multiple Inheritance,.....
3. Java is getting all the simplified syntaxes from C aned C++.

File Edit Format View Help

2. Object Oriented :

Java is an Object oriented PL, it able to store data in the form of Objects as per Object Oriented Features.

3. Platform Independent:

Java is a platform indeoendent PL, because, Java allows its applications to perform compilation is on one OS and execution is on another OS.

4. Arch Nuetral:

Java is an Arch Nuetral PL, becasue, JAVA allows its applications to perform compilation is on one H/W Arch and execution is on another H/W Arch.

5. Portable:

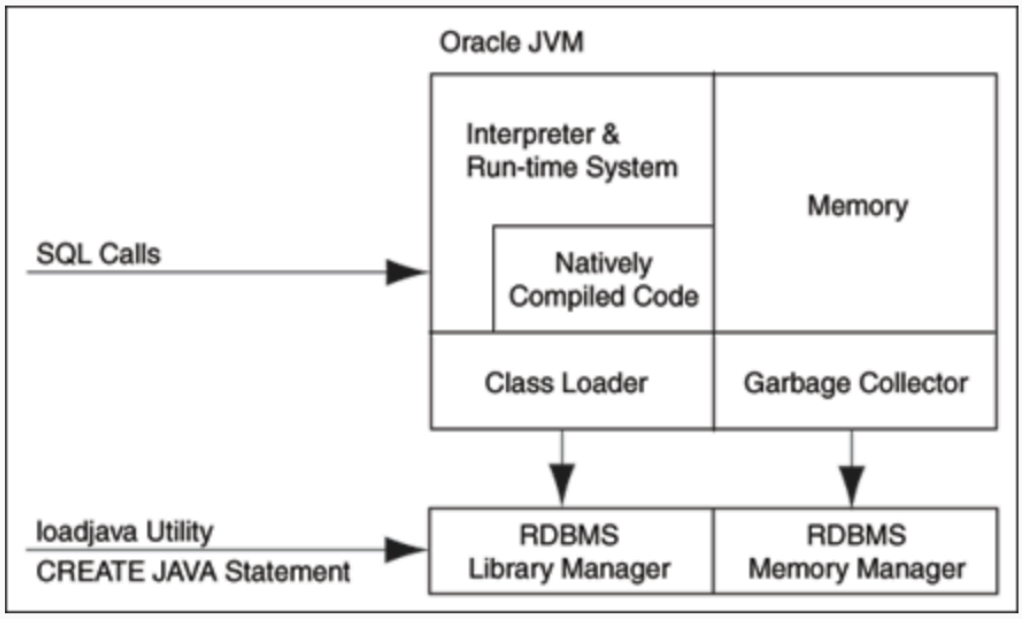
5. Portable:

Java is portable PL, because,
1. Java is able to execute its applications under all the OS and under all the H/W Systems.
2. Java is able to provide fixed memory allocations for the data types irrespective of the OS and irrespective of the H/W which we used.

6. Robust

```
6. Robust:
----------
Java is a Robust PL, because,

1. Java has very good memory management system in the form of Heap Memory
management System, it is a dynamic memory management System, it allocates and
deallocates memory for the objects at runtime.

2.JAVA has very good exception handling mechanisms, because, JAVA has provided very
good predefined library to represent and handle almost all the exceptions which are
coming in java applications.


7. Secure
```

Figure 1-8 Main Components of an Oracle JVM



Description of "Figure 1-8 Main Components of an Oracle JVM"

```
7. Secure:
-----------
1. Implicit Security : Security Manager in JVM
2. Web Security : JAAS[JAVA Authentication And Autherization Service]
3. Network Security : Predefined implementations for almost all the Network
                      security alg.
```

```
8. Dynamic:
-----------
Java is a Dynamic PL, because, JAVA allows memory allocation for the primitive data
types at runtime.

9. Distributed:
```

9. Distributed:

Java is a distributed PL, because, to preparer Distributed applications JAVA has provided a seperate module in the form of J2EE module

10.Multi Threadded:

Java is a Multithredded PL, because, Java allows to create and execute more than one thread at a time .

11. Interpretive:

Java is both compilative and interpretive PL, because,
1. To check developers mistakes in Java programs and to translate java program from High level representation to low level representation we need compilation.
2. To run JAVA programs, we need an interpretor inside JVM.

12. High Performance:

JAVA is high performance PL, because of its rich set of features like Portable, PI, Robust, dynamic, Multi Threadded, Arch Nuetral,.....

JAVA has the components like JIT Compiler to improve performance of java applications .