1. Explain in detail the features of Java Programming.

Simple:

* syntax is based on C and C++.
* No global Variables.
* No goto statements.
* No pointers.
* No unsafe Structures
* No operator overloading

**Object-Oriented**

It supports the concept of object oriented programming

**Platform independent**

* A platform is a hardware or software in which a program runs.
* Java code can be run on multiple platforms e.g. Windows, Linux, Sun Solaris, Mac/OS etc.
* Java code is compiled by the compiler and converted into byte code.
* This byte code is a platform independent code because it can be run on multiple platforms i.e. Write Once and Run Anywhere(WORA).

**Secured**

* Java is secured because:
* No explicit pointers.

Programs run inside virtual machine

**Robust**

* Robust simply means strong.
* Java uses strong memory management, automatic garbage collection, exception handling and type checking. All these points makes java robust.

**Architecture neutral**

* Neutral to the architecture of the system and supports “write once; run anywhere, any time, forever”.

**Dynamic**

It supports dynamic binding and dynamic allocation and deallocation of memory

**Interpreted**

* Java is both compiled and an interpreted language.
* First the java compiler translates source code into the byte code instructions.
* In the next stage the java interpreter converts the byte code instructions to machine code.

**High Performance**

Java is faster than traditional interpretation since byte code is "close" to native code

**Multithreaded**

A thread is like a separate program, executing concurrently.

We can write Java programs that deal with many tasks at once by defining multiple threads.

**Distributed**

We can create distributed applications in java.

RMI and EJB are used for creating distributed applications

1. What are the command line arguments used in Java? Write a java program, to demonstrate the use of command line arguments.

These are the arguments which we can pass from the command line at runtime.

The main method receives these arguments and stores as a string array.

Java application can receive any number of command line arguments ,when we are running the application from command prompt.

Example:

Class Commandline

{

Public static void main(String args[])

{

System.out.println(“The command line arguments are:”);

for(int i=0;i<args.length;i++)

System.out.println(args[]);

}

}

o/p:-1205 sushma 500

the commandline arguments are:-

1205

Sushma

500

4.What is the role and responsibility of JVM in program execution?

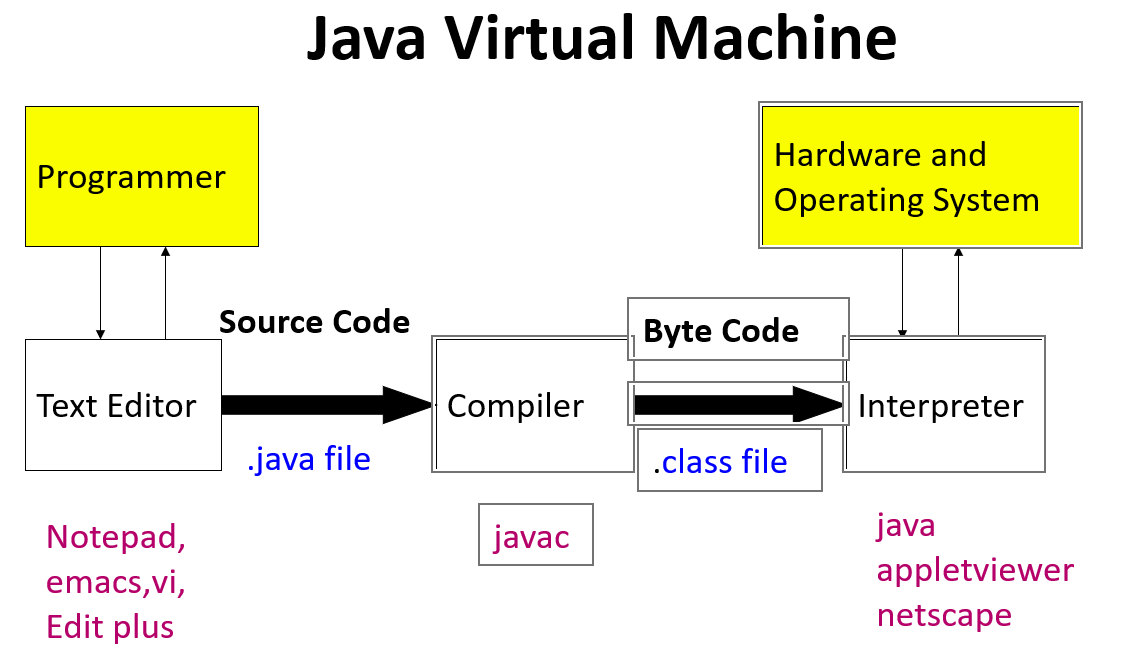
* The intermediate code namely the byte code produced by the java compiler is for a machine that exists only on the memory of a computer. This machine is known as the “Java Virtual Machine” or JVM.
* It provides runtime environment in which java byte code can be executed.
* JRE consists of the java virtual machine.

**What it does**

The JVM performs following operation:

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

JVM provides definitions for the:

* + - Memory area
    - Class file format
    - Register set
    - Garbage-collected heap
    - Fatal error reporting etc
  + Executes code

5.Explain the following: (i) super (ii) final (iii) Garbage Collection

iii)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.

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.

i)super- Super is a keyword of Java which refers to the immediate parent of a class and is used inside the subclass method definition for calling a method defined in the superclass.

Syntax-super.<method-name>();

Usage:-

* Super variables refer to the variable of a variable of the parent class.
* Super() invokes the constructor of immediate parent class.
* Super refers to the method of the parent class.

ii)Final-Final is a keyword in Java that is used to restrict the user and can be used in many respects. Final can be used with:

* Class
* Methods
* Variables

6)Write a java program to perform matrix multiplication?.pr

7)What is a string buffer class? Explain various methods associated with strings and string buffer

* Java StringBuffer class is used to create mutable(modifiable)string.
* StringBuffer allows growable and writable character sequence.
* Characters in StringBuffer can be inserted/appended/deleted any where and the size of the StringBuffer will automatically grow/shrink to make space.
* **StringBuffer Methods**
* **length():** Returns the StringBuffer object’s length.
* **capacity():** Returns the capacity of the StringBuffer object.
* **append():** appends the specified argument string representation at the end of the existing String Buffer.
* **insert():**It is used to insert text at the specified index position.
* **reverse():** Reverses the existing String or character sequence content in the buffer and returns it.
* **delete(**int startIndex, int endIndex):deletes the character sequence between startindex and endindex-1.
* **deleteCharAt(int index**): deletes single character within the String inside the buffer.

8)Write a Java program to implement constructor and constructor overloading.

Public class Home

{

Home()

{

System.out.println(“welcome Home”);

}

Home(String name){

System.out.println(“My name is”+name);

}

Home(String name,int age)

{

System.out.println(name+”is”+age+”years old”);

}

Public static void main(String args[])

{

Home obj=new Home();

Home a=new Home(“Sushma”);

Home b=new Home (“sravya”,20);

}

}

o/p:-

welcome Home

My name is Sushma

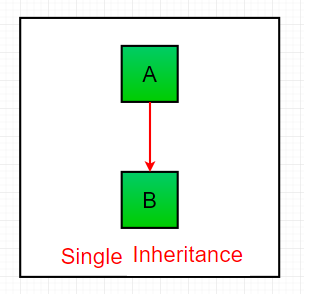
Sravya is 20 years old

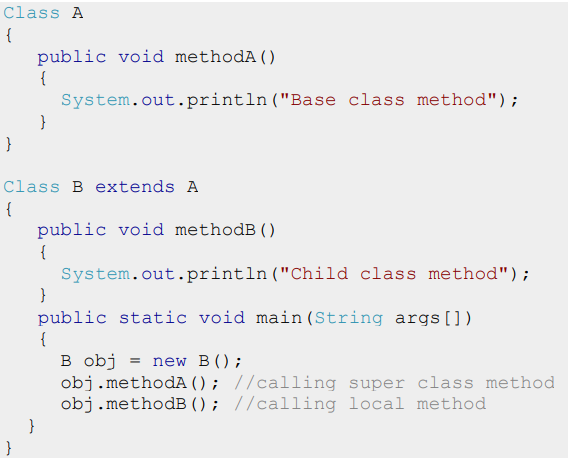
13)Explain various types of inheritance supported by JAVA with an example

.

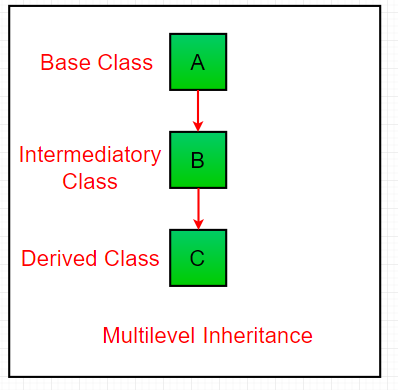
Below are the different types of inheritance which are supported by Java.

**1. Single Inheritance:**In single inheritance, subclasses inherit the features of one superclass. In the image below, class A serves as a base class for the derived class B.



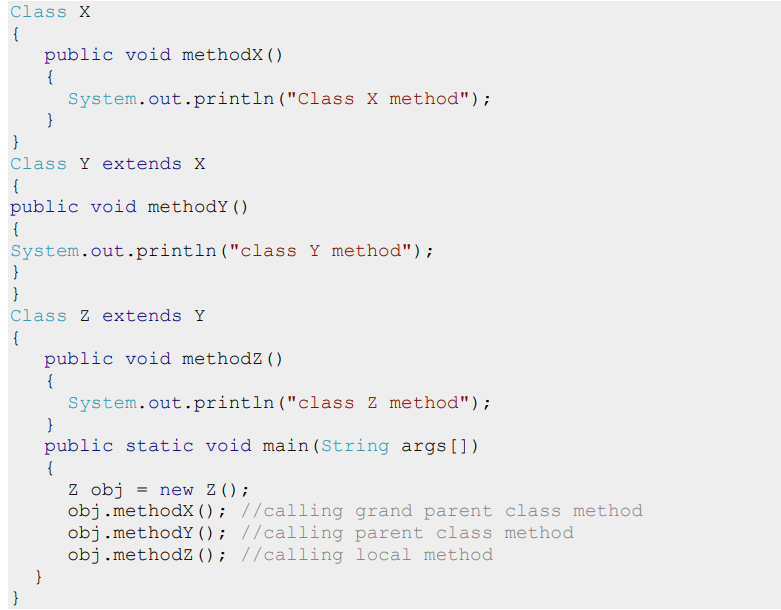


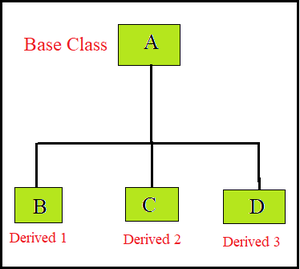
**Multilevel Inheritance:**In Multilevel Inheritance, a derived class will be inheriting a base class and as well as the derived class also act as the base class to other class. In the below image, class A serves as a base class for the derived class B, which in turn serves as a base class for the derived class C. In Java, a class cannot directly access the[grandparent’s members](https://www.geeksforgeeks.org/g-fact-91/).



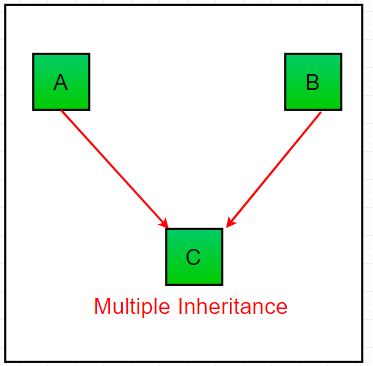
IT IS NOT SUPPORTED IN JAVA

1. **Hierarchical Inheritance:**In Hierarchical Inheritance, one class serves as a superclass (base class) for more than one subclass. In the below image, class A serves as a base class for the derived class B, C and D.





1. [**Multiple Inheritance**](https://www.geeksforgeeks.org/java-and-multiple-inheritance/)**(Through Interfaces):**In Multiple inheritances, one class can have more than one superclass and inherit features from all parent classes. Please note that Java does **not** support [multiple inheritances](https://www.geeksforgeeks.org/java-and-multiple-inheritance/) with classes. In java, we can achieve multiple inheritances only through [Interfaces](https://www.geeksforgeeks.org/interfaces-in-java/). In the image below, Class C is derived from interface A and B.



class A

{

public void methodA()

{

System.out.println("method of Class A");

}

}

class B extends A { public void methodB()

{

System.out.println("method of Class B");

}

}

class C extends A

{

public void methodC()

{

System.out.println("method of Class C");

}

}

class D extends A

{

public void methodD()

{

System.out.println("method of Class D");

}

}

class JavaExample

{

public static void main(String args[])

{

B obj1 = new B();

C obj2 = new C();

D obj3 = new D();

obj1.methodA();

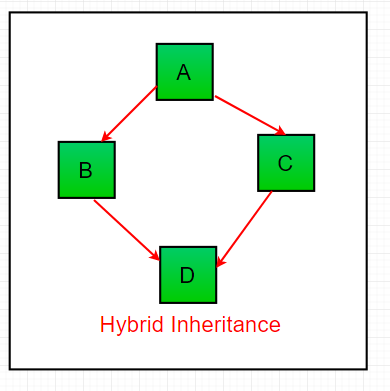
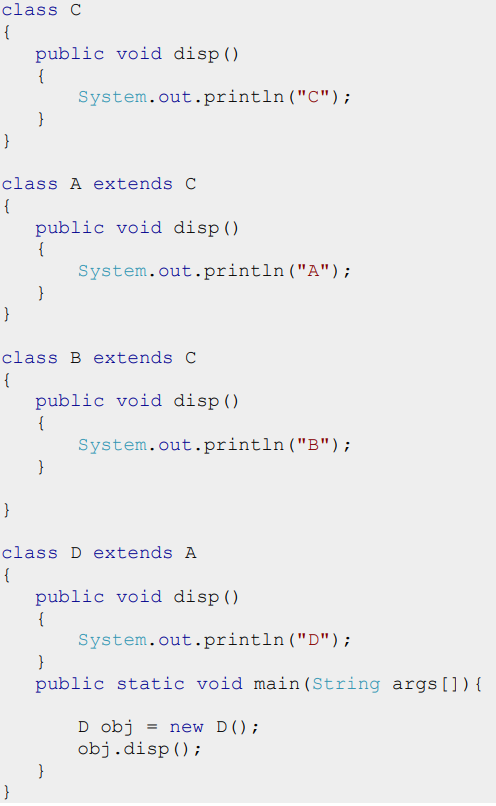
obj2.methodA();

obj3.methodA();

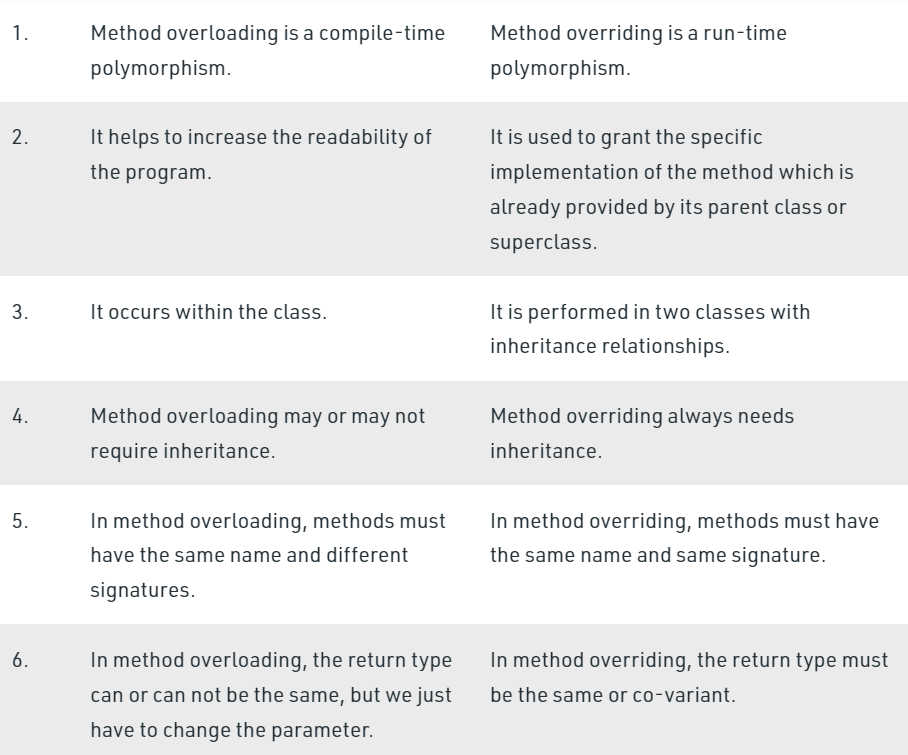
}

}

1. **Hybrid Inheritance(Through Interfaces):**It is a mix of two or more of the above types of inheritance. Since java doesn’t support multiple inheritances with classes, hybrid inheritance is also not possible with classes. In java, we can achieve hybrid inheritance only through [Interfaces](https://www.geeksforgeeks.org/interfaces-in-java/).

10)Differentiate between method overriding and method overloading



1. Discuss primitive data types in JAVA
   * + 1. boolean type

• The boolean data type has two possible values, either true or false.

• Default value: false.

2. byte type

• The byte data type can have values from -128 to 127 (8-bit signed two's complement integer). • Default value: 0

3.short type

• The short data type in Java can have values from -32768 to 32767 (16- bit signed two's complement integer).

• Default value: 0

4. int type • The int data type can have values from -231 to 231-1 (32-bit signed two's complement integer).

• Default value: 0 5. long type • The long data type can have values from -263 to 263-1 (64-bit signed two's complement integer).

• Default value: 0

6. double type

• The double data type is a double-precision 64-bit floating-point.

• Default value: 0.0 (0.0d)

7. float type •

The float data type is a single-precision 32-bit floating-point. • Default value: 0.0 (0.0f)

8. char type

• It's a 16-bit Unicode character.

• Default value: '\u0000

1. Illustrate String class methods with an example
2. [**int length()**](https://beginnersbook.com/2013/12/java-string-length-method-example/): It returns the length of a String.
3. [**char charAt(int index)**](https://beginnersbook.com/2013/12/java-string-charat-method-example/): It returns the character at the specified index
4. [**getChars(start, end, char target[ ], targetStart):**](https://beginnersbook.com/2013/12/java-string-charat-method-example/)It returns more than one character at a time.
5. [**boolean equals(String string)**](https://beginnersbook.com/2013/12/java-string-equals-and-equalsignorecase-methods-example/): Compares the string with the specified string and returns true if both matches else false.
6. [**boolean equalsIgnoreCase(String string)**](https://beginnersbook.com/2013/12/java-string-equals-and-equalsignorecase-methods-example/): It works same as equals method but it doesn’t consider the case while comparing strings.
7. [**int compareTo(String string)**](https://beginnersbook.com/2013/12/java-string-compareto-method-example/): This method compares the two strings based on the Unicode value of each character in the strings.
8. [**String substring(int beginIndex)**](https://beginnersbook.com/2013/12/java-string-substring-method-example/): It returns the substring of the string. The substring starts with the character at the specified index.
9. [**String substring(int beginIndex, int endIndex)**](https://beginnersbook.com/2013/12/java-string-substring-method-example/): Returns the substring. The substring starts with character at beginIndex and ends with the character at endIndex.
10. [**String concat(String str)**](https://beginnersbook.com/2013/12/java-string-concat-method-example/): Concatenates the specified string “str” at the end of the string.

14)Explain the use of ‘super’ keyword with an example.

i)super- Super is a keyword of Java which refers to the immediate parent of a class and is used inside the subclass method definition for calling a method defined in the superclass.

Syntax-super.<method-name>();

Usage:-

* Super variables refer to the variable of a variable of the parent class.
* Super() invokes the constructor of immediate parent class.
* Super refers to the method of the parent class.

1. **class** Animal{
2. String color="white";
3. }
4. **class** Dog **extends** Animal{
5. String color="black";
6. **void** printColor(){
7. System.out.println(color);//prints color of Dog class
8. System.out.println(**super**.color);//prints color of Animal class
9. }
10. }
11. **class** TestSuper1{
12. **public** **static** **void** main(String args[]){
13. Dog d=**new** Dog();
14. d.printColor();
15. }}

o/p:-

black

white

* 1. Explain the use of static keyword in java.

1. static keyword can be used with variable, method , block and nested classes.
2. When a member of the class is declared as static, it can be accessed before the objects of its class are created, and without any object reference.
3. The static is used for the following:
   1. variable (also known as class variable)
   2. method (also known as class method)
   3. block
   4. nested class

16)Explain what nested classes are. Explain different types of nested classes.

* In java, it is possible to define a class within another class, such classes are known as **nested classes.**
* **Syntax:**

class OuterClass

{ ...

class NestedClass

{

...

}

}

* A nested class has access to the members, including private members, of the class in which it is nested. However, reverse is not true i.e. the enclosing class does not have access to the members of the nested class.
* A nested class is also a member of its enclosing class.
* As a member of its enclosing class, a nested class can be declared private, public,  protected or default .
* Nested classes are divided into two categories:
* **static nested class :**Nested classes that are declared *static* are called static nested classes.
* We can access it without creating an object of the outer class

class OuterClass

{

int x = 10;

static class InnerClass

{

int y = 5;

}

}

public class MyMainClass

{

public static void main(String[] args)

{ OuterClass.InnerClass inner = new OuterClass.InnerClass(); System.out.println(inner.y);

}

}

* **inner class :**An inner class is a non-static nested class.
* To access the inner class, create an object of the outer class, and then create an object of the inner class.
* To create an object for the inner class, use the following syntax:

OuterClass outer = new OuterClass();

OuterClass.InnerClass

inner=outer.new InnerClass();

class OuterClass

{

int x = 10;

class InnerClass

{

int y = 5;

}

}

public class MyMainClass

{ public static void main(String[] args)

{

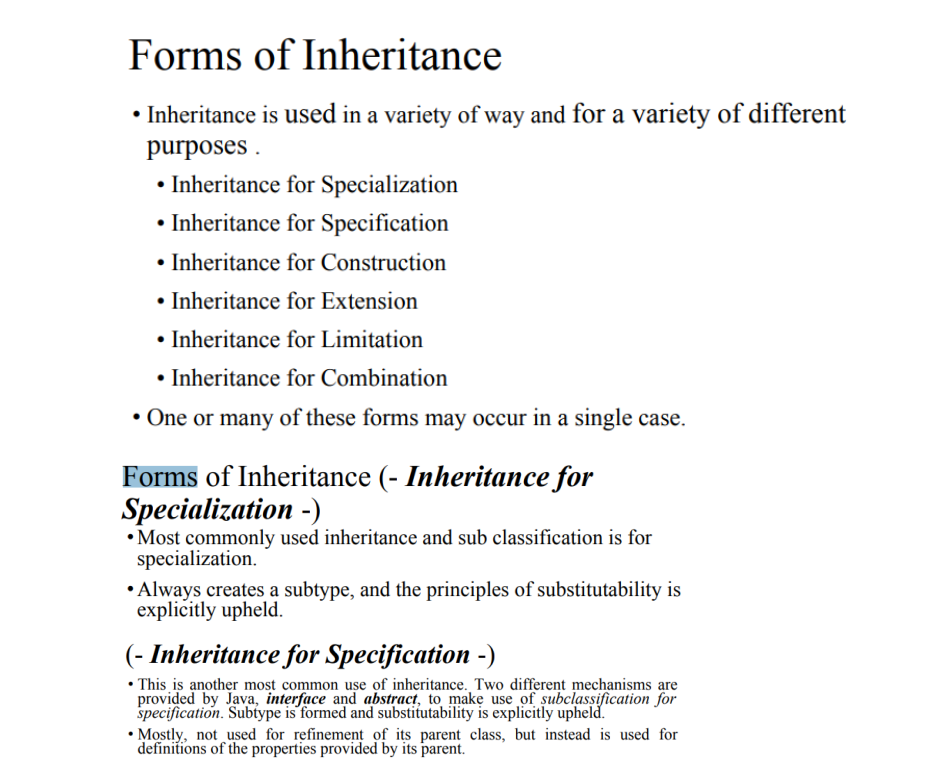
OuterClass outer = new OuterClass(); OuterClass.InnerClass inner = outer.new InnerClass();

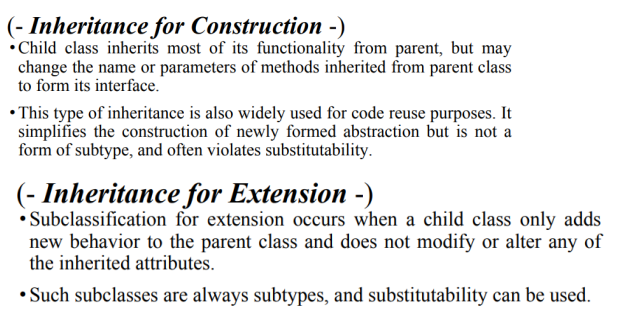
System.out.println(inner.y);

}

}

9)What are the benefits of inheritance? Explain various forms of inheritance.





3)Explain the building blocks of Java Programming

Statements – A statement is some action or sequence of actions, given as a command in code. A statement ends with a semi-colon (;).

Blocks – A block is a set of statements enclosed in set braces { }. Blocks can be nested.

Classes – A class is a blueprint for building objects in Java. Every Java program has at least one class. Programmers can define new classes There are many pre-built classes in the Java SDK Methods

Method – A method is a function (i.e. subroutine) that belongs to a class. In Java, all functions are methods, meaning they are always contained in some class A Java program can be made up of multiple classes, spread across multiple code files, and it will typically make use of some SDK libraries as well

The main method – Every Java application must have a main method, which defines where the program begins. In Java, the main method belongs to a class. Any class can have a main method. The main method looks like this:

Public static void main(String[] args)

{

// statements

}

Write a java program to perform matrix multiplication

**public** **class** MatrixMultiplicationExample{

**public** **static** **void** main(String args[]){

creating two matrices

**int** a[][]={{1,1,1},{2,2,2},{3,3,3}};

**int** b[][]={{1,1,1},{2,2,2},{3,3,3}};

//creating another matrix to store the multiplication of two matrices

**int** c[][]=**new** **int**[3][3];  //3 rows and 3 columns

//multiplying and printing multiplication of 2 matrices

**for**(**int** i=0;i<3;i++){

**for**(**int** j=0;j<3;j++){

c[i][j]=0;

**for**(**int** k=0;k<3;k++)

{

c[i][j]+=a[i][k]\*b[k][j];

}//end of k loop

System.out.print(c[i][j]+" ");  //printing matrix element

}//end of j loop

System.out.println();//new line

}

}}

o/p:- 6 6 6

12 12 12

18 18 18