

Language:

Language is a communication channel between two end users.

It is used to transfer the information from between two end users.

Many of the people use the languages like Hindi, Telugu, Tamil, English for communication.

If we want to interact with the machine, then we need one special language that is Programming Language.

```
---> By using above languages we cannot interact with machine
      so we need special language like PROGRAMMING LANGUAGE.

PROGRAMMING LANGUAGE:
  A language which is used to make communication between
  end user to machine is called PL.

  PL is a installable softwares.
  These are provides basic information(raw material).
  By using these basic information(PL) we can develop our
  own applications, technologies, tools, frameworks, Operatingsystem.
  ex: c, c++, c#, java
```

Programming Language:

A language, which is understands by the machine to process is called Programming language.

Aspects of Programming Languages:

- * Storing the data
- * Accessing the data
- * Processing the data
- * Delete the data
- * Update the data.

We have three types of programming languages.

- a. Binary language or Machine language.

b. Assembly language

c. High-level language.

Programming Language provides basic information or raw material to develop our own applications.

Programming Languages provides syntax (rules) or semantics (structure).

Binary -Language:

This is first and machine level language.

This is only understood by machine.

Machine can understand the data only in the form of zero and one's.

Each and every character first converted into ASCII / Unicode values later it is converted into zero and one.

Developing the program under binary language is very difficult and time consuming.

The main drawback of binary language is platform dependency.

Developing the application on binary language will take more time.

If application development time is increased automatically application development cost is also increased.

Platform:

It is the combination of software and hardware, it provides environment, to execute the programs.

Executing the programs means install software, update software's, typing something in notepad.

Platform Dependency:

If we are developing and compile the program under one operating system, the same program is not executing under different operating system is called platform dependency.

Assembly language:

This is low level language. It is used to avoid drawbacks of binary language. This language can avoid hard coding (complex), but unable to avoid platform dependency.

This language internally uses MNEMONICS (ADD, SUB, MUL, DIV) for developing the program.

Drawbacks:

Programmer always remember about MNEMONICS to develop the program.

To understand the program we need to write documentation

We need to take care about memories.

Debugging is very difficult.

High-Level Language:

These language are very user friendly language to easy to develop and easy to understand.

We have different languages under high level languages. Like C, C++, .Net, Java, COBOL, Pascal and etc.....

Java is the language to avoid the problems of platform dependency.

With the C, C++ we cannot develop internet support application and platform independency applications.

With the help of java we can develop internet support applications. The reason java is pure platform independent software.

Software:

It is a development tool, which is used to convert our imaginary things to real world existing things, by writing set of programs.

We have three types of software's.

System Software:

The software, which is used to develop hardware functionalities.

Ex: C, VC++, Embedded Systems, USB Driver Software's, Compilers, Translators, Printer Software's, Scanner Software's.

Application software:

All the back end or database software is comes under application software.

Ex: Oracle, MySql, Sql, DB2, SqlServer.....

Internet software:

To develop internet applications or internet support applications.

Ex: Java, .Net.

Java is coming in the form of three flavors.

- Java Standard Edition.
- Java Enterprise Edition.
- Java Mobile or Micro Edition.

Java software comes with two separate components.

- a. JDK
- b. JRE

Java Buzzwords or Java Features:

- Simple
- Platform Independency
- High Performance
- Security
- Architectural Neutral
- Portable
- Robust

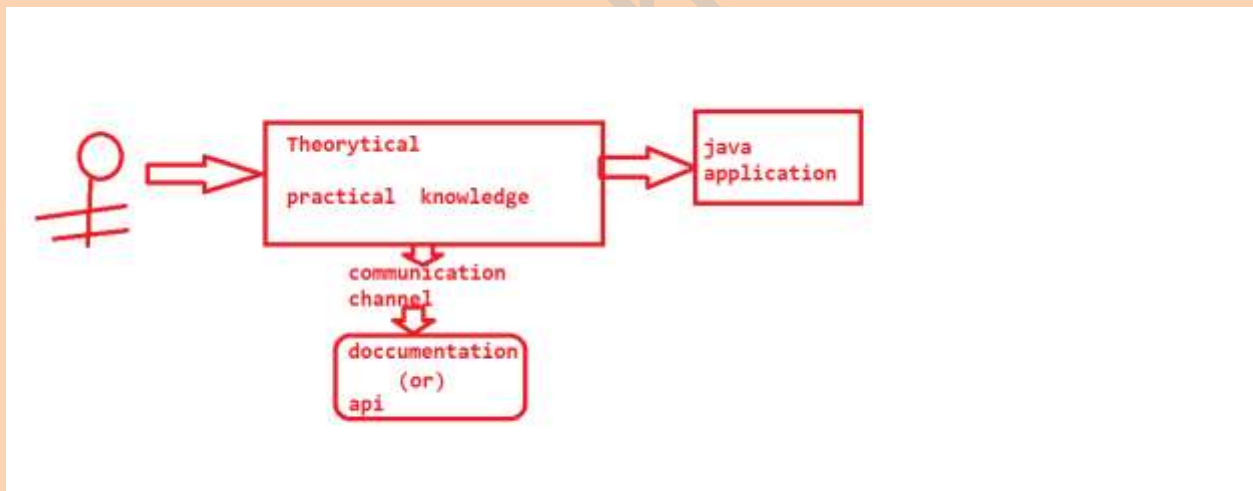
- Multithreading
- Distributed
- Dynamic
- Architectural Neutral
- Portable
- Oops

Simple:

Java is a simple language, due to the following reasons.

1. Garbage collection.
2. Provides huge predefined code. (Packages- rt.jar).
3. Documentation.
4. User Friendly Syntax.

Documentation:



Garbage Collection:

Java provides great services like Garbage Collector to clean unused memory or unreferenced memory.

This is also called as finalization.

If we go for other languages, programmer should be writes the code for both memory allocation and de-allocation.

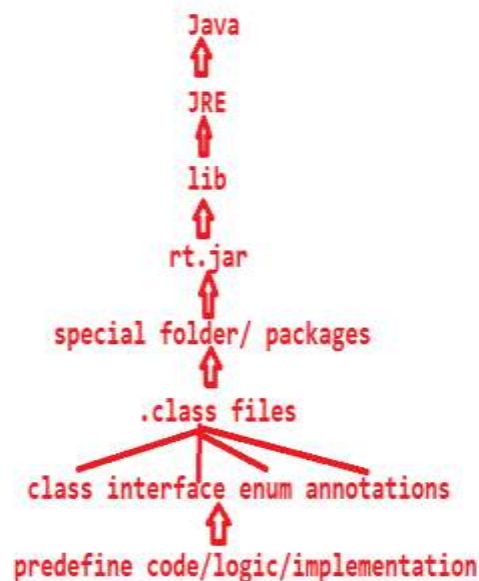
But in java, programmer is not writing any code for de-allocating the memory, JVM itself communicating with garbage collector to clean-up the memory by using garbage collector.

Java provides user friendly syntax means programmer can easily understand and remember more time, and easily placed those syntax's in developing of projects.

Java provides a huge predefine code or low level logic.

With the help of this code, we can develop our project within the less time and very easily.

All Java Standard Edition predefine logic is coming to our machine mean while of installation in the form archived file that is rt.jar

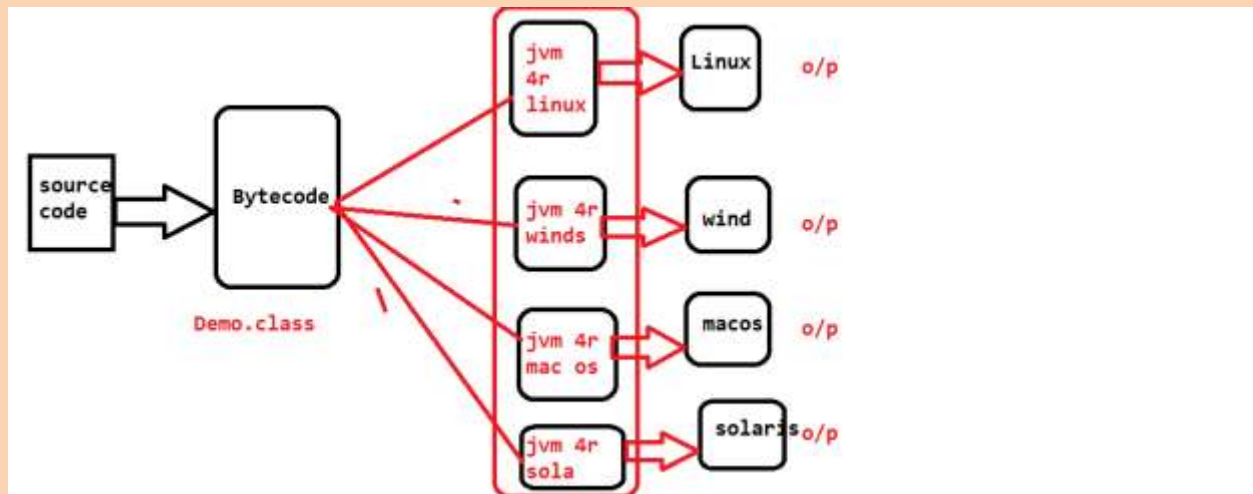


By Reading of java provide documentation, we can understand each and every java functionalities.

The syntaxes which are given by the java software, those are very user friendly.

Platform Independency:

Developing an application, compile the same application in one operating system, if we execute the same application in all operating systems is called platform independency.



Source-Code:

The code which is writing in any language (C, C++, .Net, Java) that code is called source code.

Machine code or Binary code: The code which is understands by the machine is called binary code or machine code.

Source code is not understands by the machine. So we need translate source code to machine code. To translate the source to machine code, java provides two translators.

1. Compiler
2. JVM

Translator: It is program which is used to convert from one type of code to another type of code.

Every java name must be ended with .java file, once we save the file, then that file available in hard disk.

Within the .java file we can write multiple classes.

If we want to interact with the compiler we need one predefined development tool that is "javac".

"javac" always followed by FileName.java

FileName.java must be contains source code.

Ex: javac FirstDemo.java

Whenever we written the above syntax in command prompt automatically, compiler will come in to the picture, it will interact with .java file and finally compiler will communicate with hard disk.

If the file not available compiler will give one error like file not found error.

If available ".java" file is loading from secondary memory to primary memory.

Mean while loading compiler will check is source code properly designed under java grammar rule or not, if not compiler will give Compile Time Error.

If code is properly designed then compiler will convert into byte code or JVM understandable code or intermediate code, and finally that code will be placed into ".class" files.

As many classes we have in one ".java" file, those many ".class" files will be created by the compiler.

Different class components are having different ".class" files and different byte code.

After creating the ".class" files, compiler will place all the ".class" into same directly (folder) in hard disk.

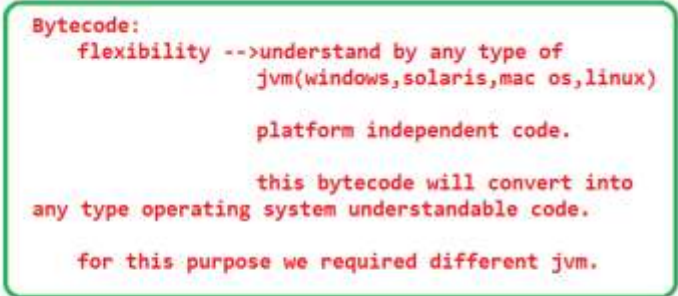
Bytecode:

The code which is generated by the compiler is called bytecode. It is not understand by the programmer and machine.

Bytecode is nothing but the intermediate representation of Java source code which is produced by the Java compiler by compiling that source code. This byte code is a machine independent code. It is not a completely a compiled code but it is an intermediate code somewhere in the middle which is later interpreted and executed by JVM. Bytecode is a machine code for JVM. But

the machine code is platform specific whereas bytecode is platform independent that is the main difference between them. It is stored in ".class" file which is created after compiling the source code. Most developers although have never seen byte code (nor have ever wanted to see it!) One way to view the byte code is to compile your class and then open the ".class" file in a hex editor and translate the bytecodes by referring to the virtual machine specification. A much easier way is to utilize the command-line utility javap. The Java SE DK from Sun includes the javap disassembler that will convert the byte codes into human-readable mnemonics.

Compilation: Converting from source code to bytecode.



Bytecode:
flexibility --> understand by any type of
jvm(windows,solaris,mac os,linux)
platform independent code.
this bytecode will convert into
any type operating system understandable code.
for this purpose we required different jvm.

The code which is available in ".class" file, that code representation is always in the form of bytes.

Whatever the code is generated by compiler, that is not understood by the machine. So again we need to convert from byte code to machine code, for that purpose we need one more translator that is JVM.

If we want to interact with JVM we need one development tool that is "java".

"java" command always followed by class name.

Ex: java Demo

Whenever we write above syntax in command prompt then JVM will come into the picture, and interact with ".class" file name, and finally loaded from secondary memory to primary memory. If byte code is properly organized

then we will get executable code that code is executing under all operating system. Then we can say java is a platform independency language.

But JVM is pure platform dependent component.

The reason is we have different JVM'S.

Whatever code is generated by compiler that code is understood by all JVM'S. Those JVM'S again convert and executed in their dependent operating systems.

Execution: Converting from byte code to executable code.

Slogan:

Write Once and Reuse Anywhere (WORA)

Write Once and Run Anywhere

Architectural Neutral:

Developing and compile the application under one processor, executing the same application under different processors is called Architectural neutral.

Portable:

The combination of Platform Independency and Architectural Neutral is called portable.

OOPS: (Object Oriented Programming System):

-->It is a methodology/technique

-->It will provide rules and guidelines, by using these rules we can developing languages or technologies

-->but it will not provide syntax

for example inheritance:

it will tell to all programming languages like access

data from one class to another class but not syntax:

in C# --> inheritance we can develop like bellow

A : B

in Java--> inheritance we can develop like bellow

A extends B

-->It is not a technology.

OOPS provides some rules and regulation and designs which help to develop application very easily.

We have the following OOPS principles. Those are

- 1) Class
- 2) Object
- 3) Encapsulation
- 4) Abstraction
- 5) Inheritance
- 6) Polymorphism

Q) Is java software independent or not?

A) Java software is platform dependent, the reason is we have different java softwares for different os.
One os support java software is not executing in other os.

Linux ARM 32

Linux x86

Linux x64

Mac OS X

Solaris SPARC 64-bit

Windows x64/

Q) Is java source code platform independent or not?

A) Whatever the program we writing under any operating system java syntax are never be changes so source code is PI.

Q) Is java bytecode platform independent or not?

A) java bytecode can be run on all os. so java bytecode is PI.

Q) Is java compiler platform dependent or not?

A) **NO.** Java compiler is platform independent, the reason is whatever the code we writing in the java that will common for all os, so all os java compiler architecture is same. so compiler is platform independent.

Q) Is java project/application PI or not?

A) Java project or applications are PI only, the reason is java application can be running under all os.

Q) Is jvm platform independent?

A) **No.** One jvm is not suitable for converting from bytecode to os understandable code we required different jvm. so jvm is not a platform independent. It is pure platform dependent component.

History of Java:

Development of java is start from Green Team (Java Team).

Initially java had introduced for digital devices (set-top boxes, televisions etc).

Now java is used internet programming, mobile devices, games, e-business).

James gosling, Patrick Naughton introduced in 1991 June.

Initially it was called greentalk and file extension is ".gt".

Later this language renamed as OAK.

OAK is symbol of strength.

OAK is the tree name.

It is the national tree for Germany, USA, France

In 1995 OAK renamed as Java.

No abbreviations for OAK and Java.

Java is an island of Indonesia.

This island is famous for coffee been.

In 1995 java introduced version like JDK Alpha and Beta.

In 1996, JDK 1.0 version released in the market.

JDK Alpha and Beta (1995)

JDK 1.0 (23rd Jan, 1996)

JDK 1.1 (19th Feb, 1997)

J2SE 1.2 (8th Dec, 1998)

J2SE 1.3 (8th May, 2000)

J2SE 1.4 (6th Feb, 2002)

J2SE 5.0 (30th Sep, 2004)

Java SE 6 (11th Dec, 2006)

Java SE 7 (28th July, 2011)

Java SE 8 (18th March, 2014)

Java Software released in the market in the form of three flavors/editions.

Java Standard Edition

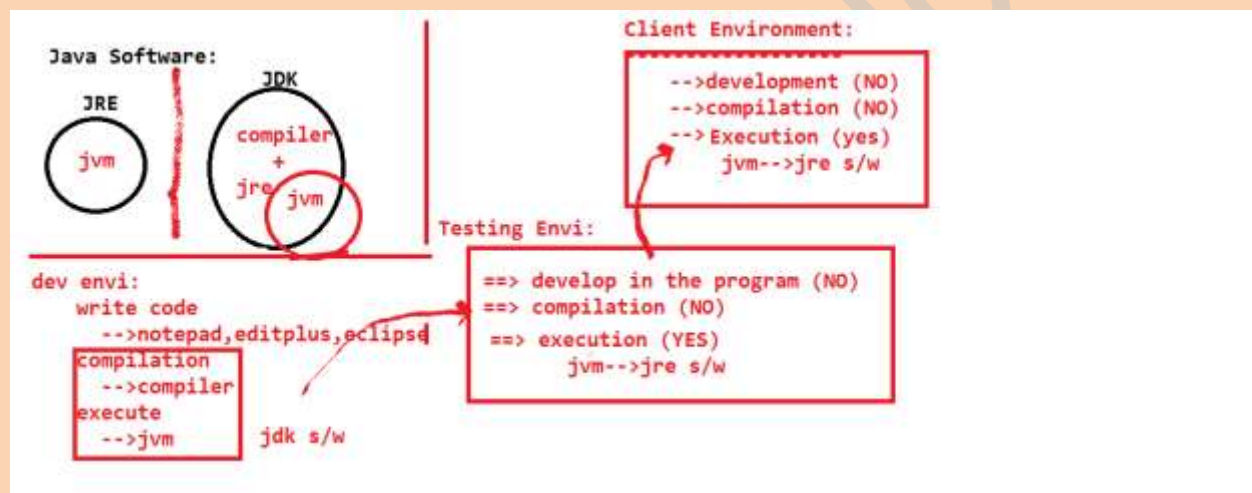
Java Enterprise Edition

Java Mobile/Micro Edition



Java software come into the market in the form of two flavours.

1. JDK
2. JRE



Why java is open source?

- > Download/install java soft ware.
free of cost for downloading from networks/inernet
- > Developing project on java.
Notepad, Editplus, Eclipse
(Text editors)
- > Developing project within the less and performance
(Frameworks--> Spring, Hibernate, Struts)
- > Running project on Server.
Apache Tomact, GlassFish,.....
- > To know internals of java software.
(source code)

JDK:

===

java standard edition development kit. It contains compiler and jre.it will provide:

- > Developments tools
- ===> source code
- > public jre
- > java db

Q) Why java is open source?

- * Java related software like JDK and JRE are freely available in the network.
- * Java software development code is also freely avialable in the market.
- * Java supported softwares (Servers(Apache tomcat), IDE's (Eclipse), Frameworks(Spring,Hibernate,Struts)) are also freely avialble in the market

With the help above flavors we can develop the following applications.

1. Standalone/Desktop applications.

2. Client-Server programming (socket programming)
3. Database interaction applications.
4. Web Applications.
5. Enterprise/Distributed applications.
6. Interoperable applications.

High Performance:

Application performance is depends upon the following things.

- 1) Less response time
- 2) Less memory utilization
- 3) More end users.

If we go for other languages, they are using only one translator for executing the program so those languages will take more time to execute. If any application will take more time to execute that application performance will be decreased.

To avoiding this problem java internally uses two translators

1. Interpreter.
2. JIT Compiler

Compiler:

It is a translator. It will convert our source code to byte code. It is check all line at a time, if ant syntax errors available first those errors will be placed into memory, after checking all the statements finally it will print all error information on the output device (console).

Interpreter will translate from byte code to machine understandable code. It will check line by line. If current line valid then control goes to next line otherwise it will print error or exception message on the console.

Interpreter is very good at execute the single time execute statements.

Whereas JIT compiler good at execute the looping statements.

But these two components don't have any capability to understand whether the statements are single time execute or looping statements.

In JVM, there is one special component i.e. "hotspot profiler".

This hotspot profiler will recognize the statement behavior.

If statement is single time execution statement then those statements will give to Interpreter, if statements are looping statements then those statements are handover to JIT compiler.

In java both Interpreter and JIT compiler works together and execute the program within the less time, if program is execute within the less time automatically the performance of an application will be increase.

By this reason, we can say java is a high performance language.

Robust:

Java is a robust language due to the following reasons.

- 1) Proper Memory Management.
- 2) Exception Handling.
- 3) Casting.

1) Proper Memory Management:

Java internally used one background program, which is used to maintain memory properly.

The background program is Garbage Collector. This Garbage Collector, will check is there any unused memory or not, if yes that memory will be cleanup, the same memory will be allocated to other data.

2) Exception Handling:

Java is giving very great support to handle the both compile time and runtime exceptions.

When ever exception is raised, internally java uses some predefine code to generate exception message in the following manner.

In the exception message it will give information about

- 1) File Name Information.
- 2) Class Name Information.
- 3) Method Name Information.
- 4) Line Number Information.
- 5) Pre/User Define Exception Class Information.
- 5) Description of an exception (Exception Message)
- 6) Package Information.

By using above information programmer can resolve error and exception within the less time with our facing complexity.

3) Casting:

In java both compilation time and runtime both compiler and JVM will check variable type and range and value.

By above three advantages, we can say java is a robust language.

Secure:

Java provides huge security in the following ways.

- 1) Avoiding the pointers.
- 2) Packages
- 3) Security Manager
- 4) Verifier.

→ If are avoiding the pointers we can provide security, avoid virus and hacking code.

→ With the help of packages we can provide security to both default and protected data.

Packages will provide fully security to default data, but packages are not provides full security to protected data.

If we are doing some modification on protected data, we can access from outside of the package also.

Those modifications are data make as static and other package class is the sub class of current package class.

→ Verifier will check whether the byte code is properly organized or not and is there any virus and hacking code or not.

If yes verifier is provides one error like `java.lang.VerifyError`.

Distributed:

Whatever applications developed under the java, those application services are equally distributes/sharable to all end users.

Data is shared between all the machines in the form of object within the same time.

Dynamic:

With the help of java we can develop dynamic pages and animated games and allocates the memory dynamically.

Static Applications:

The application which are always provides same response to all end user are called static applicaitons.

Dynamic Application:

The applications which always provide different response to different endusers at different execution time are dynamic applications.

Note: With the help of java we can develop both dynamic and static applications.

Memory allocation:

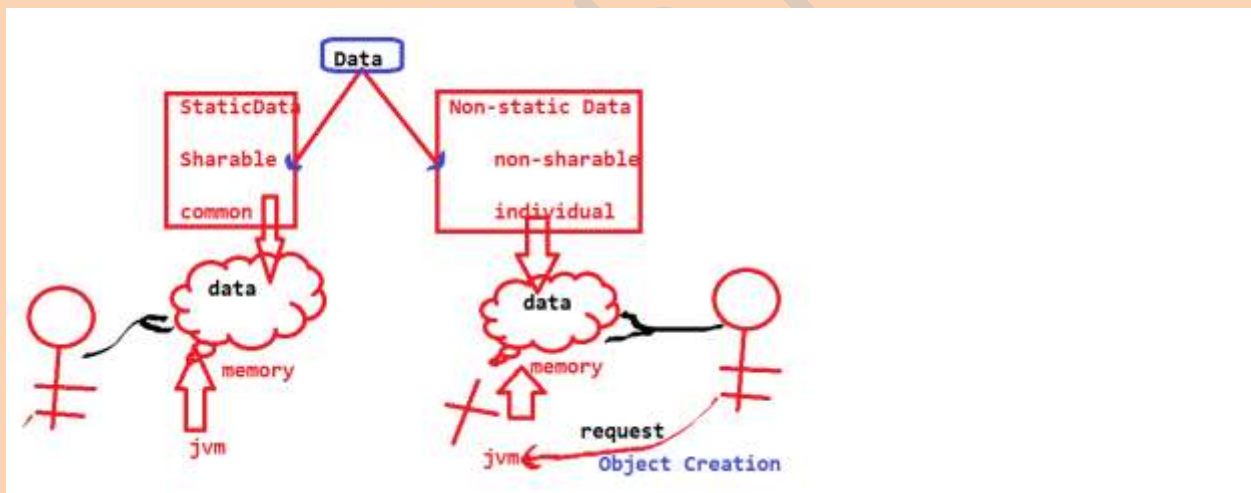
Java supports two types of memory allocations.

Static memory allocation:

By default jvm will provide the memory for all static data.

Dynamic memory allocation:

When we require memory for non-static data, then only we can make request to jvm through object creation for allocating memory for non-static data.



Multithreading:

Executing the multiple threads concurrently is called multithreading.

With the help of multithreading we can execute our programs with the in the less time, then automatically we will improve the performance of an application.

This concept will provide proper co-ordination between gaming elements.

In general our java program contains two types of threads

- 1) Non-Daemon (MainThread).
- 2) Daemon (Garbage collector).

With the help main thread we can call methods, we can create memory.

With the help of garbage collector we can clean-up the memory.

Object Oriented Programming System:

Any language which has been developed based on object oriented programming system principles those language are called OOPL.

Ex: C++, .Net, java.

We have the following OOPS.

- 1) Class.
- 2) Object.
- 3) Encapsulation.
- 4) Abstraction.
- 5) Inheritance.
- 6) Polymorphism.

Major and Minor versions:

Java software comes in the market in the form of two versions.

- 1) Major Version.
- 2) Minor version.

Major Version: If we want to add new features to the existed software, then we should go for major version.

Ex: jdk 1.6.0

jdk 1.7.0

Minor version:

If we want add bug fixing code to the existed software then we can release our software as an minor version.

Ex: jdk 1.6.0_20

One java software is not suitable for all the operating systems. Different operating systems have their own java software.

Environment Variables Setting:

Command prompt is comes with operating system, whereas java development tools (binary files) and library files are comes with java software.

So our command prompt is unable to recognize the development tools.

Then programmer should provide the information about binary and library files to command prompt, with the help of environment variables.

OS by default uses environment variables to recognize software.

For java we have two types of path settings.

- 1) Temporary settings.
- 2) Permanent settings.

Temporary Settings:

Whatever the setting which we are done at one command prompt those settings are applicable to that command prompt only not applicable to other

command prompt and also if we did close the command prompt automatically all the settings are gone.

Syntax: set path=C:\Program Files\Java\jdk\bin;

With the help of above settings our command prompt recognize the binary files. (javac, java, javap command).

Syntax": set classpath=.;C:\Program Files\java\jre\lib\rt.jar;

With the help of above settings our compiler and jvm can recognize the predefine .class files.

Binary files used to compile and execute the program, where as Library files are used develop the user define program.

Permanent Settings:

The settings which are applicable for entire system, those settings are called permanent settings.

Steps to permanent settings:

Right click on my computer → Properties → Advanced System Settings
→ Advance → Environment Variables → under System variables.

Click on new button set the path.

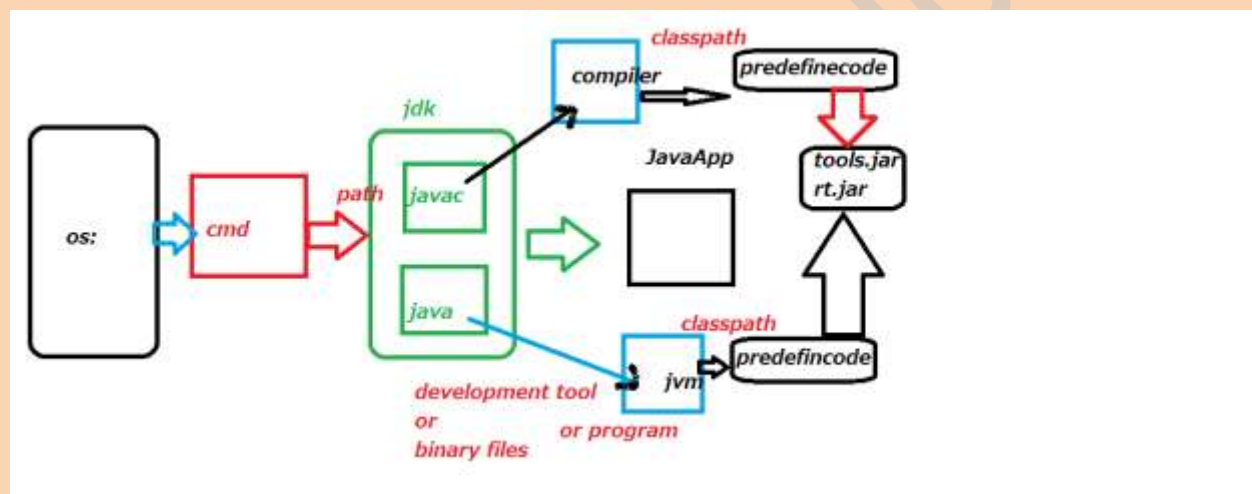
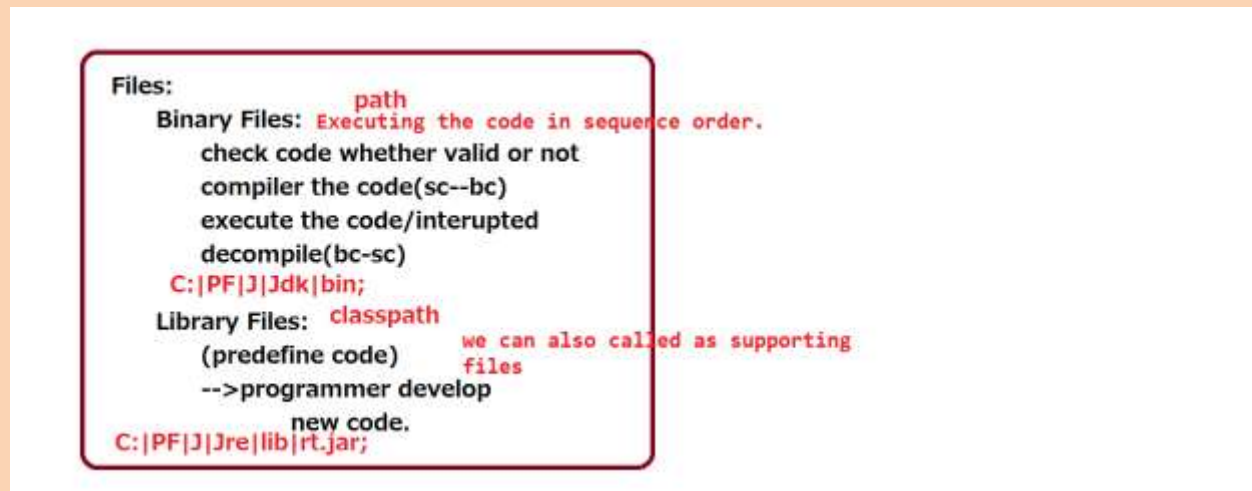
Variable Name: path

Variable Value: C:\Program Files\Java\jdk1.6.0\bin;

Click on new button set the classpath.

Variable Name: classpath

Variable Value: .;C:\Program Files\Java\jre6\lib\rt.jar;



Object:

It is a real world existing thing.

It is provides memory for non-sharable data or non-static data or instance data.

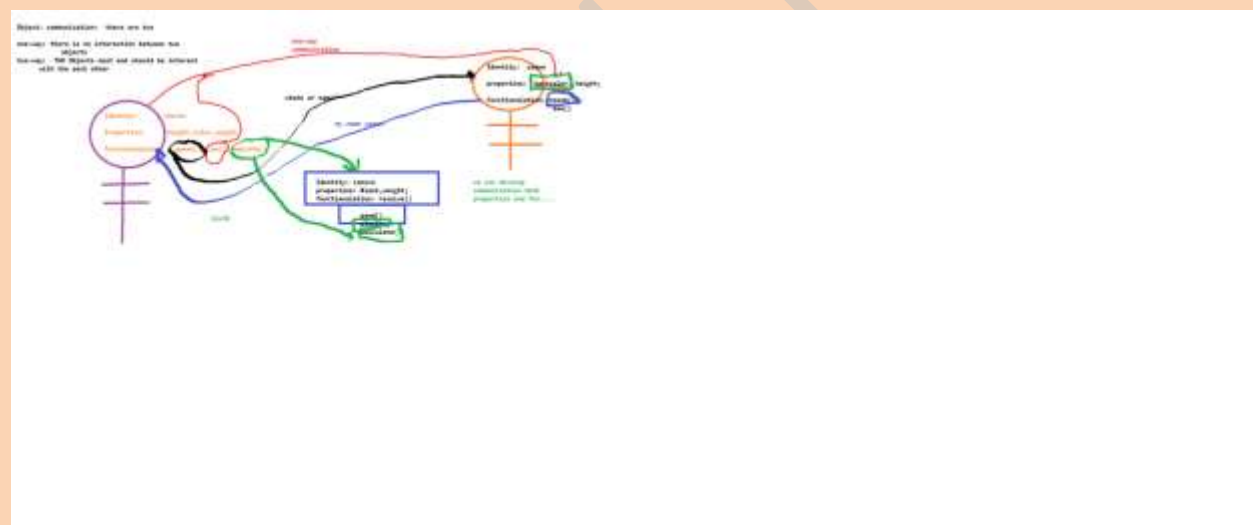
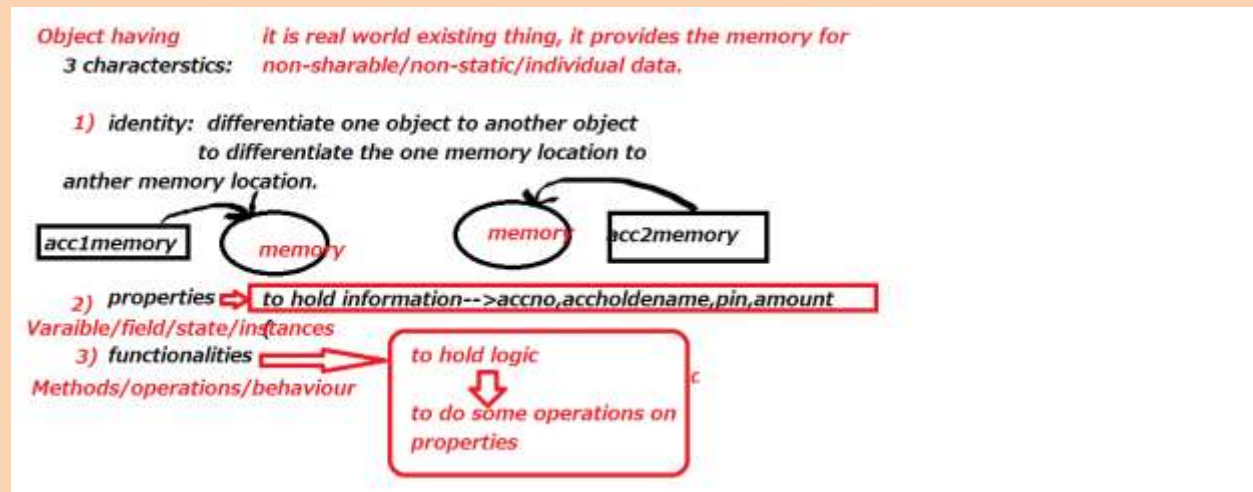
Object is having following characteristics.

Identity: To differentiate the object to another
 (Differentiate one memory to another memory).

Properties: To hold the information (variables, Fields, Instances, Member Variables, State).

Functionalities: to hold the logic (to do some operations on properties). (Methods, Operations, Behavior, Functions).

One object is communicating with another object with the help of functionalities.



real time example for class and object:



properties:
cardNumber
pin
cardType

functionalities:
withdrawl
shopping
mobilerecharge

Class:

It is a imaginary thing, it contains to hold object information (identity + functionalities + properties).

Class is a combination of

Variables + Methods or

State + Behavior or

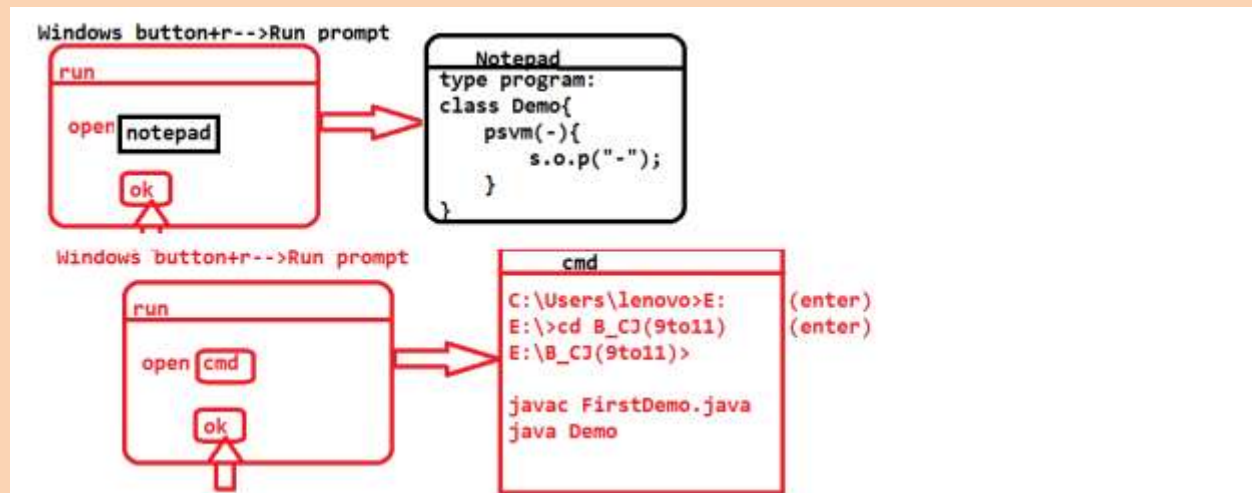
Member Variables + Member Methods

It is a model.

It is a design.

It is a blueprint.

How to open notepad and command prompt and compile and execution:



First java program:

```
class Demo{
    public static void main(java.lang.String[] ram){
        java.lang.System.out.println("hi friends");
    }
}
```

Q) Why main is public?

JVM is one program. It is located in one package. Our program/class available in one more package. So one package data want to communicate with another package data we should use public keyword.

Q) Why main method is static?

Without providing memory by programmer with the help of JVM, for main method, JVM itself want to provide the memory for main and calling purpose, the method (main) must be static.

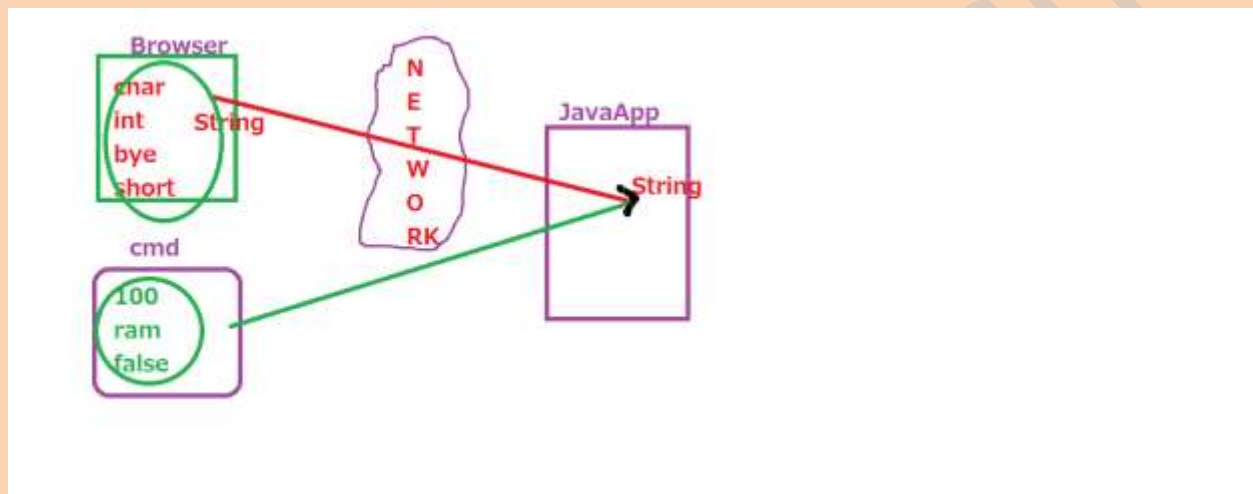
That means without creation object if we want to call the main method by using class name by the JVM, we mention the main is static.

Q) Why main method is having void?

Main method is not giving any value to JVM.

Q) Why main method is having String [] parameter?

The data which is coming from browser or command prompt that data is always string format. To holding that data we need String [] variable.



File Name: FirstDemo.java

Class Name: Demo

Compilation: javac FirstDemo.java

Execution: java Demo

If we are doing any modification to source file programmer must be save file and compile the program again.

Valid syntax:

```
String [] ram;
```

```
String [] ram;
```

```
String []ram;
```

```
String ram[];
```

Invalid syntax:

```
[] String ram;
```

Note: we cannot place array symbol before data type.

```
class Demo{  
    static public void main(java.lang.String[] ram){  
        java.lang.System.out.println("hi friends");  
    }  
}
```

Note: We change places between static and public.

```
class Demo{  
}
```

→We can develop empty .java file. This file can be compiled but we cannot get any ".class" file, the reason is there is no source.

We cannot execute the program. The reason is there is no class name.

→We can create empty java class. We can compile but we cannot execute.

In java 1.6 we got one error: NoSuchMechError:main

```
class {  
}
```

We cannot develop class without class_name.

If we write the following code we will get one compile time error that is identifier expected.

```
class Demo{
```

```
public static void main(String [] args){  
    System.out.println ("java program")  
}  
}
```

Every statement must be ended with semicolon, otherwise we get compile time error that is ';' expected.

```
class Demo{  
    public static main(String [] args){  
        System.out.println ("java program");  
    }  
}
```

In java method must be have return type. If we are not writing, we will get compile time error that is invalid method declaration, return type required.

```
class Demo{  
    static{  
        System.out.println("this is static block");  
        System.exit(0);  
    }  
}
```

Without main method also we can compile and execute the program from java 1.0 to java 1.6. But in java 1.7 and 1.8 main method is mandatory.

System.exit(0) means, we are explicitly stopping the program.

```
class Demo{  
    static public void main(String... ram){
```

```

        System.out.println("this is main method");
    }
}

```

→(...) called var args (Variable arguments). We can also call it ellipse.

Internally var args maintain array concept only.

This feature is introduced in java 1.5.

Valid Syntax:

```

String...ram;
String... ram;
String ...ram;
String ... ram;

```

Invalid Syntax:

```

String ram...;
...String ram;
String. ..ram;
String.. .ram;
String . . . ram;

```

```

class Demo{
    static public void main(String... ram){
        System.out.println("this is main method");
    }
}

class A{

```

```
}
```

```
class B{
```

```
}
```

```
class C{
```

```
}
```

Within the one “.java” file we can create multiple classes. As many class keywords we have those many “.class” file will be generated by the compiler.

In the execution time, we should give class name, which have main method.

```
class Demo{
```

```
    static public void main(String... ram){
```

```
        System.out.println("this is Demo main method");
```

```
    }
```

```
}
```

```
class A{
```

```
    static public void main(String... ram){
```

```
        System.out.println("this is A main method");
```

```
    }
```

```
}
```

```
class B{
```

```
    static public void main(String... ram){
```

```
        System.out.println("this is B main method");
```

```
    }
```

```
}
```


Within the one ".java" file we can write multiple classes and also in all the classes we can write main method also.

Which class name we are going to be give at execution time that class main method will be execute.

```
javac FirstDemo.java
```

```
java Demo
```

```
java A
```

```
java B
```

We can compile more than one .java file at a time with the help of following syntax.

```
javac *.java
```

But we cannot execute more than one ".class" files at a time directly.

If given .java file is not existed, then compiler will give file not found error.

if given .class file is not existed, then JVM will provide
NoClassDefFoundError: class name

File Name no need of same as class Name. But if the class is public type then our filename must be same as class Name otherwise compiler will give one compile time error i.e.

```
class <class-name> is public, we should declared file name as <class-  
name>.java
```

```
public class Demo{  
  
}
```

In the above scenario we should save filename as Demo.java

We cannot write more than one public class within the one ".java" file as bellow.

```
public class Demo{
```

```
}  
  
public class C{  
  
}  
  
*****
```

No user define/predefine method will be executed automatically in java, except main method.

If we want to execute the user define/predefine method then we should call explicitly.

```
class B{  
    static void m1(){  
        System.out.println("m1 method");  
    }  
    static public void main(String... ram){  
        System.out.println("this is B main method");  
        //m1();  
    }  
}
```

We can call main method explicitly by using below syntax.

```
main(new String[0]);
```

If we call main method of other class then we should use below syntax

```
Class_name.main(new String[0]);
```

```
class A{  
    psv main(String... ram){
```

```

        System.out.println("hi");

        main(new String[0]);

    }

}

```

We can't call main method within the main method. This syntax will reach infinity loop.

Finally JVM will provide one runtime error is "java.lang.StackOverfloeError".

At a time we can load only one .class file by using development tool like java, javaw, javaws.

If we want load other ".class" file we should use the following methodologies.

1. By using development tools (java A)(javac,javap,javaw)
2. By using class_name and calling static variables or static method(A.main(-)).
3. By using object creation (new A())
4. By using Reflection API (Class.forName(-))
5. By Inheritance
6. ClassName.class
7. De-serialization Procedure.

Compiler will check class information first in method later in class later in ".java" file later in package later in current working directly later finally control goes to predefine class.

Without main method also we can execute the program upto 1.6 not from 1.7 onwards.

Upto 1.6 jvm without checking main method exists or not first control give to static blocok later main method

But from 1.7 onwards jvm first checks whether main method exists or not. If exists, jvm will control to static block later main method

If not exists we will get runtime error.

```
public class Demo{  
    static{  
        System.out.println("sb2");  
        System.exit(0);  
    }  
}
```

Java Programming Elements:

The following are the java programming language elements (The syntax which is used to develop the program)

Those are

1) Comments.

2) Packages

3) Java Tokens

a.Keywords.

b.Literals.

c.Operators.

d.Separators.

e. escape characters

f.identifiers

i.Class

1.Methods

2.Variables

ii.Interface

1.Methods

2.Variables

iii.Enum

1.Methods

2.Variables

4) Annotations

5) Constructors

6) Blocks

7) Import statements

Identifiers:

It is name or word or character which is used differentiates from one java element to another java elements.

```
class Student{  
    public static void main(String... ram){  
        System.out.println("hi");  
    }  
}
```

To differentiate from one java element to another java element, then we can go for identifiers'.

Identifier is a name, character, word which is differentiate one java element to another java element.

Identifiers:

```
String sname="yaane"
double amount=123.23;
```

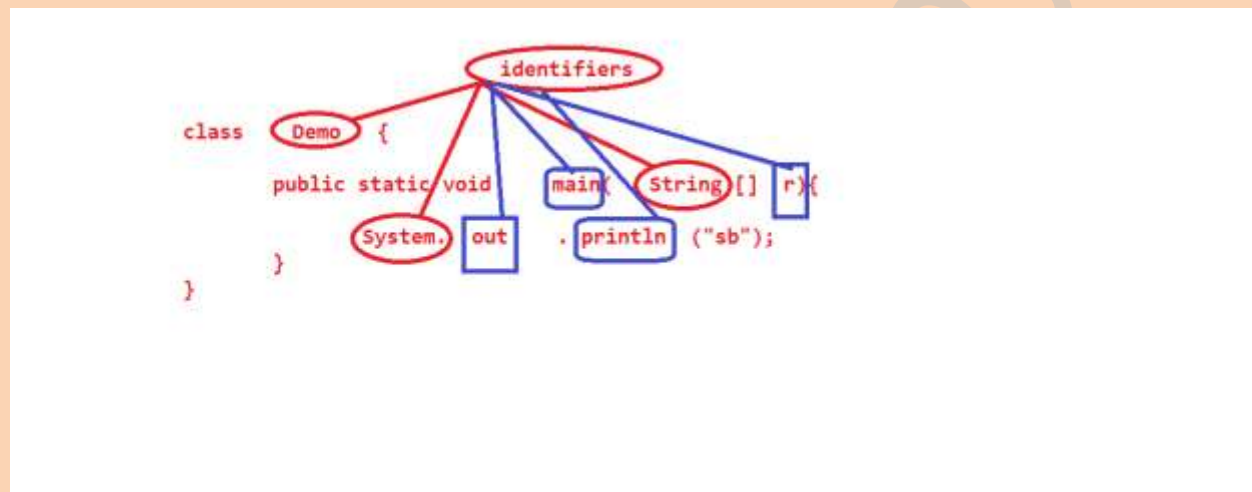
is name or word or character which is used differentiate one value to another or one logic to another.

```
class Employee{
    logic1--->employee
}

class Student{
    logic2--->Student
}

class Bank
{
    int withdrawl(){
        //withdrawl logic
    }

    void deposit(){
    }
}
```



Naming Conventions in Java:

1) Class:

Class name must be starts with capital letter

Ex: Student

If any class name having multiple words first word first letter must be capital letter, the remaining words first letter must be capital letter.

Ex: StudenPersonalDetails

We can use digits in the class name

Ex: Student1;

But don't start class with digits.

We can use two special characters with in the class name.

ex: Student_Info

Ex: Student\$Details

We can start class name with the help of '_' and '\$' and combination also.

Ex: __ Student

\$_\$Bank

We can use predefine class name as user define class name. But in this time we cannot get functionalities from predefine class.

```
Ex: class String{  
    }
```

Note: Don't use predefine class names as user define class names.

Don't use keywords as a class names.

Invalid Class Names:

Class name never be start with digits.

#ex: 1Student //wrong

Class name doest have any spaces.

Ex: Student Details//wrong

Student Personal Details//wrong

Class name doesn't have any special characters except under(_) and dollar(\$).

Ex: Student"Info //wrong

Ex: Student=Info //wrong

We cannot use keyword for class names.

```
Eex: class while{ //wrong  
    }
```

Whatever the rules we have in java about to classes, the same rules applicable to interfaces, enum, annotations.

Keywords: Keywords must be write within the small case.

Ex: for, if, else, private, public

Ex: For, Native, Strictfp //wrong syntax

Variables:

Variables are always starts with small letter.

Ex: String name;

byte age;

float sal;

Variables name having more than one word then first word first letter must be small letter, the remaining words first letter must be capital letter. The remaining rules same as class rules.

Ex: String studentName;

String accountHolderName;

Methods:

Whatever the rules we have related to variables, the same rules we have for methods also.

There is a small different between variables and methods is method always ended with '(' and ')'.
(Note: The original image contains a typo 'method' instead of 'methods' in this sentence.)

Constants:

All the final variables are comes under constants in java

Ex: final double PI= 3.14;

final variables or constant variable names are always in the capital letters.

Ex: MIN_PRIORITY→1

MAX_PRIORITY→10

NORM_PRIORITY→5

Packages:

Package names are must be in the small case.

Ex: java.lang

java.util

java.io

java.net

com.ram

com.nit

com.google

Literals: A values which is assigns to variable is called literal

Ex: int a = 10; //10 is an int literal

char c = 'b'; //b is an char literal

boolean b = false; //false and true are boolean literals

boolean b1 = true; //true--boolean literal

String s = "ram"; //ram-- String literal

Student s = null; // null---is also called as literal

double d = 23.34d; //23.34---is an double literal

char literal is always within the single quote.

Literals: The value which is assign to variable.

1)null
2>false
3>true

→

predefine literals

5 types of literals
=====

1. Integral Literal:
byte b = 100;
short b1 = 200;
int b2 = 400;
long b3 = 40000;

2. Floating Literals
float f1 = 23.34f;
float f2 = 23.34;

3. Boolean literal
boolean b1 = false;

4. Char literal:
char c = 'a';

5. String literal
String s = "yaane";

We cannot write more than one character with in the single quote.

Ex: 'a' or '4'

'abc' //wrong

boolean literal → it is always either true or false.

String literal → always within the double quotes.

float literal always ended with either small 'f' or 'F'

Ex: float f1 = 23.23f;

float f2 = 56.61F;

ended 'f' or 'F' is mandatory.

double d = 34.45;

double d1 = 34.45d;

double d2 = 34.45D;

double literal can be followed by either 'd' or 'D'. It is not a mandatory.

By default all number in java comes under int literal.

By default all fractional numbers in java comes under double literal.

```
long l = 100;
```

```
long l1= 100L;
```

```
long l2 = 100l;
```

Ended 'l' and 'L' is not mandatory.

Keywords:

Keywords are predefined words or reserved words.

These are having some functionality through logic.

Keywords are used to decrease the size of a program.

With help of keywords we can develop the program within the less time.

These functionalities will help us to communicate with both compiler and JVM.

Whenever our program executes internally keywords functionalities also executed.

We have 50 keywords in java among them 48 are used and 2 are unused

false, true, null these are also predefine words but these not comes under keywords, these are comes under literals.

```
boolean b = false;
```

```
boolean b1 = true;
```

```
String s = null;
```

List of keywords:

Creating java files:

class

interface

enum

data types:

byte

short

int

long

float

double

char

boolean

For return type:

void.

Controle statements:

if

else

switch

case

default

Looping statements:

for

do

while

Transfer the controle:

break

continue

return

AccessModifier:

private

public

protected

Modifier:

final

abstract

synchronized

strictfp

volatile

native

transient

object related keyword:

this

super

instanceof

relationship:

extends

implements

package:

package

import

Exception handling:

try

catch

finally

throw

throws

assert

memory allocation:

static

new

unused:

goto

const

Note: keywords must be writes in small case.

String is a predefine class, which is available in java.lang package. But we can use String as a data type.

50 (48--used)	Keywords =====	2--unused (goto,const)
1.class	13.static	23.break
2.interface	14.new	24.continue
3.enum	15.if	25.return
4.byte	16.else	26.this
5.short	17.for	27.super
6.int	18.while	28 instanceof
7.long	19.do	29.private
8.float	20.switch	30.protected
9.double	21.case	31.public
10.char	22.default	32.extends
11.boolean		33.implements
12.void		34.package
		35.import
		36.try
		37.catch
		38.finally
		39.throw
		40.throws
		41.assert
		42.final
		43.abstract
		44.synchronized
		45.transient
		46.strictfp
		47.native
		48.volatile

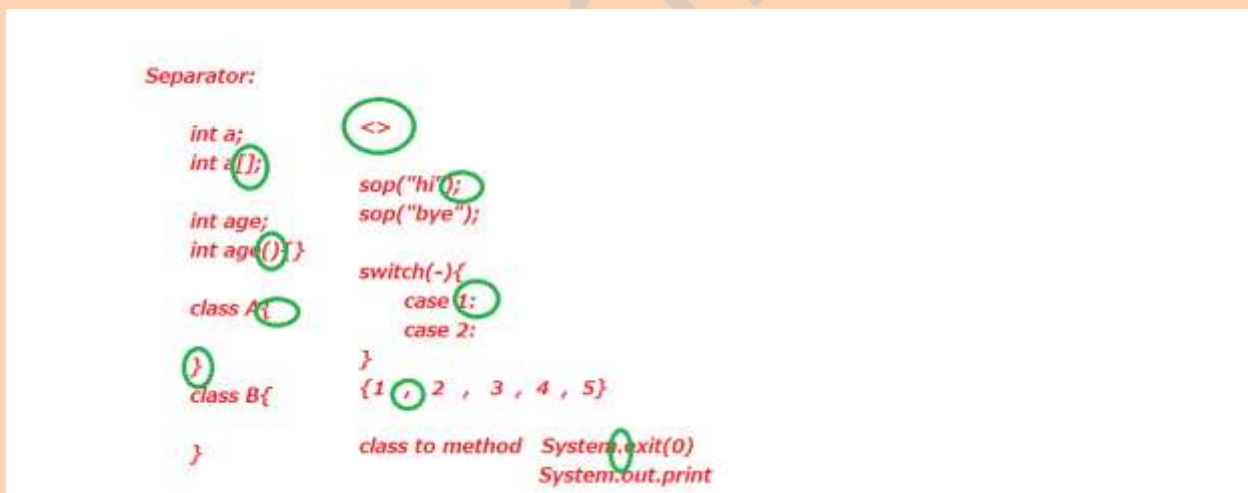
1.

Separators:

It is a small java element which is used to differentiate from one java code to another code (element).

These are some of the special characters.

Ex: We have totally eight separators.



Those are:

"()" → To differentiate variable to method

"{}" → Differentiate from class to method or method to method (scope).

"[]" → Differentiate primitive variable to reference variable.

"." → Make a relation between class to method and class to variable

"," → Use to separate one value to another value

"," → To separate one statement to another

":" → To separate one case to another in switch

"<>" → To separate normal collection object to generic

Escape Characters:

These are used print the data in different locations/format. (Style).

```
class EscapeCharacters{  
    public static void main(String...ram){  
        //System.out.println("hi\tbye");  
        //System.out.println("hi\nbye");  
        //System.out.println("hi\bbye");  
        //System.out.println("hi\rbye");  
        System.out.println("hi\'bye");  
        System.out.println("hi\"bye");  
        System.out.println("hi\\fbye");  
    }  
}
```

Data types:

Data types are pre-reserved words, which is used to specify the type of literal/value.

```
int a = 10;
```

```
boolean b = false;
```


With help of data type we can provide the memory for variables.

```
int a = 10; // 4 bytes of memory.
```

Data types are allows valid operations.

```
boolean b = true;
```

```
b = b++; //invalid syntax
```

Data types are provides proper result/output.

```
String s1 = "10";
```

```
String s2 = "20";
```

```
S.o.p(s1+s2);//1020
```

```
int a = 10;
```

```
int b = 20;
```

```
S.o.p(a+b);//30
```

Data types are used to differentiate the one value to another value.

Data types are provides proper memory management.

Classification of data types:

Diagram relation to classification.

Q) Why java uses these many data types?

A) To use the memory in proper manner then java uses different data types.

If the value is small then we can go for small range of data type.

If the value is big then we can go for higher range of data type.

Ranges of data types:

byte: (1 byte)---> $1 * 8 = 8$ bits

byte b = 10;

If we want to place the data in the memory, first we should convert into binary number system. (0,1). Zero and one are two digits.

$$\begin{array}{rcl} 8 & & \\ 2^8 & = & 256 \\ & \text{---} & \\ & & 2 \end{array} \quad \begin{array}{rcl} 7 & & \\ & = & 128 \\ & = & 2 \end{array}$$

-ve number, + numbers

-128 to 128

-128 to 0 to 127

$$\begin{array}{rcl} 7 & & 7 \\ -2^7 & \text{to } 0 & \text{to } 2^7 - 1 \end{array}$$

$$\begin{array}{rcl} 8-1 & & 8-1 \\ -2^{8-1} & \text{to } 0 & \text{to } 2^{8-1} - 1 \end{array}$$

The range of float, double is greater than the byte,short,int,long datatypes.

Casting:

Converting from one data type variable value to another data type variable value is called Casting.

Casting is two types.

1. Implicit casting.
2. Explicit casting.

1. Implicit Casting: Converting lower data type value to higher data type values is called implicit casting.

We can pass the values to variables within the range only we cannot pass beyond the range values.

```
int a = 10;
```

- 1) In the above statement compiler will check the destination variable type (int).
- 2) Based on the type, compiler will check range
- 3) Compiler is also checking the value of a variable(source value).
- 4) If value is within the range then compiler will not give any error otherwise compiler will give one compiler time error. That is possible lossy conversion.

Sometimes it will give incompatible error.

```
Char a1 = 'd'
```

In the above statement compiler convert 'd' into Unicode later checks that Unicode is within the range or not.

```
class Casting{  
    public static void main(String args[]){  
        System.out.println("main method");  
        byte b = 127;
```

```
        System.out.println("b: "+b);  
        byte b2 = -128;  
        System.out.println("b2: "+b2);  
        //byte b1 = 128;  
        //System.out.println("b1: "+b1);  
        short s = 32767;  
        System.out.println("s: "+s);  
        short s1 = -32769;  
        System.out.println("s1: "+s1);  
    }  
}
```

```
class Casting{  
    public static void main(String... ram){  
        char b8 = 'A';  
        int b9 = b8;  
        System.out.println(b9);  
        byte b1 = 10;  
        short b2 = b1;  
        int b3 = b1;  
        System.out.println(b3);  
        float b4 = b1;  
        System.out.println(b4);  
        long b5 = 123L;
```

```

        float b6 = b5;
        System.out.println(b6);
float b7 = 34.34F;//float must be ended //with either 'f' or 'F'
        char b10 = 97;
        System.out.println(b10);
        for(int i=48;i<58;i++){
            System.out.print((char)i+" ");
        }
        System.out.println();
        for(int i=65;i<=90;i++){
            System.out.print((char)i+" ");
        }
        System.out.println();
        for(int i=97;i<=122;i++){
            System.out.print((char)i+" ");
        }System.out.println();
        char c = '?';
        System.out.println((int)c);
    }
}

```

Whenever we use any non-fractional number without mention datatype that is always treated as int type value and if we mention fraction number without mention datatype that is always treated double type value.

Whenever we use any non-fractional number with datatype, here there is one operation has been executing.

a.If the value is within the range, then that value will treat as that datatype value only otherwise treated as int type value.

Whenever we use any fractional number with datatype, here there is one operation has been executing.

a.If the value is within the range, then that value will treat as that datatype value only otherwise treated as double type value.

Assigning the values to variables:

We have 7 ways to assign the values to variables.

1. Assigns the values directly to variable.

```
int a = 111;
```

2. Assign the values to variable by using another variables.

```
byte b = 222
```

```
int c = b;
```

3. Assign the values to variables by using method return type.

```
int m1 = m2();//calling area
```

Destination part initialization part

```
int m2(){//called area
```

```
return 333;
```

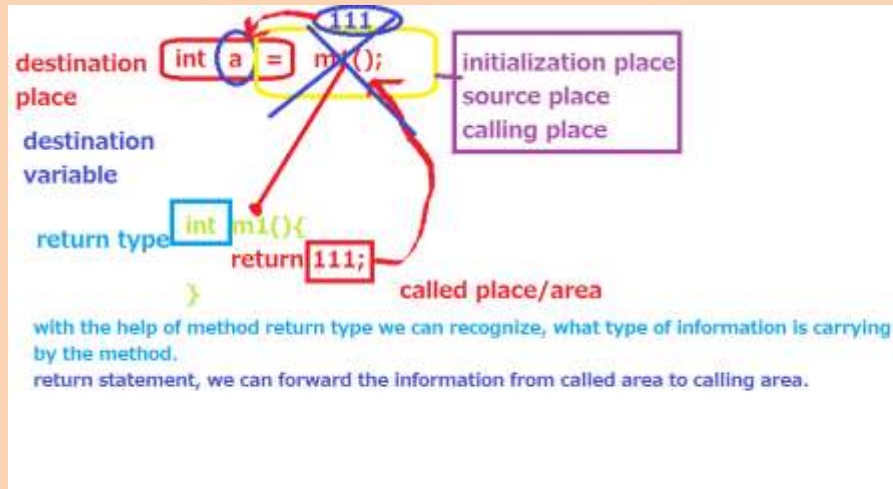
```
}
```

Note: "return" statement will forward the data from called area to calling area.

If any method calling is available in initialization part, then that method must be non-void method.

non-void method means, method carrying some information. That information must be compatible with destination variable datatype.

Method must have return type. Return type also must be compatible with destination variable datatype. If any method having return type then that method must have return statement with value. Again value must be compatible with method return type.



Internal functionalities of compiler meanwhile of assigning the value to variable with the help of method return type.

```
byte b = m1();
1. compiler check destination variable type(byte)
2. compiler check destination variable type range(-128 to 0 127)
3. compiler check source value( m1())
4. compiler control goes to method(m1()) and check method return
   type and return value compatible or not
   if not compiler error
   if yes
       compiler checks method return type and destination
       variable type compatible or not
       if not compiler time error
       if yes no error.
```

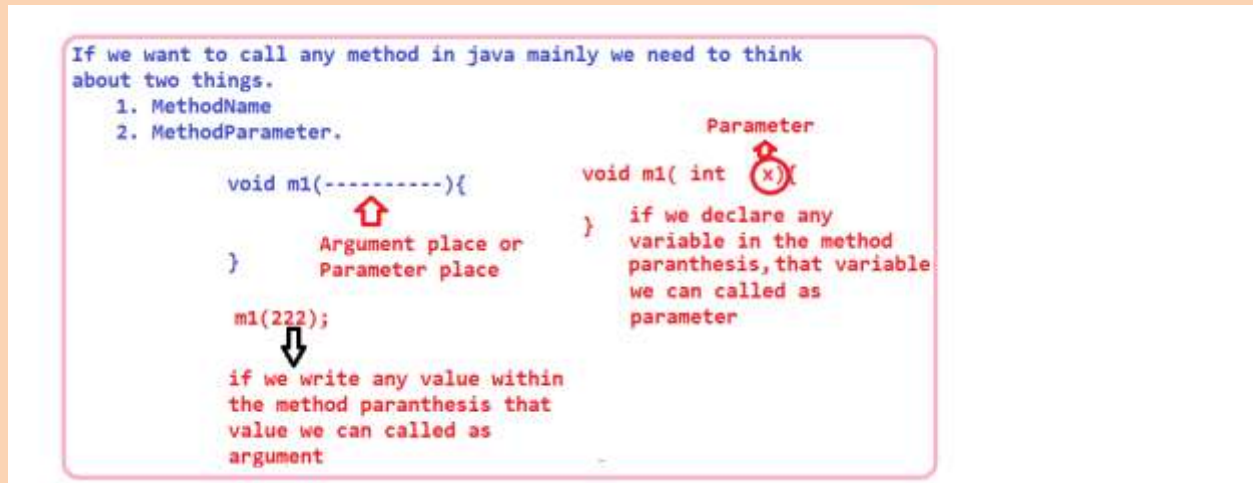
4. Passing the value as a argument to other method.

```
void m3(String s){
}
}
```

calling the m3 method: m3("ram");

```
void m2(boolean b){  
  
}
```

calling the m2 method: m2(true);



5) We can assign the value to variable with the help of expression.

int c = 10+20;//here 10+20 is an expression

```
int a = 10;
```

```
int b = 20;
```

```
int d = a+b;//a+b is an expression.
```

6) By using cast operator.

```
byte b = (byte)130;
```

7) By using object.

```
int I = new Integer(100);
```

Functionalities of compiler and jvm in the casting:

compiler:

compiler checks type of an variables(both source and destination variables)

Based on the type, compiler checks range of an variables.(dattypes).

If Destination variable Range greater than source variables Range then compiler will satisfy otherwise compiler will give error.

(DVR>=SVR)

Compiler not check any value.

```
short b = 111;
```

```
int i = b;//valid
```

```
DVR >= SVR
```

```
byte b1 = b;//not valid
```

Jvm: jvm will check type of an variable, range of an variable, DVR >= SVR or not, finally jvm will check "value" also.

Note: if we assign the value directly to variable compiler will check type and range and value.

```
byte b = 127;
```

ExplacitCasting: Converting from higher datatype value to lower datatype value.

By default it is not possible in java.

but we can do with the help of cast operator.

```
int b = 127;
```

```
byte b1 = b;//invalid
```

```
byte b2 = (byte)b;//valid
```

In the explicit casting, if the source value is within the range, then jvm will place the same value assigned to destination variable. Otherwise jvm will place some other value.

```
public class ExplicitCasting {
```

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

```
        byte b = 127;
```

```
        int i = b;
```

```
        int j = 127;
```

```
        byte b1 = (byte)j;
```

```
        System.out.println(b1);
```

```
        int k = 130;
```

```
        byte b2 = (byte)k;
```

```
        System.out.println(b2);
```

```
        int k1 = 150;
```

```
        byte b3 = (byte)k1;
```

```
        System.out.println(b3);
```

```
        int k2 = 300;
```

```

        byte b4 = (byte)k2;

        System.out.println(b4);
    }

}

```

Note: we can not convert boolean data type values to another data type.

```

boolean b = false;

int a = b;//invalid

```

boolean value we cannot place into other datatype and any other datatype value cannot be placed into boolean datatype.

```

byte b = 100;

boolean b1 = b;//invalid

```

```

public class ExplicitCasting {

    public static void main(String[] args) {

        char a = 'a';
        System.out.println();
        System.out.println("====");
        System.out.println(a);//s.o.p('a');
        char a1 = 'z';
        System.out.println(a1);
        //System.out.println(ram); //invalid
        //ram is a variable we didn't declare
        System.out.println("a");
        System.out.println('a');
        //System.out.println('ab');//invalid
        System.out.println("ram");
        char a2 = 100;
        System.out.println("=====");
        System.out.println(a2);
        //char a3 = -100;//invalid
        char a4 = 0b1000001;
    }
}

```

```

System.out.println(a4);
char a5 = 00061;
System.out.println(a5);
char a6 = 0x0061;
System.out.println(a6);
char a7 = '\u0062';
System.out.println(a7);
System.out.println("unicode values for A to Z");
for(int i=65;i<=90;i=i+1){
    System.out.println(i+"...."+(char)i);
}
System.out.println("unicode values for a to z");
for(int i=97;i<=122;i=i+1){
    System.out.println(i+"...."+(char)i);
}
System.out.println("unicode values for numerics");
for(int i=35;i<=64;i=i+1){
    System.out.println(i+"...."+(char)i);
}
char a8 = 100;
//byte a9 = a8;//invalid
//short a10 = a8;//invalid
char a11 = 0;
//boolean a12 = a11;//invalid

char c1 = 'a';

//char c2 = 'ab';

//we cannot write more than one character

//with in the single quote

char c2 = 100;

System.out.println(c2);

System.out.println((int)' ');

System.out.println((int)'f');

char c3 = '\u0061';//hexadecimal //number system

System.out.println(c3);

char c4 = 0061;//octal number system

System.out.println(c4);

```

```
//short s = c4;  
//boolean c5 = false;  
//byte b = c5;
```

```
//boolean c6 = c4;  
float f = 23.34f;  
int f1 = (int)f;  
System.out.println(f1);
```

```
double f2 = 435.1234567890d;  
long f3 = (long)f2;  
System.out.println(f3);  
float f4 = (float)f2;  
System.out.println(f4);
```

```
}
```

```
}
```

char variable can hold character values, numerical values, binary, octal, hexadecimal, Unicode type values.

Difference between print() and println():

with the help of print() and println(), we can print the data on the console.

print() is printing on the console, after printing the data the cursor position is in the same line, whereas println() will print the data on the console, after printing the data the cursor position is in the next line.

```
ex: s.o.print("hi");  
    s.o.print("bye");
```

o/p: hi
bye

```
ex: s.o.println("hi");  
    s.o.println("bye");
```

o/p: hi
bye

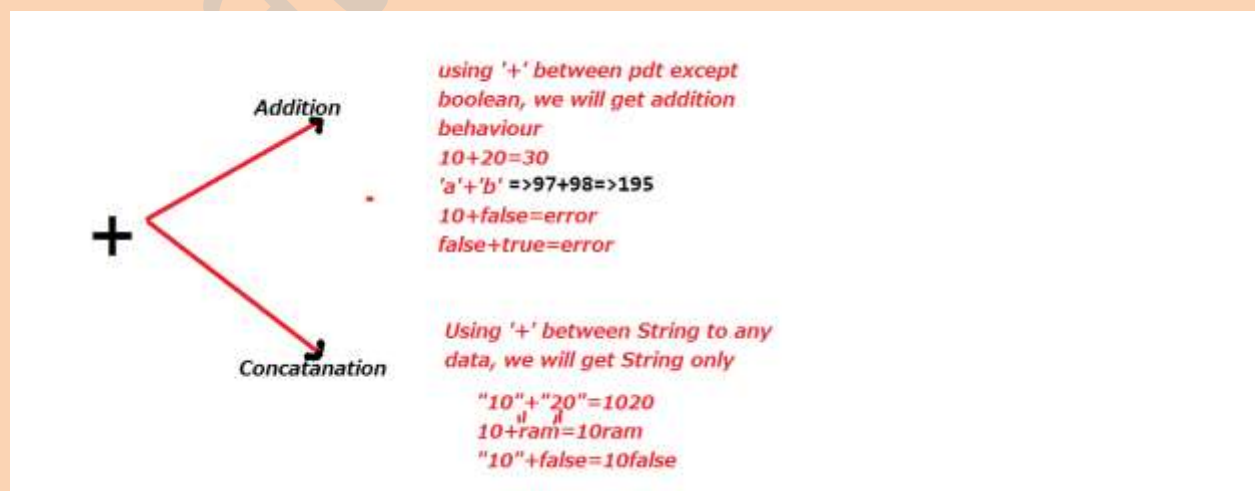
In java we have only "+" operator is overloaded.

'+' is used for both

1. concatenation
2. addition

if we are using '+' in between two number datatypes then we will get addition functionalities.

if we are using '+' in between number and string then we will get concatenation functionalities.



within the println() statement we can write

- 1)empty.
- 2)value
- 3)variable
- 4)expression
- 5)non-void method calling statement
- 6)reference/object
- 7)String
- 8)array

```
public class ExplicitCasting {  
    public static void main(String[] args) {  
        double d = 23.34D;  
        float f = (byte)(char)(int)(short)(int)(double) (long)(float)d;  
        byte b1 = 100;  
        byte b2 = 20;  
        //byte b3 = b1+b2;  
        //byte+byte=int.  
        int a = 111;  
        int b = 222;  
        int c = a+b;  
        //int+int = int;  
        byte b3 = 100;  
        int b4 = 100;
```

```
//short b5 = b3+b4;
//byte+int=int
short b6 = 100;
short b7 = 200;
//short b8 = b6+b7;
//short + short = int
//short b9 = b3+b7;
//byte+short=int
System.out.println("10+20: "+(10+20));
int f1 = 111;
int f2 = 222;
System.out.println("f1"+" "+"f2:"+" " + (f1+f2));
System.out.println(11+"ram");
System.out.println("ram"+22);
System.out.println("ram"+"22");
System.out.println(10+20+"ram");
System.out.println(10+20+"ram"+10+20);
System.out.println(10+20+"ram"+(10+20));
//System.out.println(10+false);
System.out.println("ram"+'A');
System.out.println('A'+32);
//int+char = int
System.out.println(32+'A');
long ll = 34l;
```



```

        float ff = ll;
        long l2 = ll+ff;
        //long+float=float
    }
}

class ExplicitCasting{
    public static void main(String[] s){
        long a1 = 130;
        float a3 = (byte)a1;//130--> (-126) --> -126.0
        System.out.println(a3);
    }
}

/*    long a1 = 130;
    float a3 = (byte)a1;//130-->
    System.out.println(a3);

```

According to above statements we can give one conclusion that is based on cast

operator and destination type value will change.

```
*/
```

Comments:

If we want to write any statements not related to Java grammar rule then we can write within the comments.

Comments are used to provide the information about Java elements.

Whatever the code is available under the comments, that code is simply ignored by the compiler; that comment source code is not converted into byte code. So JVM is also ignored.

With the help of the comments we can easily understand a program.

Java provides three types of comments

1. Single line comment. (//)
2. Multiline comment. (/*.....*/)
3. Document comment. (/**.....*/)

Single Line Comment:

By using this comment we can write only one line of code.

This can be represented with "//" (two forward slashes)

Multiline comments:

By using this comment we can comment more than one line of code.

This can be represented with "/*.....*/".

Document comment:

This is also same as multiline comment; By using this we can comment more than one line of code.

This is represented with "/***/"

Example:

```
/*Author: Ram
```

```
DOW:14/12/2015
```

```
Vendor: nit
```

java Version: 1.7

```
*/  
  
public class CommentDemo {  
    //Without main method we can't execute the //program.  
    public static void main(String[] args) {  
        /**  
        * int a = 10;  
        *     int b = 20;  
        *System.out.println(a+b);  
        */  
    }  
}
```

Valid comments:

```
// // // //  
// /* */  
// /** */  
/* // */  
/* /* */  
/** /** */  
/** /* */
```

Invalid comments:

```
// /* (invalid)
```

```

//    /**    (invalid)

//    */    (invalid)

/*  */ */

/**/** */

/**/* */

/**  */ */

/* /** */ */

```

Escape characters:

```

class EscapeDemo{
    static public void main(String...ram){
        //System.out.println("hibye");//hibye
        //System.out.println("hi\nbye");    //hi
                                                //bye
        //System.out.println("hi\tbye");//hi    bye
        //System.out.println("hi\bbye");//hbye
        //System.out.println("hi\ bbye");//error
        //System.out.println("hihi\bbye");//hihbye
        //System.out.println("hi\rbye");//bye
        //System.out.println("hihi\rbye");//byei
        //System.out.println("hi\fbye");
        //System.out.println("hi\'bye");//hi'bye
        System.out.println("hi\"bye");//hi"bye
    }
}

```

}

Note: Don't give any space between '\ ' and character (t, n, b, r, f, ', ", \).

Variable:

Variable is a named memory location, it can be holds the value, the value can be change from one statement to another statement.

```
int a = 10;//value --10
```

```
a = a+1; //value--11
```

```
a = a*2;//value--22.
```

Based on the data types variables can be classified into two types.

1. Primitive Variable
2. Referenced Variable

Primitive Variable:

A variable, which can define with the help of primitive data type is called primitive variable.

```
Ex: int x=10;
```

```
float y=45.56f;
```

Drawbacks of Primitive Variable:

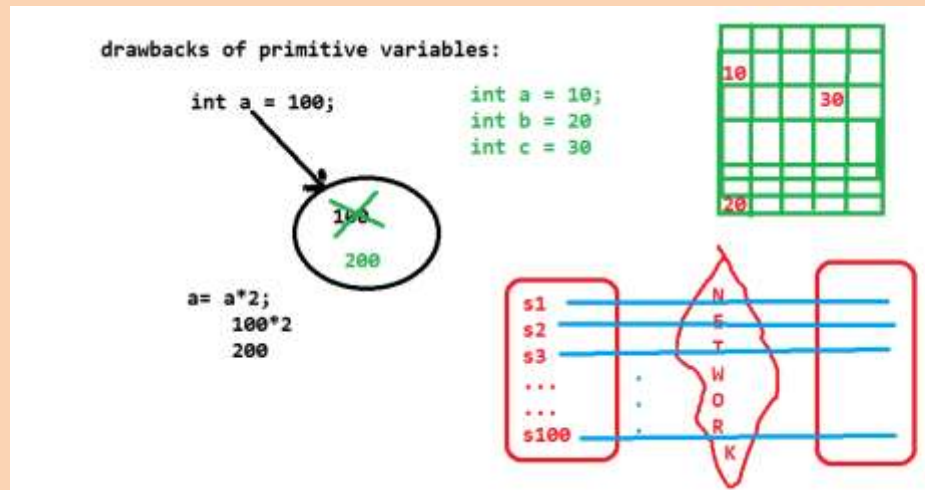
1. Primitive variables can hold only single value at a time.

It is unable hold more than one value.

2. If we are placing the data into memory with the help of primitive variables, the data is not stored in the memory in sequential order, the data will be stored in random order.

While retrieving the data from memory will take more time.

3. If we are sending the information from machine to another machine with the help of primitive variable, will take more network calls.



If we want resolve above draw backs then we can go for referenced data types.

1. Array
2. Class
3. Interface
4. Enum
5. Annotation

Array: Array is a single entity which holds continuous memory locations, which hold more than one value with same data type or its lower range data types.

```
int a[] = {10,20,30};
```

Each and every element can recognize, with the help of index position.

Within the one index position, we can write only one value.

Q) How to represents arrays in java?

- Holding multiple values
- We need "{}"
- To differentiate one value to another we need ','
- Statements ended with semicolon';'
- For assignment we need '='
- To hold value we need Variable
- To represents array '[]'
- To represent type of values data type

```
int[]a = {10, 20, 30, 40, 50};
```

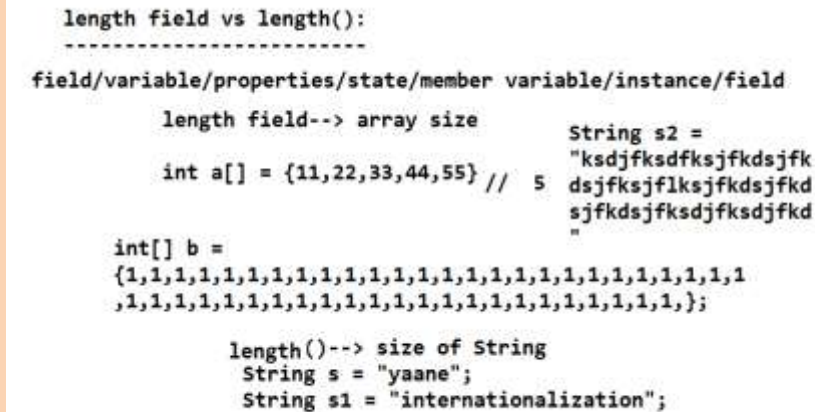
```
array:
-->is a single entity
-->contains mutiple memory locations for
-->holding multiple values of
-->same type(homogeneous)/lower range type/
type(hetrogeneous).
-->in a sequence order.

Q)What is the use of index?
A) it is used to read the data or write the data from or into array.

Q) Why should we use index?
A) We can not determind the memory locations/address of values.
The memory locations are changed from one time execution to another
time.
To resolve these problem we should we use index.
```

```
How to represents arrays in java?
-->holding multiple values
--> we need "{}"
-->to differentiate one value to another we need ','
-->statements ended with semicolon';'
-->for assignment we need '='
-->variable
-->to represents array '[]'
-->to represent type of values datatype

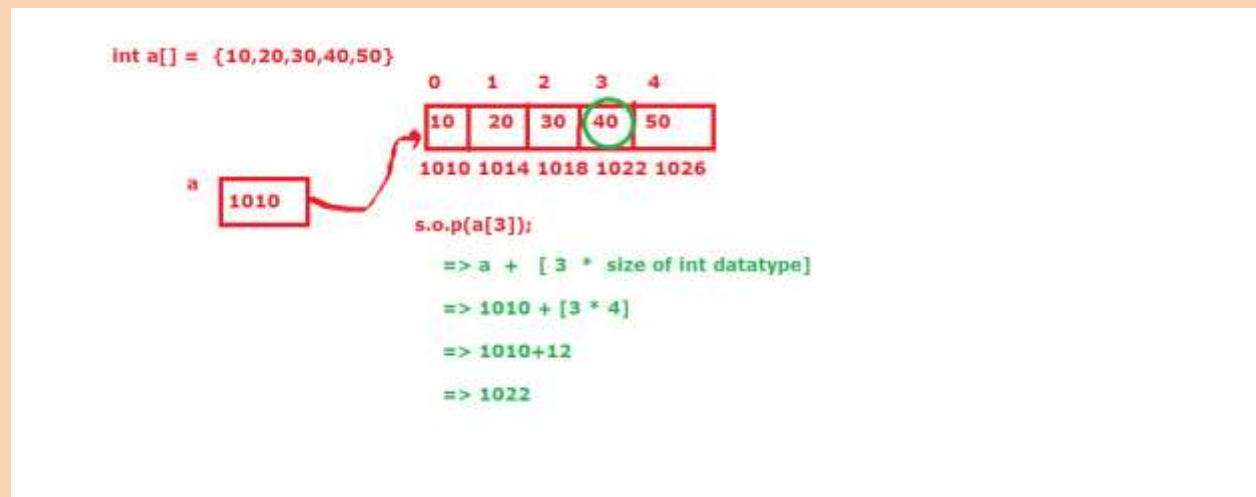
int[]a = {10, 20, 30, 40, 50};
```



```
int a[] = new int[8];  
a[0] = 10;  
a[1] = (byte)10;  
a[2] = (short)10;  
a[3] = 'a';  
a[4] = (long)10;  
a[5] = 10.00f;  
a[6] = 10.00;  
a[7] = false;
```

```
double d[] = new double[8];  
d[0] = (byte)100;  
d[1] = (short)100;  
d[2] = 100;  
d[3] = 100L;  
d[4] = 100.00f;  
d[5] = 100.00;  
d[6] = 'a';  
d[7] = false;
```

Note: With the help of castoperator we can do both implicit and explicit casting.



Representation of arrays:

We have three ways to represents arrays.

1. `int a[] = {11,22,33,44};`

2. `int b[] = new int[5];`

//here 5 is called size or dimentions

//mention the size is mandatory.

3. `int c[] = new int[]{11,22,33,44,55};`

Arrays can be represents with help of "[]".

Difference between length field and length ():

Length filed can be applicable on array variables, to know the size of an array.

```
int a [] = {0,1,2,3,4,5,6,7,8,9};
```

```
s.o.p(a.length);//10
```

```
String s[] = {"kiran","ram","suji","varun","kishore"};
```

```
s.o.p(s.length); //5
```

```
s.o.p(s[4].length()); //7
```

length (): This is used for to know the number character are existed in string.

```
String s = "internationalization";  
s.o.p(s.length()); //20  
String s1 = "ram chandra rao";  
s.o.p(s1.length()); //15
```

Arrays can be classified in the following manner.

1. Single dimensional array
2. Double dimensional array
3. Multi dimensional array

Single dimensional array:

Store the elements either in horizontal (rows) or vertical manner (columns).

```
Ex: int a[] = {111,222,333,444,555};
```

```
public class ArrayDemo {  
    public static void main(String[] args) {  
        int a[]={111,222,333,444,555};  
        /*System.out.println(a[0]);  
        System.out.println(a[1]);  
        System.out.println(a[2]);  
        System.out.println(a[3]);  
        System.out.println(a[4]);*/  
        for(int i = 0; i < a.length; i = i+1 ){  
            System.out.println(a[i]);  
        }  
    }  
}
```

```

    }
}
}

```

Randomly selecting the data from array by using Random class:

```

import java.util.Random;
public class ArrayDemo {
    public static void main(String[] args) {
        int a[] = {11,22,33,44,55};
        Random r = new Random();
        int count=0;
        while(count<a.length){
            for(int i=0;i<a.length;i++){
                int rsn = r.nextInt(60);
                if(rsn==a[i]){
                    System.out.println("rsn: "+rsn);
                    count++;
                    break;
                }
            }
        }
    }
}

package array;

import java.util.Scanner;

public class ArrayDemo {
    public static void main(String[] args) {
        /*System.out.println("main method");
        int[] a = {11,22,33,44,55};
        System.out.println("=====for loop=====");
        for(int i=0;i<a.length;i++){
            System.out.println(a[i]);
        }
        System.out.println("=====while loop=====");
        int j=0;
        while(j<a.length){
            System.out.println(a[j]);
            ++j;
        }
        */
    }
}

```

```

System.out.println("=====do-while loop=====");
int k=0;
do{
    System.out.println(a[k]);
    ++k;
}while(k<a.length);
System.out.println("=====for-each loop=====");
for(int a1 : a){
    System.out.println(a1);
}*/

```

```

/*int[] a = {11,22,33,44,55};
System.out.println("=====for loop=====");
for(int i=0;i<a.length;i++){
    System.out.println(a[i]);
}
System.out.println("enter to change the values");
for(int i=0;i<a.length;i++){
    System.out.println(a[i]);
    a[i]=new Scanner(System.in).nextInt();
}
System.out.println("=====for loop=====");
for(int i=0;i<a.length;i++){
    System.out.println(a[i]);
}*/

```

```

int[] a = {11,22,33,44,55};
System.out.println("=====for-each loop=====");
for(int a1 : a){
    System.out.println(a1);
}

```

```

System.out.println("enter the data for change");

```

```

for(int a1 : a){
    System.out.println(a1);
    a1=new Scanner(System.in).nextInt();
}

```

```

System.out.println("=====for-each loop=====");
for(int a1 : a){
    System.out.println(a1);
}

```

```

}
}

```