Core Java

Topics need to cover:

1. 2nd way to configure path and classpath
2. For identifier naming convention

Session 1:

Topics

1. Java history
2. Java structure about Byte code and Unicode.
3. Difference between jdk, jre, jvm, jit compiler
4. Java installation – software

JAVA

1. **Java was released on 25-5-1995 by sun micro system.**
2. **Initially was for small device (like mobile)**

Father--------------Gosling.

1st name-----------OAK but change to java because a language OAK was already present at that time

1. **How C programming works**

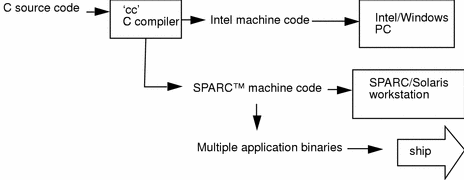
**Source code ->hello.c**

**Compiler -> compile the program**

**Executable file ->hello.exe(native code/object code)**

* **No intermediate file generated**

**Traditional Compile-Time Environment**



1. **How Java programming works**

**Source code ->Hello.java**

**Compiler ->java compiler(JVM)**

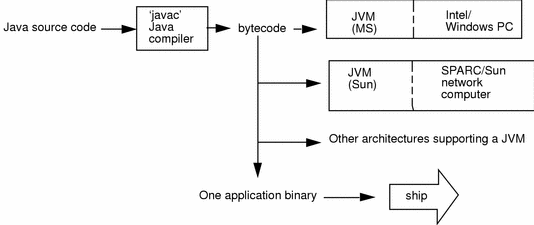
**Intermediate code ->Hello.class (byte code)**

**Interpreter ->java interpreter(JVM)**

**Executable file ->file(native code) // what is the file format?**

**Intermediate compiled file(.class) has been generated by compiler**

**New Portable Java Compile-Time Environment**



1. Characteristicsof java

Platform Independent

Object oriented

Portable

Support networking

Robustness

Security

Simple and secure

Java has the following characteristics:

* **Object oriented** - Java provides the basic object technology of C++ with some enhancements and some deletions.
* **Architecture neutral** - Java source code is compiled into architecture-independent object code. The object code is interpreted by a Java Virtual Machine (JVM) on the target architecture.
* **Portable** - Java implements additional portability standards. For example, ints are always 32-bit, 2's-complemented integers. User interfaces are built through an abstract window system that is readily implemented in Solaris and other operating environments.
* **Distributed** - Java contains extensive TCP/IP networking facilities. Library routines support protocols such as HyperText Transfer Protocol (HTTP) and file transfer protocol (FTP).
* **Robust** - Both the Java compiler and the Java interpreter provide extensive error checking. Java manages all dynamic memory, checks array bounds, and other exceptions.
* **Secure** - Features of C and C++ that often result in illegal memory accesses are not in the Java language. The interpreter also applies several tests to the compiled code to check for illegal code. After these tests, the compiled code causes no operand stack over- or underflows, performs no illegal data conversions, performs only legal object field accesses, and all opcode parameter types are verified as legal.
* **High performance** - Compilation of programs to an architecture independent machine-like language, results in a small efficient interpreter of Java programs. The Java environment also compiles the Java bytecode into native machine code at runtime.
* **Multithreaded** - Multithreading is built into the Java language. It can improve interactive performance by allowing operations, such as loading an image, to be performed while continuing to process user actions.
* **Dynamic** - Java does not link invoked modules until runtime.
* **Simple** - Java is similar to C++, but with most of the more complex features of C and C++ removed.

Java does not provide:

* + Programmer-controlled dynamic memory
  + Pointer arithmetic
  + struct
  + typedefs
  + #define

1. **Java compiler: -----**

* **Java compiler is a program which is implemented in c language.**
* **Java compiler verify the java program such as grammatical errors,**
* **When no errors found then source code will be converted into byte code**
* **Java compiler is available as a file called “javac.exe”**
* **javac is the java compiler**

Run

javap -c <fully qualified class\_name>

* to see the bytecode for each of the methods of the named class.

For Example:

javap -c Example5

Run

javap <fully qualified class\_name>

* to see all members of a class in the command prompt.

1. **Java interpreter:-----**

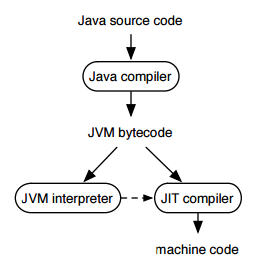
* **Java interpreter is a program which is implemented in c language.**
* **java interpreter is responsible for converting the byte code into native code and executing the native code line by line.**
* **Java interpreter is available as a file called as “java.exe”.**
* **Earlier java was implemented for small application and small memory was required for these applications. Due to small memory java interpreter was checking and loading one line at a time and then unload it for the next line into RAM. But now Java is used for large applications. So now java interpreter has been modified to improve the performance.**

**Now Java interpreter can run the whole code at time (like the compiler).**

The bytecode can be changed to the suitable language in one of two ways.

**(Difference between Java interpreter and JIT compiler)**

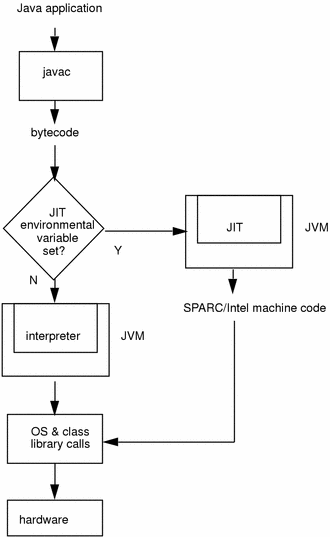
* **Interpreter**: Reads, interprets and executes the bytecode instructions one by one. As it interprets and executes instructions one by one, it can quickly interpret one bytecode, but slowly executes the interpreted result. This is the disadvantage of the interpret language. The 'language' called Bytecode basically runs like an interpreter.
* **JIT (Just-In-Time) compiler**: The JIT compiler has been introduced to compensate for the disadvantages of the interpreter. The execution engine runs as an interpreter first, and at the appropriate time, the JIT compiler compiles the entire bytecode to change it to native code. After that, the execution engine no longer interprets the method, but directly executes using native code. Execution in native code is much faster than interpreting instructions one by one. The compiled code can be executed quickly since the native code is stored in the cache.



Javac is Java Compiler -- Compiles your Java code into **Bytecode**

JVM is Java Virtual Machine -- Runs/Interprets/ translates Bytecode into **Native Machine Code**

JIT is Just In Time Compiler -- Compiles the given bytecode instruction sequence to machine code at **runtime** before executing it natively. It's main purpose is to do heavy optimizations in performance.



1. What is JIT?

JIT:

----JIT is just-in-time;

The Just-In-Time (JIT) compiler is a component of the Java Runtime Environment that improves the performance of Java applications at run time.

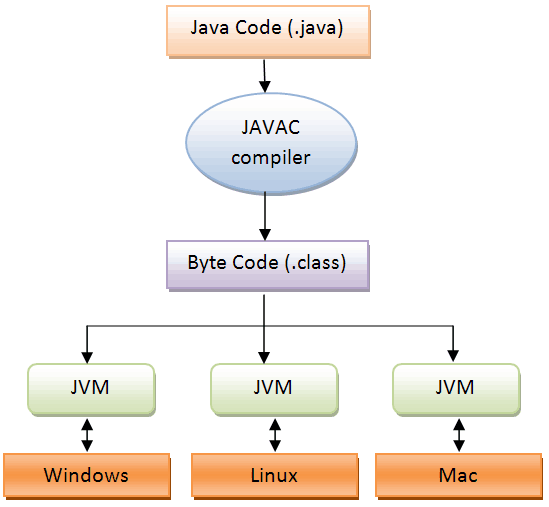
1. What is JVM?

JVM:

* **J*ava virtual machine***:

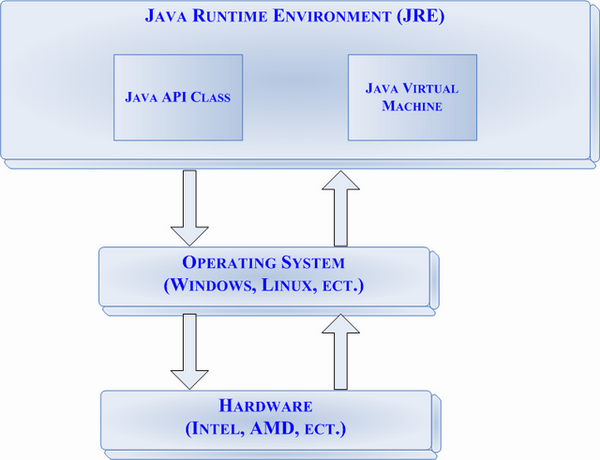
**JVM is responsible for executing the bytecodes on the specific operating system and hardware combinations.**

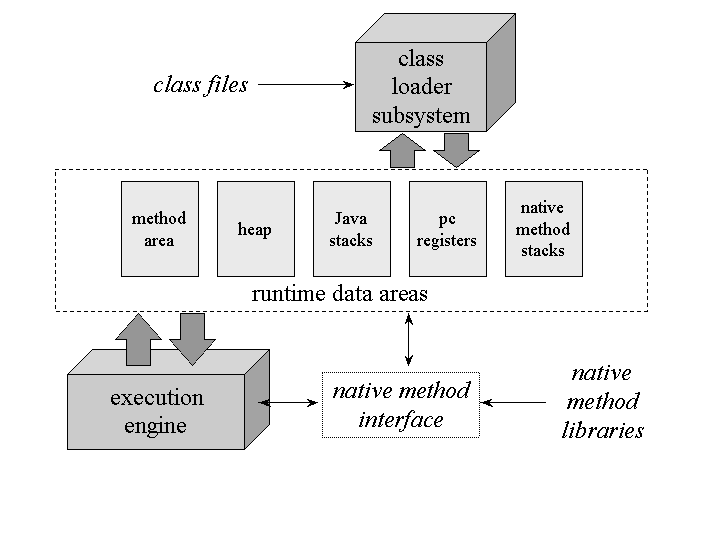
**JVM converts bytecode to native machine code which will be executed by OS.**



* JVM is a virtual platform that resides on your RAM
* Its component, **Class loader** loads the .class file into the RAM
* The **Byte code Verifier** component in JVM checks if there are any access restriction violations in your code. (This is one of the principle reasons why java is secure) i.e. Provide fundamental runtime security such as bytecode verification.
* The **Execution Engine** component converts the Bytecode into executable machine code
* This machine code will be used by the OS to execute the program.

The JVM is called "virtual" because it provides a machine interface that does not depend on the underlying operating system and machine hardware architecture. This independence from hardware and operating system is a cornerstone of the write-once run-anywhere value of Java programs. Java programs are compiled into "bytecodes" that target the abstract virtual machine; the **JVM is responsible for executing the bytecodes on the specific operating system and hardware combinations.**





Inside JVM in Detail:

About JVM:

<http://architects.dzone.com/articles/understanding-jvm-internals>

<https://www.artima.com/insidejvm/ed2/jvmP.html>

<http://www.codeproject.com/Articles/30422/How-the-Java-Virtual-Machine-JVM-Works>

<http://www.javatpoint.com/internal-details-of-jvm>

<https://kkarthikeyanblog.wordpress.com/2012/08/23/helloworld-in-jvms-view-how-java-program-executed-internally-in-jvm/>

<http://viralpatel.net/blogs/java-virtual-machine-an-inside-story/>

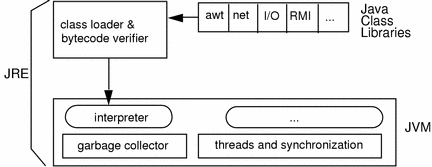
Question: what is the difference between jdk and jre?

JDK:

* Java Developer Kit contains tools needed to develop the Java programs, and ***JRE*** to run the programs.
* The tools include compiler (javac.exe), Java application launcher (java.exe), Appletviewer, etc… Compiler converts java code into byte code. Java application launcher opens a ***JRE***, loads the class, and invokes its main method.
* You need ***JDK***, if at all you want to write your own programs, and to compile the program. For running java programs, JRE is sufficient. JRE is targeted for execution of Java files i.e. **JRE** = **JVM** + Java Packages Classes(like util, math, lang, awt,swingetc)+runtime libraries.
* ***JDK*** is mainly targeted for java development. I.e. You can create a Java file (with the help of Java packages), compile a Java file and run a java file.

JRE:

Java Runtime Environment contains JVM, class libraries, and other supporting files. It does not contain any development tools such as compiler, debugger, etc. Actually JVM runs the program, and it uses the class libraries, and other supporting files provided in JRE. If you want to run any java program, you need to have JRE installed in the system

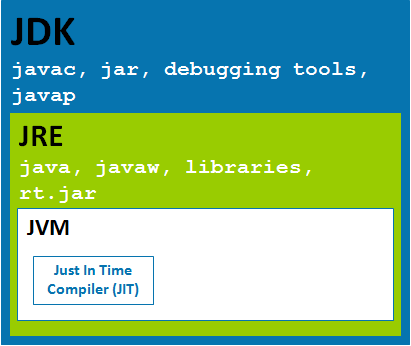


**The JRE is composed of the Java API and the JVM. The role of the JVM is to read the Java application through the Class Loader and execute it along with the Java API.**

JVM resides on RAM and java class libraries resides on hard disk.

**Jdk we can install after executing “jdk……….exe” file.**

**We can install only jre executable file to install only JVM on our system.**



Jdk = tools + jre

Jre = jvm + class libraries

Jvm=> Converts Bytecode to native machine code, jvm resides on RAM

Check T/F:----

JVM is platform independent---------------F

JDK is platform independent---------------T

Java compiler is platform independent-----T

Java interpreter (JIT) is “ “ -----F

C/c++ is platform independent: ------------F

Java is “ “ :-----------T

1. What is JDK?

JDK:

* java development kit, which is a collection of various tools to require to develop and run your java programs.

Some of the developed kits are JRE, java compiler, java interpreter and many more.

You can download the jdk from Oracle site ,when you downloading JDK you need to select the JDK version which is compatible with your OS .i.e. various software provided for various OS .you need to download one which is suitable for your OS because JDK is platform independent.

1. **Installation of java**
2. Install JDK-----------install JDK properly in your system
3. After installation we can use “java -version” command to see the version of the java.
4. Set the path and classpath variable

Two way to set the path and classpath variable

1. In the command prompt
2. In the environment variable
3. In the command prompt

path

Execute below command in the command prompt

**set PATH=%PATH%;c:\j2sdk1.4.2\_01\bin;**

classpath

**SET CLASSPATH=%CLASSPATH%;c:\someclasses\stuff.jar;**

After installation set the path and class path in command prompt as follows:--

set path=%path%;e:\j2sdk 1.4.2\bin;

setclasspath=%classpath%;c:\j2sdk 1.4.2\lib;

Path:--------- %path%;e:/j2sdk1.4.2\_04\bin;

Classpath:---- %classpath%;e:\j2sdk1.4.2\_04\lib;

Question: What is path and classpath?

Answer:

PATH and CLASSPATH are operating system level environment variables.

Question: how to set environment variable for the os using command prompt.

We can use set command to see the available values of any variables.

For ex:

set path

setclasspath

|  |  |
| --- | --- |
| prompt>**set***varname*  prompt>**set***varname***=***value*  prompt>**set***varname***=**  prompt>**set** | Display the value of the variable  Set or change the value of the variable (Note: no space before and after '=')  Delete the variable by setting to empty string (Note: nothing after '=')  Display ALL the environment variables |

What is difference between path and classpath?

* *PATH environment variable is used by operating system; this is true for both Windows and Linux environment while CLASSPATH is used by Java ClassLoaders (JVM) to load class files.*

Question: How OS use path and JVM use Classpath variables?

Answer:

Whenever you type something into a command prompt, if it is not a system command then it will search in the directory that you have already set in the PATH variable.

For example:

1.

>notepad

Os will search for “notepad.exe” file inside the path variables.

If this file found then it’ll be executed (.exe).

If this file will not found then it’ll display below error

'xx' is not recognized as an internal or external command,operable program or batch file.

1. When we type javac or java with classname, in the similar manner thenos will check in the path variables, if it’s defined there, then the executable file will run.

When you type

>javac classname.java

or,

>javaclassname

then first os will search forjavac or java in path variable and execute them.

And after java or javac execution the JVM will search in the classpath variables for classname.java or classname for further execution.

* JVM will search first in the current and if not found in folder then it will search in classpath variables.
* When set a path for binary then these binary will be accessible in all the folders and drives of your OS.

Path: --------- %path%;e:/j2sdk1.4.2\_04\bin;

* When you set a classpath for library then these classes will be available in all the folders of your OS(personal computer).

Classpath: ---- %classpath%;e:\j2sdk1.4.2\_04\lib;

Note:

1. After setting path of $JAVA\_HOME\bin, we can execute the “java” command from any folder.
2. After setting the classpath of $JAVA\_HOME\lib, we can execute the “javac” command from any folder.

But the argument available after java or javac command should be also be available in the current folder.

Because JVM search the classname available after the java, javac from the current folder first, then from the classpath.

So if we set the classpath of the “classname” folder then we can execute from anywhere.

Question:

Try the execution after setting the path and classpath as below

At folder1 contains classname

At folder2 run javac classname.java

At folder3 run java classname

Set path for $JAVA\_HOME\bin

Set classpath for folder1, folder2

Question: After execution of the “javac classname.java”,wherewill the .class be generated?

**These topic may go in java package chapter**

Q. Difference between Knowledge and general Knowledge.

What are the ways available to set the path and classpath?

There are two ways: ----

Using the command prompt

Using environment variable

My computer ------properties-----advance tab-------environment variable ---click on new button under user variable

Variable name----------path

Variable value --------- %path%;e:/j2sdk1.4.2\_04\bin;

Variable name----------classpath

Variable value --------- %classpath%;e:\j2sdk1.4.2\_04\lib;

The path &classpath what are you setting in dos prompt is temporary where as it is permanent in environment variable / system variable.

Coding:

1. Write first java program for hello?

|  |
| --- |
| class Example1{  public static void main(String [] args){  System.out.println("Hello First Java Program");  }  } |

1. Write a java program with command line arguments.

|  |
| --- |
| class Example2{  public static void main(String[] args){  System.out.println("Hi 2nd Java Program");  System.out.println("Total no of arguments ="+ args.length);  System.out.println(args[0]);  System.out.println(args[1]);  System.out.println(args[2]);  //System.out.println(args[5]);//Exception in thread "main" //java.lang.ArrayIndexOutOfBoundsException: 5  }  } |

Output:

D:\javaclass\programs>java Example2 12 gfhhemant

Hi 2nd Java Program

Total no of arguments =3

12

gfh

hemant

1. Write java program for addition, substraction, multiplication and division?

Java language

1. Character set
2. Keyword/Reserved words
3. Identifier/User Defined words
4. Data type
5. Variable
6. Constants
7. Literals
8. Operators
9. Control Statement
10. Array
11. Character Sets:

Digits 0-9

Alphabets a-z A-Z

Special Symbol + -\* / % $\_ !<> { } [ ] ( ) = ~ ^ ; : ?

White Space Character

New Line\n

Carriagereturn \r

Tab \t

Backspace \b

Question: Write a java program to display white space characters?

|  |
| --- |
| class Example3{  public static void main(String[] args){  System.out.println("Hi 3rd Java Program");  System.out.println("New Line \n character ");  System.out.println("carriage return\r ");  System.out.println("tab\tcharacter");  System.out.println("backspace\bcharacter");  }  } |

Output:

Hi 3rd Java Program

New Line

character

arriage return

tab character

backspaccharacter

1. Keyword:

Keywords are predefined words available in java. These are defined for specific purpose and cannot be used as a user defined words.

There are total 50 keywords among them 2 are not being used.

Data type(8): ------- boolean(1b), char(2B), byte(1B), short(2B), int(4B), long(8B), float(4B), double(8B)

Control Statements(11): --- If, else, for, do, while, switch, case, break, default, continue, goto

Class/object(11): -----class, interface, new, return, extends, implements, this, super, new, instanceof

Modifiers (9): ------ static, final, abstract, volatile, assert, native, strictfp, synchronization, transient,const

Access Modifiers (3): ---- public, private, protected.

Package(2): ----package, import

Exception handling(5):--catch, throws, finally, try, throw

Misc(1): void

The keywords const and goto are reserved,but they are not currently used.

 true,false, and null might seem like keywords, but they are actually literals; you cannot use them as identifiers in your programs.

1. Identifiers/User defined words:

Sometimes while writing the code you will need some name for variables, constants, methods, classes, etc. these names are called as user defined words or Identifiers

When you are writing the identifiers you should follow following rules: -------------

1. Identifiers can contain alphabets, digits and two special symbols \_,$.
2. 1st character must be an alphabets or $ or\_.
3. Keywords shouldn’t be used as identifiers.

123abc F

Abc 123 F

if9 T

if F

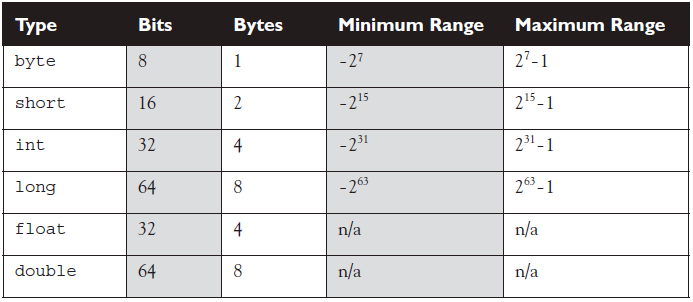
int T

\_123 T

1. Data Types

There are two kind of data types:

1. Primitive data types:
2. Reference data types:
   * + 1. Primitive Data types:



b1 = 127

b2 = -128

s1 = 32767

s2 = -32768

i1 = 2147483647

i2 = -2147483648

l1 = 9223372036854775807

l2 = -9223372036854775808

char 16 8

boolean true, false

* The positive range is one less than thenegative range because the number zero is stored as a positive binary number.
* Weuse the formula -2(bits-1) to calculate the negative range, and we use 2(bits-1)–1 for thepositive range.

**Question:**Why java character is taking two bytes?

**Answer:**

To provide the internalization facility. Which allow you to support multiple languages.

* + - 1. User Defined data types/ Reference type:
* Class type
* Interface type
* Enum type
* Annotation type

E.g.:

String

1. Variable:

* Variable is the container which holds user data.
* Memory will be all
* Value of the variable can be changed any number of times during program execution.

Syntax:

<Data type><var\_name>;

<Data type><var\_name> = <value>;

int a;

int b = 10;

String str;

String str1 = “abce”;

Question:

Write a program to declare all primitive type data and String type data.

|  |
| --- |
| class Example4{  public static void main(String[] args){  int a;  int b = 10;  byte by1;  byte by2 = 122;  short sh1;  short sh2 = 23455;  long l1;  long l2 = 34456l;  float f1 = 23432.345f;  float f2;  double d1 = 3234324.534;  double d2;  System.out.println(b);  System.out.println(by2);  System.out.println(sh2);  System.out.println(l2);  System.out.println(f1);  System.out.println(d1);  //Example4.java:31: variable a might not have been initialized  //System.out.println(a);  //System.out.println(by1);  //System.out.println(sh1);  //System.out.println(l1);  //System.out.println(f2);  //System.out.println(d2);  }  } |

Variable Declaration

Different way to declare the variables:

1. Variable declaration without value

Example;

int a;

2.Varibable declaration with value

Example:

Int b =10;

String str = “Hi! Welcome in java program.”;

* + - 1. Variable declaration with modifiers:

Syntax:

[modifiers] <data type><variable\_name>;

[modifiers] <data type><variable\_name> = <value>;

* + - 1. Variable declaration of multiple variables in single line

Syntax:

[modifiers] <data type><variable\_name1>, <variable\_name2>, <variable\_name3>, <variable\_name4>;

For example,

inta,b,c;

String str1, str2, str3, str4;

* **Initialization of variables:**

1. Initialization of one variable at a time.

a = 10;

b = 23;

str1 = “Hi !”;

str2 = “Hemant Kumar”;

1. Initialization of multiple variables at a time.

a = b = c = 10;

str1 = str2 = str3 = “Kumar”;

Note: In above examples we initialize the variables after the declaration in the separate line.

We can declare and initialize the variable together in a single line.

* + - 1. Declare variable with initialization:

Syntax:

<Data type><variable\_name1> =< value1>, <variable\_name2> =<value2>, <variable\_name3>= <value3>;

Ex:

int a = 10, b= 20, c=30;

String str1= “Hi”, str2 = “Hello”, str3 = “welcome”;

boolean b1 = true;

boolean b2;

boolean b3 = true, b4 = false, b5 = true;

char c1;

char c2 = ‘a’;

char c3 = ‘b’, c4 = ‘5’, c5 = ‘H’;

Note: if we are declaring multiple variable in a line then we can declare only one kind of data type at a time.

This is also applicable with multiple initializations at a time.

Question: Write a program to declare and initialize all kinds of data type in each and every way.

1. Constant:

* Constant is a special variable whose value can’t be modified during the program execution.
* Constant is also called as a final variable.
* A variable that is declared as final and not initialized is called as **blank final variable.**

Syntax:

[modifier] final <data type><var\_name> = <value>;

[modifier] final <data type><var\_name>;

<var\_name> = <value>;

* If the variable is declared as final and initialized in the same statement then in the “.class” file compiler will replace that variable with the actual value.

Question:

1. Program to demonstrate the modification of the value of normal variable.
2. Program to demonstrate the modification of the value of constant variable or final variable.
3. Program to demonstrate the use of constant variable without initialization.

class Example5{

public static void main(String[] ar){

int a;

System.out.println("a = " + a);

a = 10;

System.out.println("a = " + a);

final int b = 11;// declaration with initialization

System.out.println("b = " +b);

final int c;

//System.out.println("c = " + c);// compile time error

c = 33;

System.out.println("c = " +c);

//b = 34;// compile time error

//c = 45;//compile time error

a = 67;

System.out.println("a = " +a);

System.out.println("b = " +b);

System.out.println("c = " +c);

}

}

1. Literal:

* Literal are actual values which assigns to variable or constant.
* Literal are also used to perform any operations.
* Constant value in programs is normally called as literal.

Example:

int a = 102;

Datatype Varibale name Literal

String str = “Welcome to Java!”;

Types of Literals:

1. Integral Literal
2. Floating Literal
3. Character Literal
4. Boolean Literal
5. String Literal
6. Null Literal
7. Integral Literal

* Integral literal can be assigned to one of the integral data type: byte, short, int, long
* Size of the integral literal, depends on the data type.
* Default type of Integral literal is “int”.
* There are four types of Integer Literals:

1. Decimal literals
2. Octal literals
3. Hexadecimal literals
4. Binary Literals (from Java 7)
5. Decimal Literals:
6. A valid number form decimal number system is decimal literal.
7. The base or radix of decimal number system is 10.
8. It consists 10 digits ranging from 0 to 9 to allow forming the decimal literals.
9. Decimal literals must not start with 0.

|  |  |
| --- | --- |
| **Data Type** | **Default Value (for fields)** |
| byte | 0 |
| short | 0 |
| int | 0 |
| long | 0L |
| float | 0.0f |
| double | 0.0d |
| char | '\u0000' |
| String (or any object) | null |
| boolean | false |

For Example:

12345

98989898

Note:- Long type integral literal must be followed by l or L .

Question: Write a program to demonstrate the assignment of decimal literal to byte, short, int, long.

Question: Write a program to demonstrate the assignment of maximum decimal literal to byte, short, int, long.

Program:

class Example6{

public static void main(String[] arg){

byte b1 = 127;

byte b2 = -128;

//byte b3 = 128;

//byte b4 = -129;

// without declare if we use // compile time error: cannot find symbol

System.out.println("b1 = " + b1);

System.out.println("b2 = " + b2);

//System.out.println("b3 = " + b3);// compile time error: possible loss of precision

//System.out.println("b4 = " + b4);// compile time error: possible loss of precision

short s1 = 32767;

short s2 = -32768;

//short s3 = 32768;

//short s4 = -32769;

System.out.println("s1 = " + s1);

System.out.println("s2 = " + s2);

//System.out.println("s3 = " + s3);// compile time error: possible loss of precision

//System.out.println("s4 = " + s4);// compile time error: possible loss of precision

int i1 = 2147483647;

int i2 = -2147483648;

//int i3 = 2147483648;

//int i4 = -2147483649;

System.out.println("i1 = " + i1);

System.out.println("i2 = " + i2);

//System.out.println("i3 = " + i3);// compile time error: integer number too large: 2147483648

//System.out.println("i4 = " + i4);// compile time error: integer number too large: -2147483649

//long l1 = 9223372036854775807;// compile time error: integer number too large: 9223372036854775807

//long l2 = -9223372036854775808;// compile time error: integer number too large: -9223372036854775808

long l1 = 9223372036854775807l;

long l2 = -9223372036854775808l;

//long l3 = 9223372036854775808l;

//long l4 = -9223372036854775809l;

System.out.println("l1 = " + l1);

System.out.println("l2 = " + l2);

//System.out.println("l3 = " + l3);// compile time error: integer number too large: 9223372036854775808

//System.out.println("l4 = " + l4);// compile time error: integer number too large: -9223372036854775809

}

}

Output:

b1 = 127

b2 = -128

s1 = 32767

s2 = -32768

i1 = 2147483647

i2 = -2147483648

l1 = 9223372036854775807

l2 = -9223372036854775808

1. Octal literals

* A valid number of octal number system is octal literal.
* The base or radix of octal number system is 8.
* Only 8 digits ranging from 0 to 7 are allowed to form the octal literals.
* Octal literals must start with 0 (zero)

For example:

0101

0567

05432347

Question: Write a program to demonstrate the assignment of octal number system.

Program:

class Example7{

public static void main(String[] ar){

System.out.println(0101);//65

byte b = 0177;//127

//byte b1 = 0200;

short b1= 0200;//128

System.out.println(b);

System.out.println(b1);

}

}

1. Hexadecimal literals

* A valid number of hexadecimal number system is hexadecimal literal.
* The base or radix of hexadecimal number system is 16.
* Only 16 digits ranging from 0 to 9 and A/a to F/f are allowed to form the hexadecimal literals.
* hexadecimal literals must start with 0x/0X.

For Example:

0x12fd43

0Xfdabc4336

Question: Write a program to demonstrate the assignment of hexadecimal number system.

1. Binary Literals (From java7)

* A valid number of binary number system is Binary Literals.
* The base or radix of binary number system is 2
* Only 2 digits 0 and 1 are allowed to form the binary literals.
* Binary Literal must start with 0B/0b.

For Example:

0b1010

0B11110000

Question: Write a program to demonstrate the assignment of binary number system.

1. Floating Point Literal

* Floating Point Literal can be assigned to one of the floating point data type:

float, double

* Size of the Floating point literal depends on floating point data type
* Default type of floating point literal is “double”.
* You can use D/d as a suffix for the double value optionally.
* You must use f/F as a suffix for float value.
* Floating Point Literals can be represented using following notations:
  + - Standard floating point notation
    - Exponent/scientific notation
    - Hexadecimal notation (from java5)
    - Standard floating point notation
* It consist of a whole number followed by a decimal point and fractional part.

12345.56

4554.66455

Program:

* + - Exponent/scientific notation
* The exponent is indicated by an E or e followed by a decimal number, which can be positive or negative.

123.4567e2 -> 123.4567\*102

123.4567E+2 -> 123.4567\*102

123.4567E-2 -> 123.4567\*10-2

Program:

* + - Hexadecimal notation (from java5)
* It allows literals of the float and double to be written primarily in base 16 rather than base 10.

double x = 0XaP0;

a -> 10

P0 -> 20

aP0 -> 10\*20

double y = 0XfP2D;

f -> 15

fP2 -> 15\*22

class Example8{

public static void main(String[] args){

float f = 34.45f;

double d = 67.54;

System.out.println(f);//34.45

System.out.println(d);//67.54

float f1 = 434325346545465.54345f;

double d1 = 3454356467576878980945436547.67687;

System.out.println(f1);//4.34325347E14

System.out.println(d1);//3.454356467576879E27

double d2 = 453465464.565d;

System.out.println(d2);//4.53465464565E8

float f3 = 3.454356467576879E27f;

double d3 = 453465464.565E50D;

System.out.println(f3);//3.4543565E27

System.out.println(d3);//4.53465464565E58

float f4 = 0xff;

System.out.println(f4);//255.0

double d4 = 0xffP0d;

float f5 = 0xffp1f;

System.out.println(d4);//255.0

System.out.println(f5);//510.0

}

}

Some more example:

Note:

**Underscore in numeric literals(from java7)**

* + - To represent the unit of the values you can use comma(,) symbol in number representation.

E.g. :

2,45,12,452

* + - You can do the same thing with underscore(\_) from java 7 onwards.

For E.g.:

2\_45\_12\_452

1. Boolean Literal

* Boolean Literal can be assigned to a “boolean” type variable.
* There are two Boolean literals

true

flase

Question: Write a program to demonstrate the assignment of Boolean literal.

class Example9{

public static void main(String[] args){

boolean b1 = true;

boolean b2 = false;

System.out.println(b1);// true

System.out.println(b2);// false

b2 = b1;

System.out.println(b1);// true

System.out.println(b2);// true

//boolean b3= 1; //Compile time Error: incompatible types

//boolean b4 = True;//Compile time Error: cannot find symbol

boolean b4 = b2;

System.out.println(b4);//true

}

}

1. Character Literal
   * + Character Literal is a single character enclosed within a single quotation mark.
     + Character Literal can be assigned to “char” type variable.

For Example

char c1 = ‘1’;

char c2 = ‘b’;

Question: Write a program to demonstrate the assignment of Character literal.

class Example10{

public static void main(String[] args){

char c1 = '1';

char c2 = 'h';

System.out.println(c1);//1

System.out.println(c2);//h

c1 = c2;

System.out.println(c1);//h

System.out.println(c2);//h

char c3 = '^';

System.out.println(c3);//^

//char c4 = '';// empty character

char c5 = ' ';

//System.out.println(c4);

//Example10.java:18: empty character literal

//char c4 = '';

// ^

//Example10.java:18: unclosed character literal

//char c4 = '';

// ^

//Example10.java:18: ';' expected

//char c4 = '';

// ^

System.out.print(c5);

System.out.println("asdf");// asdf

char c6 = ' ';

System.out.print(c6);

System.out.println("asdf");// asdf

//char c7 = '23';

//Example10.java:38: unclosed character literal

//char c7 = '23';

// ^

//Example10.java:38: unclosed character literal

//char c7 = '23';

// ^

//Example10.java:38: not a statement

//char c7 = '23';

// ^

//char c8 = '\';// compilet time error

char c9 = '"';

//char c10 = ''';// compile time error

char c11 = '/';

System.out.println(c9);//"

System.out.println(c11);///

}

}

Escape Sequence:

* + - Escape sequence is a special notation which is used to represent some special characters which can’t be represented as it is.

|  |  |
| --- | --- |
| Escape Sequence | Description |
| \t | Tab space |
| \b | Back space |
| \n | Newline |
| \r | Carriage return |
| \f | Form feed |
| \’ | Single quote character |
| \” | double quote character |
| \\ | Backslash character |

Question: Write a program to demonstrate to display in the escape sequence.

class Example11{

public static void main(String[] args){

//System.out.println("\");

//Example11.java:4: unclosed string literal

//System.out.println("\");

// ^

/\*

System.out.println("'");

Example11.java:5: illegal start of expression

System.out.println("'");

^

Systme.out.println(""");

Example11.java:6: unclosed string literal

Systme.out.println(""");

^

\*/

System.out.println("\\");//\

System.out.println("\'");//'

System.out.println("\"");//"

}

}

**ASCII Character Sets:**

* ASCII stands for *American Standard Code for Information Interchange.*
* Every character enclosed in single quotation marks will have an integer equivalent value called as ASCII value.
* ASCII value range is 0 to 255.
* ASCII value can be assigned to a char type variable.
* If we assign an integer to a char type variable then compiler will convert the integer into its corresponding ASCII character.
* If we do arithmetic operation on the character (not on the character type variable), then compiler will do the arithmetic operation on the corresponding ASCII value of that character.

Program:

**UNICODE Characters**

* Unicode stand for Universal code.
* Every Character will have Unicode value.

Program:

1. String Literal

* String Literal is a collection of zero and more characters enclosed between double quotation marks(“”).
* String Literal can be assigned to reference variable of type String.
* String Literal can be represented as array of characters in memory.

Example:

“aldfs”

“1232356677”

“Hi! Welocme to Hello World.”

1. Null Literal:

* “null” is value which can be assigned to the reference type variable.
* It is default value for any reference variable.
* If the value of the reference variable is assigned as null, it means, there is no any address is assigned to the reference variable.

Question:

Program to demonstrate the assignment of String literal and null literal.

|  |
| --- |
| class Example13{  static String str2;  public static void main(String [] args){  String str1 = "dsfdghgjvdfxc";  System.out.println(str1);//dsfdghgjvdfxc  str1 = null;  System.out.println(str1);//null  System.out.println(str2);//null  }  } |

Note:

1. The literal must be of the same size of the data type. If literal’s size is more than the data type, in which literal has to be assigned, then we’ll get compile time error

For byte: Possible loss of precision

For Shore: Possible loss of precision

For int: Integer number is too large

For long: Integer number is too large

For float: Floating point number is too large

For double: Floating point number is too large

1. Boolean literal can be only true and false.
2. Character literal must be enclosed by ‘’.
3. If we assign numeric data to the char variable then compiler converts numeric data to corresponding character using the ASCII value and store the character in the variable.
4. OPERATORS

* Operators are used to perform operation by using operands.
* There are three types of operators (according to operands)

1. Unary Operator: Only one operand is required.
2. Binary Operator: two operands are required.
3. Ternary Operator: three operands are required.

Types of Operators:

|  |  |
| --- | --- |
| **Operator name** | **Operator Symbol** |
| Arithmetic operator | +,-,\*,/,% |
| Assignment operator | =, +=, -=, \*=, /=, %= |
| Increment/Decrement operator | ++,-- |
| Relational operator | >, >=, <, <=, ==, != |
| Logical operator | &&, ||, ! |
|  |  |
| Ternary operator | ? : |
| New operator | New |
| Bitwise operator | >>,<<,&,|,~ |
| Instance operator | instanceof |
|  |  |

1. Assignment Operator:
2. Simple assignment Operator
3. Compound assignment operator
4. Arithmetic Operator
5. Unary Arithmetic Operator
6. Binary Arithmetic Operator
7. String Concatenation Operator
8. Increment & Decrement Operator
9. Relational Operator
10. Logical Operator
11. “new” operator
12. “instanceof” operator
13. Conditional or Ternary operator
14. Bitwise Operator
15. Assignment Operator
16. Simple Assignment operator:
    * + It is binary operator.
      + It’s used to assign the value to the variable.

**Syntax:**

<operand1> = <operand2>;

* + - operand1 i.e. left side operand must be variable.
    - operand2 can be variable, value or expression.

Program: Demonstrate the use of assignment operator.

|  |
| --- |
| class Example14{  public static void main(String [] args){  int i;  int j,k,l;  i = 10;  j = 11;  //10 = j;// unexpected type  k = i;  l = i+j\*k;  System.out.println(i);//10  System.out.println(j);//11  System.out.println(k);//10  System.out.println(l);//120  }  } |

Type Casting:

<operand1> = <operand2>;

* + - Here operand2 is the source and operand1 is desination.

Destination = source

* + - It means we assign the source value to the destination.
    - Usually destination type is same as source type.
    - It means

int i;

i = 10; //( int = int)

float f;

f = 12.8f;

//i = f;//(int = float)  **?**

* + - When source type is not same as the destination type then, source must be converted into the destination type.
    - **The process to convert the type one value to another type is called as Type casting.**

There are two kinds of type casting

1. Implicit type casting
2. Explicit type casting

|  |  |
| --- | --- |
| **Implicit Type Casting** | **Explicit type casting** |
| 1. When type casting is done by JVM automatically then it’s called as implicit type casting. | 1. When type casting is done by programmer explicitly then it’s called as explicit type casting. |
| Example:  long l = 4545; | Example:  //int i = 34546l;// compile time error  int i = (int)345456l; |
|  | Syntax:  **<destType> <destVar> = (<destType>) <source value>;** |
| int is getting stored into long. means int is converted into long.  This conversion is called as  Widening: lower type to higher type | Float is stored into int.  Means float is converted into int.  This conversion is called as Narrowing: higher to lower type |

Program:

|  |
| --- |
| class Example15{  public static void main(String [] args){  byte b;  int i;  b = 45;  i = b;//implicit type casting or widening  System.out.println(i);  i = 300;  //b = i;//possible loss of precision  b = (byte)i;  System.out.println("i = "+ i);//300  System.out.println("b = "+ b);//44  }  } |

300 => 00000000 00000000 00000001 00101100

byte –> 8 bit

byte data: 00000000 00000000 00000001 00101100 => 44

🡨---------------------------------------------------------------------------------

Implicit type casting (means byte literal can be assigned to short)

-----------------------------------------------------------------------------------🡪

Explicit type casting (means to assign short literal to byte, we need to do type casting)

Character assignment:

1. We can assign maximum 65536 numeric value (short, int, byte) to a char variable.
2. If we assign long, float and double value to the char variable we’ll get “possible loss of precision” compile time error.
3. If we assign greater than the maximum size we need to type cast the value to byte, short or char otherwise we’ll get compile time error as “possible loss of precision”.

Note: if numeric value is less than greater size than we can type cast into “int” also.

1. If we assign float, long and double we need to type cast the value to byte, short or char otherwise we’ll get compile time error as “possible loss of precision”.

Example:

Demonstrate the char assignment.

|  |
| --- |
| class Example16{  public static void main(String[] args){  char c1;  char c2;  char c3;  c1 = 45;  c2 = 65535;  //c3 = 65536;// possible loss of precision  //c3 = 65.6f;//possible loss of precision  //c3 = 65l;//possible loss of precision  //c3 = 65.6//possible loss of precision  System.out.println(c1);  System.out.println(c2);  //System.out.println(c3);  char c4;  char c5;  char c6;  char c7;  char c8;  char c9;  c4 = (short)65536;  c5 = (int)65.3;  //c6 = (int)65556.6;// possible loss of precision  c6 = (int)65535.78;  //c7 = (int)65536.78;//possible loss of precision  System.out.println(c4);  System.out.println(c5);  System.out.println(c6);  c7 = (byte)65536.78;  c8 = (short)65536.78;  c9 = (char)655365456.78;  System.out.println(c7);  System.out.println(c8);  System.out.println(c9);  }  } |

Numeric assignment:

1. If we assign numeric value in byte, short greater than\
2. Int, long
3. Float
4. Integer literal which is larger than the int maximum size (2147483647) cannot be type casted into int because this value is not defined as int or long.
5. Long literal which is larger than the long maximum size (9223372036854775807l) cannot be type casted into long because this value is not defined as long.

Example:

class Example17{

public static void main(String[] args){

byte b1;

byte b2;

b1 = 127;

//b2 = 128;//possible loss of precision

b2 = (byte)128;

byte b3;

b3 = (byte)129;

byte b4;

b4 = (byte)256;

byte b5;

b5 = (byte)260;

System.out.println(b1);

System.out.println(b2);//-128

System.out.println(b3);//-127

System.out.println(b4);//0

System.out.println(b5);//4

}

}

1. Compound Assignment Operator:
2. Arithmetic Operator
   1. Unary Arithmetic Operator
   2. Binary Arithmetic Operator
3. String Concatenation Operator
4. Increment & Decrement Operator
5. Relational Operator
6. Logical Operator
7. “new” operator
8. “instanceof” operator
9. Conditional or Ternary operator
10. Bitwise Operator

Arithmetic operator:-- when you combine arithmetic operator with two operant then it called as arithmetic expression. Exp:---4+4,6\*9,3-4

The result of arithmetic expression may be int/float value.

All arithmetic operators are binary operators

+,- are also unary operator.

Assignment operator: ----- All assignment operators are binary operators. Assignment operators are used to assign values of right side operant to left side operant.

Exp:--b=a here value of a is assign into the operator b.

When you use the assignment operators you should use same type of operant.

Exp:--when you write a=b then a and b must have same data type.

Sometimes you may get a requirement to assign one type of value to another type then you need to do type casting.

Casting is a process of converting value in one data type to another data type .

Syntax:------------ var1(dest.) =(data type) var2(source)

There are two type of type casting:-------------

Implicit casting

Explicit casting

When you assign one variable to another and when source is smaller and destination is bigger then JVM will do the type casting automatically, this is called as implicit casting. it is also called as widening exp:-- converting integer to double ,this will happen automatically.

When you assign one value or variable to another variable and when source is bigger and destination is smaller then you have to do the casting. This is called as explicit casting and this is also called as narrowing. Exp: -- converting double to integer.

Relational operator: ---- all relational operators are binary operators relational operators form the relational expression whose result is Boolean values.

Logical operator: ---- in this &&,||,!(binary NOT operator) is unary operator.logical operators found a logical expression whose result is a Boolean value.the operands of logical operators must be a Boolean type.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A | B | A &&B | A||B | !A |
| T | T | T | F | F |
| T | F | F | F | F |
| F | T | F | F | T |
| F | F | F | T | T |

Increment/Decrement operator :---

Ternary operator: --- ternary operator is used to perform simple condition check.

Syntax: -----var=(condition)?expression1:expression2;

Exp: -------- max=(a>b)?a:b;

Ist condition will be executed .when the condition is true then true block value is assign into the variable. When the condition will be false then false value is assign to the variable.

Instance operator: ------ this is used too check whether given object is object of given class or not.

New operator: ------ new operator is used to create the object of a class or used to allocate the memory for instance variable of the class.

Bitwise operator: ----- it is invoked on individual bits of given no(<<,>>,&,|,~,^)

<<left shift

>>right shift

& bitwise AND

| bitwise OR

~ bitwise NOT

^ bitwise XOR

1. Control statement :-------------

If statement

For statement

Do\_while

Continue

Break

Switch statement

While statement

If statement: -----------it is used to perform conditional check. It is used to perform one task between two task based on the condition.

If (condition)

{ //statement

}

Stat x;

If (condition)

{ //statement1

}

Else

{ //statement2

}

Stat x;

If (condition)

{ //statement1

}

If else (condition)

{

//Statement2

}

Else

{ //statement3

}

Stat x;

When if statement is Encrusted first condition is evaluated, if the condition is true then true block is executed. When the condition is false then statement inside the false block will be executed.

NOTE:--- inside the if condition always you should pass Boolean type variable.

If you passing a non-Boolean type the following compilation error will be accrued

Incomparable types

Found: int

Require : Boolean

For statement :------ it is used to execute set of statement multiple times.

for (initialization ;condition ;increment/decrement)

{

Statement 1 ------ statement n;

}

Statement x

When for statement is encountered then the following steps will occurs: ------------------------

Fist initialization statement will be executed (only once)

Condition will be verified

If the condition is true then statement inside the block will be executed.

Then increment/decrement statement will be executed.

After increment/decrement again condition is verified if condition is true then same process will be repeated. When condition is false then control will come out the loop and execute statement x.

While statement : ------- this statement also do the same task similar to for statement.

Syntax :----

Initialization;

While (condition)

{

Increment/decrement;

}

Statement x;

When while statement is encountered, condition will be verified first, when the condition is true then statement inside while block will be executed and again condition will be verified. Statement inside the while block will be executed as long as condition is true. Once the condition is false then control will come out the loop and then statement x will be executed.

do-while statement: ----------

This is used to execute a set of statement repeatedly, until the logical test results in false. This is called the post-test loop, because the test for repetition is made at the end of each pass.

Syntax: ------------

do

{

Statement;

}

while(condition);

{

Increment/decrement;

}

Statement x;

In do-while first all the statement in the block will be executed and then condition will be verified. When the condition will be true then loop will be repeated and when condition is false then control come out the loop.

In while statement first condition will be verified if the condition is true then loop will be repeated and condition will false then control come out the loop.

In while when condition is false first time then statement inside the block will execute Zero time whereas in do-while statement will execute at one time.

NOTE: -------At least once the body of do-while is executed. Because the logical test, for repetition of the loop, is carried out at the end of each pass.

Switch statement: ---------------------

It is used to perform single task among multiple task. Switch statement provides a multiple way branching.

Syntax: -------

Switch (Expression)

{

Case val1: statement 1; break;

Case val2: statement 2; break;

Case val3: statement 3; break;

Default: statement x;

}

When switch statement is encountered, first expression will be evaluated and resultant value of expression will be compared with values given in the cases. If any matching case is found then the statement related to the case will be executed and the control will be transferred to end of the switch statement. If matching case is not found then the default case will be executed.

NOTE :-------

Type of expression and case value is always integer.

Byte, char, short types are also allowed because these three types will be converted to integer automatically.

Boolean, long, float and double types are not allowed.

If I am not writing break cases, all the cases will be executed starting from the matching case.

We can write default case anywhere, But at the end when we are writing default we no need to write break statement after it.

Break and continue statement: ------------

Break is used to transfer the control to the end of the block unconditionally

Continue is used to transfer the control to the beginning of the block unconditionally as above.

Finally break is used to move the control to end of the loop and continue is used to move the control to the beginning of the loop.

Break statement: ------------------------

While (condition)

{

// statement

Break; // end of the loop

}

Continue statement: ------------------------

While (condition)

{

// statement

Continue; // beginning of the loop

}

Array: --------Array is collection of similar elements.

In java array are objects

Data\_type[].array\_name =new data\_type [size];

Exp: - int[].x=new int[3];

When you are declare the array we should not specify the size. When specify the size .where you are trying to use the index which is not available then array index out of bounds exception is throw at run time.

Two dimensional arrays: -----two dimensional arrays is nothing but array of arrays.

Syntax: ------- data type [] [] array\_name=new data type [size][size]

Int [][]x=new int [3][3];

Question: Write code to copy array values from one array to another.

OOPS in JAVA

Object oriented principles: ------------------------

Abstraction

Encapsulation

Polymorphism

Inheritance

Abstraction: --- providing essential property and operation of an object by hiding internal things is called as abstraction.

Exp:- -- abstracting the monitor is nothing but providing some properties like color, height, width ,company, manufacturing date, cost etc, and some operation like switch ON ,switch OFF, increase brightness, decrease brightness etc. by hiding internal functionality of the monitor.

Encapsulation: ----- placing all the properties and operation of an object at one place is called as encapsulation. The place is called as a class and properties are called as variable and operation are called as method/behavior.

*In programming language*: -------- writing a java class by placing variables and method inside that is called encapsulation

Polymorphism: ---- one form behaving differently in different situation is called as polymorphism.

Exp: -----one button in CPU will do two task called as switch ON and switch OFF.

Inheritance: ------ getting existing property and operation of an existing class to a new class is called as inheritance. The existing class is called as parent class/ base class and the new class is called as sub class/ child class/derived class.

In java: ------- the process of using the super class (parent class) functionality in subclass (child class) is called as inheritance.

Class and objects

Syntax to declare class: -----------

[Modifier] class class\_name

{

[Modifier] data type var1, var2, var3……;

----------------------------

[Modifier] return type method\_name (data type var1, datatype var2)

{

Statement1;

Staemnent2;

---------------

}

}

Exp:-- class A

{

Int a;

Int b;

Void show ()

{

System.out.println (a);

System.out.println (b);

}

}

Creating object:---------

Class\_namevar\_name=new class\_name();

Student s=new student();

A obj =new A();

When the above line is executed then the following things will happen:----------

A obj: ---------Memory will be allocated for the reference variable obj of type class A and default null have assigned .

obj

|  |
| --- |
| NULL |

8 byte

New operator: ---- this is used to allocate the memory for instance variable of the class.

In the above class A there are two instance variables called a and b , so in this new operator allocate the memory for two variables a and b .

A() :---- if constructor is available then it will be called to initialize the object or initialize the instance variable.

= :---------- object address will assign to reference variable object

In this example obj is reference variable of type A and it can store address of object of type A only.

Variable and methods of the class are called as member of the class.

Reference variable.member\_name;

Exp:---s.show(); method

s.sid ; variable

How many different types of members you can write insides the class ?

We can write four types of members in a class

Variable

Block

Method

Class

Types of variable :---------------

*Class level variable*: ----------------------------

instance variable/non-static variable

local variable

final variable

static variable

reference variable

*Method level variable: -------------------*

local variable

final variable

primitive variable

reference variable

Instance variable: --- variable inside the class is called as instance variable. When you creating object for class then memory will be allocated for all instance variable. After allocating memory for instance variables JVM initialize the variable with default value.

Instance variable scope is within the class where they declare. Sometimes it will access outside but it will need to refer with reference variable of the class.

Local variable: --- The variable which is declared inside the method is called as local variable. Memory will be allocated for local variable when you call the method.

You should initialize the local variable before using otherwise compilation error will came.

(Variable might not have been initialize)

Scope of local variable is within the method where it is declare.

Final variable: ---- variable which are declare with final keyword are called as final variables.

When you declare the final variable then you should assign them values.

Only one value is assigned for final variables, it is fixed and can’t be changed.

Instance variable can be a final variable.

Local variable can be a final variable.

Final variable can be written within the class or within the method.

Static variable: ------- when you create an object

Student s=new s();

Student class will be loaded into main memory.

Int a; primitive variable

Student s reference variable

Variable which are declare inside the class with the static keyword are called as static variables.

Memory will be allocated for static variables when class is loaded into memory.

After allocating memory JVM initialize the static variable with default values.

Scope of static variable is within the class where you defined, sometimes you can use static variable outside the class with object or with class.

Only one copy of memory will be allocated for the static variables for all the object creation.

Memorywill be allocated for local variables after calling the method only

Memory will be allocated for instance variable while you are creating the object.

Difference between instance and static variable: -----------

|  |  |
| --- | --- |
| Instance variable  Variable declare inside the class without a static keyword is called instance variable  Memory will be allocated for the instance variable while you are creating the object.  When you create the multiple objects for the class then the multiple copies is created for instance variable. i.e. 1 copy per an object.  Instance variable can be referred only with an object. | Static variable  Variable declare inside the class with a static keyword is called static variable  Memory will be allocated for the static variable while class is loading into memory  When you create the multiple object for the class then only one copy of static variable will be allocated for all the object . i.e. 1 copy for all objects.  Static variable can be referred only with an object and class name. |

Don’t load two class until the memory loaded for the class1 is empty.

Memory will be allocated for static variable (only one copy) when you create multiple object for a class. Where as in case of instance variables multiple copy will be created for multiple objects.

Static keyword is not allowed for local variable.

Variables are divided into two category: ----------

Primitive variable

Reference variable

|  |  |
| --- | --- |
| Primitive variable  Variable declare with any of the primitive data type are called as primitive variables.  Memory allocation is done for primitive variable based on primitive data type used. Exp:--int ---4 byte double ------8 byte  Initial values of primitive variables depend on data types used. | Reference variable  Variables which are declared with any class type are called as reference variables.  Always 8 bytes of memory will be allocated for reference variable.  Always null will be assign as default for the reference variable |

Similarity :-----------

|  |  |
| --- | --- |
| Primitive variable  Primitive variables can be instance variable, static variable and local variable. | Reference variable  Reference variables can be instance variable, static variable and local variable. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Instance var. | Local variable | Final var. | Static variable |
| Primitive var. | Y | Y | Y | Y |
| Reference var. | Y | Y | Y | Y |

We can write one more type of member in the class called as block

Blocks will be executed automatically whenever you create object.

Blocks: -------- block is set of statements which are enclosed between { }.

We can write the block inside the method also. There blocks are called as local block. Local blocks will be executed whenever you call the methods inside a class.

Creating object = creating instance

There are two type of block: ---------

Instance block

Static block

|  |  |
| --- | --- |
| Instance block  Blocks defined inside a class without static keyword is called as instance block.  Instance block will be executed when you create the object(instance) | Static block  Blocks defined inside a class with static keyword is called as static block.  Static block will be executed whenever class will be loaded into memory. |

When a class will be loaded into memory then following two things will be happens : ------

Memory will be allocated for static variable

Static blocks will be executed.

When you are not writing any constructor in the class then JVM insert one default constructor which is also called as implicit constructor.

When you write any constructor in the class then JVM wouldn’t insert any default constructor.

When you specify a return type as constructor that will be treated as method. it can call ,it will be an object.

Constructor can’t be call with an object.

Methods: -------------Method represents some operation and it is simply a block of statements with the following syntax.

Syntax -------- [modifier] return\_typemethod\_name(parameters)

{

Statements

Return type ----------- //optional

}

Method takes some data as a parameter and does some process and return some value. Some times. Sometimes without taking any data (no parameter) method can do some process and return some data.

Method can return only one value when you are specifying the return type that must be the data type of the value which you are returning.

If you not returning any data type then you should write return type void.

When return type is void then no need to provide the return statement in method.

When return types any other than void then we should provide the return statement in method.

Can I write multiple return statements inside a method?

Yes , we can write multiple return statement in the method conditionally i.e. all the return statements you have written must be reachable .

The above syntax is used to define a method. once method is defined, then we can call method with the following syntax.

[variable ]=reference \_variable. method\_name(parameter)

h.show();

h1.show(10,20);

sum= h2.show(10,20) ;

A method can take multiple parameters.

Method can return only one value.

Parameter type and return value type may be primitive or reference.

Static method: ----- Methods which are declared with static keyword is called as static method.

It can be invoked by the class name directly.

We can also invoke the static method with the object.

NOTE: ---

In the case of instance method or non-static methods we have to use objects to invoke the methods.

Non static variable cannot be referenced from the static method or static block.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Static block | Non static block | Static method Non- Static method |
| Static variable | √ | √ | √ √ |
| Non-static variable | X | √ | X √ |
| Static method | √ | √ | √ √ |
| Non-static method | X | √ | X √ |

Can I declare top level class as final?

Yes

Can I declare top level class as static?

No

Can I declare constructor as final?

No

Can I declare constructor as static?

No

Can I declare class level variable as static?

Yes

Can I declare method level variable as static?

No

Can I declare class level variable as final?

Yes

Can I declare method level variable as final?

No

Can I declare method as static?

Yes

Can I declare method as final?

Yes

Can I declare blocks as static?

Yes

Can I declare blocks as final?

No

Can you write block inside the method?

Yes, these blocks are called as local block. Local block will be executed whenever a method is called.

When you are writing the methods you need to write some parameter and some return type which require. This parameter and return type may be of primitive type or reference type also.

Intm1(inta,int b) ------------- with primitive

Hello m2(hello h) -------------- with reference type

Instance methods: ------------- methods defined without static keyword is called as instance methods.

You can call instance methods with objects.

You can refer all kinds of variables and methods inside the instance methods.

Static methods: ------ methods which are declared with the static keyword is called as static methods.

You can call static methods with the class name & also with the object.

You can refer only static variables and static methods inside the static methods.

Instance variables are not allowed inside the static block and static method directly.

Instance method calling is not allowed inside static block and static method directly.

Instance variable, static variable, instance methods and static methods are allowed inside the instance block and instance method directly.

We can call the instance variables and instance methods with objects because instance member belongs to object.

Static variable and static methods can be called with class name because static members belongs to class static members can also be referred with object.

How to load the class?

Same type of more than one reference variable you cannot use as in the main class

Hello h=new Hello();

Hello h = new Hello();

It gives error as h is already defined in main (java.lang.string [])

But

h.m();

h.m();

No problem java can use where m is a method.

We can not declare any variable as a static inside any block including static block too and any method like.

{

Static int a=3;

}

Or

m()

{

Static int a;

}

Because that variable is nothing but a local variable which can not be declare static.

Hence only instance variable can be declared as static.

Only static block is executed as the class loading time as well as memory is allocated for static variable and its value is assigned to it.

In static block we can also write

Static

{

int m=3;

System.out.println(m);

}

This is executed only one time at class loading time. But local variable can never be declared as static.

Call by value V/S call by reference

When you call a method by passing primitive value as parameters then it is called as call by value.

When you call a method by passing object as parameters then it is called as call by reference.

In the case of call by value , when the method modify the parameter value then those modification will not affected caller of methods where as in case of call by reference modification happens in the method will be affected to caller of the methods.

Constructor

Constructor is a special method whose name is same as class name. Constructors don’t return any value even void also.

Constructor will be invoked by JVM when object is created .constructor will be used to initialize instance variables of the class.

Constructor shouldn’t be called by the developer, they should be call by JVM only when you are creating object for class.

Use of constructor is to initialize the object (instance variable).

Constructor can be parameterized i.e. we can write multiple constructors in a same class in the same name by changing the parameter.

When you creating an object then the corresponding constructor will be invoked based on the parameters only.

When you don’t have any constructors in the class ,JVM inserts one default constructor as follows

Hello ()

{ }

When you have any other constructors in the class JVM don’t insert any default constructor.

We should not specify any return type for the constructor including void also .

When you specify any return type that will be considered as normal method.

Default constructor means the constructor which has zero argument.

Method overloading

In a class you can write more than one method with the same name by changing the parameters as follows: --------- i.e.

No of parameters

Types of parameters

Order of parameters

Method invoking depends on parameter.

When you overriding methods, return type is not a matter i.e. you can use any return type which will not be used in method invocation.

Constructor overloading: -----writing multiple constructors in a class by changing the parameter is called constructor overloading.

Return type: ------------no need.

This keyword: -------------- ‘this’ is reference variables which contain the object of current class.

Sometimes you can take the same name for local and instance variable. Then inside the method local variables hides the instance variables. To differentiate local & instance variable, we use this to refer the instance variables. ‘this’ is the instance reference variable which is not allowed from static block and static methods.

When this reference variable is created by JVM? ---------------Class loading time.

What is class ?

Class is a user defined data type which contains logical representation of operation and properties.

What is an object ?

Object is a physical existence of properties (data).

Class loading by class loaders: ---------

Class loaders are part of JVM whose responsibility is loading the classes into main memory when you are trying to use a class then class will be loaded into memory.

The following four scenarios where we can use the class: ---------------

When you are running the java class from class prompts.

Exp:- java test9; // then test9 will be loaded.

When you are trying to create object for the class then that class will be loaded.

Exp:--------- Hai raj= new Hai();

When you are invoking a static method with the reference variable which contains null then the class of that references variable will be loaded.

Exp:--------

Hello.h=null;

h.m();

A.m1();

When you are invoking a static method in the class name then the class will be loading first.

Exp:---------

A.m1();

In java we can only assign null address to other reference variable , other are not allowed.

NOTE: -------- when you are invoking a static method with a reference variable which is pointing some project then class will not be loaded into the memory at that point because class is already loaded before to this when you are creating the object.

Hello h=new Hello(); // loads the class

h.m1(); //wouldn’t load the class

When you are invoking a non-static method with a reference variable which contains null then null pointer exception will happen at run time.

When you are invoking a static method with the reference variable which contains null then method will be invoked successfully without error.

Object creation time: ------------

Class will be loaded.

Memory will be allocated for the reference variable.

Memory will be allocated for instance variable.

Instance block will be executed.

Constructor will be invoked.

Assign address to object into reference variable.

Class loading time: -------------

Create the default object

Creating the stack.

Static variable memory allocation

Static block execution

Access modifiers

There are four access modifiers in java: ------------------

Private

(Default)

Protected

Public

Access modifier will be used to specify the scope of class, methods and variables.

Private member are allows in the class where they are defined.

Private member is not allowed in other classes which are available in same package.

Default members, protected members and public members are allowed within the class and also in other classes which are available in the same package.

When you are not specifying any package then classes will be stored in default package.

Top level classes can’t be private and protected and can be default and public.

All the class level variables can be private, protected, public, default(static variable and instance variable both)

Local variable cannot be any thing such as public, private or default.

Method within the class can be private, public, protected or default.

Constructors can be private, public, protected and default.

Blocks cannot be private, public, protected, final, default.

When you declare a top level class as public, you must save that in a separate source file with the same file name.

Exp: ---- when you declare Hello class as public you must save that source file with Hello. java

How many public classes are allowed in one source file?---------------------Only one

Why top level classes can’t be private and public?

How many class file can I write inside a source file?

Multiple classes we can write inside a source file. Public class only once allowed and default classes are allowed many times.

INHERITANCE

It is a process that extending the existing class functionality with the new class. When new class extending with existing class then all the members of existing class will be used in the new class directly.

When we extends one class for other class then all super class members becomes member of subclass.

With the super class object we can access only super class members.

With sub class object we can access both super and subclass members.

Types of inheritance: --------------------

Simple Inheritance

Multilevel Inheritance

Hierarchical Inheritance

Multiple Inheritance

Hybrid Inheritance

Simple Inheritance: --- in simple inheritance there will be exactly one super class and one subclass i.e. subclass can get the functionality from exactly one super class.

Multilevel Inheritance: ------- In Multilevel Inheritance one super class can have many subclasses, in that one subclass is directly and all remaining are indirect.

Like that one subclass can have many super classes & one is direct super class and all remaining are indirect super class.

Hierarchical Inheritance: -- In this one super class have many direct subclasses.

Multiple Inheritances: ----In this one subclass can have multiple direct super classes.

Multiple Inheritances is not allowed in java, using classes (due to various region)

Hybrid Inheritance: ----- Hybrid Inheritance is a combination of multiple inheritance, multilevel inheritance and hierarchical inheritance.

When you write constructor inside the subclass then JVM inserts one default super as a first statement. This default super invokes the immediate super class default constructor.

If you want to invoke super class argument constructors then you need to write super with parameters.

Super()---------invokes super class default constructor .

Super(a,b)----- invokes super class 2 argument constructor

Super class be the 1st statement inside the constructor.

Only one super class statement is allowed inside the constructor.

When you write the super class statement inside the constructor then JVM wouldn’t insert the default super.

Super is also a reference variable like “this”.

‘This’ reference variable points object of current class whereas ‘super’ reference variable points object of immediate super class.

With ‘this’ reference variable we can refer current class variable and methods where as with ‘super’ reference variable, we can refer immediate super class variable and methods

Using this () you can invoke current class constructor. But using super () you can invoke immediate constructor.

Can I have same name for super class variable, subclass variable and local variable?

Yes ,you can have , but subclass member hides the super class members further you have to differentiate the subclass and super class members refer the subclass member with ‘this’ and super class member with ‘super’.

METHOD OVERRIDING

Method overriding: ------- Implementing the super class method in subclass with the same name and with the same signature. This is called as method overriding.

Rules for method overriding: --------

Subclass method name must be same as super class method name.

Subclass parameters must be same as super class parameter.

Subclass method return type must be same as super class method return type.

Subclass method access specifier must be same as super class method access specifier or higher than super class method specifier

|  |  |
| --- | --- |
| Super class  Private  Default  Protected  Public | Subclass  Private, default ,protected ,public  Default ,protected ,public  Protected ,public  public |

When super class method is throwing an exception in subclass, then you do one of following: --

You can omit the method level exception in subclass

You can use same method level exception in subclass

Sub class methods can throw subclasses to super class exception.

Subclass method can’t throw super class to super class exception class.

Assume that super class has a method of follows: ---------

Void m1(int x)throws B

You can override this method with one of the following form: ----

Void m1(int x)[ protected void m1(int a)throws B]

Void m1(int x)[ public void m1(int a)throws B]

Dynamic dispatch

Assigning subclass object to super class reference variable dynamically at run time is called as dynamic dispatch.

We can not assign super class object to sub class reference variable.

When A is supper class and B is subclass which of the following statement are --------- T/F

A obj = new A(); ---------------- T

B obj = new B(); ---------------- T

A obj = new B(); ---------------- T

B obj = new A(); ---------------- F

Can I override private method? -------------------------- No

If I want to achieve run time polymorphism what are required?

Dynamic dispatch and method overriding

Polymorphism

One form behaving differently in different situation is called polymorphism.

In java we have two types of polymorphism: ---

Static/compile time

Dynamic/run time

You can implement compile time by using method overloading

You can implement run time by using method overriding and dynamic dispatch

When you have static and private method in super class, we can’t achieve the run time polymorphism.

When super class method is static we have to override that method as static only in the subclass.

What is the use of final keyword?

We can use final keyword for variables, methods and classes.

Final variable can’t be modified.

Final variable can’t be overridden.

Final variable can’t be inherited.

Abstract class

When you writing the classes, sometime you are unable to implement some of the methods, i.e. you are unable to provide the query for that because implementation details are unclear.

When you not providing query for the methods, you must declare method as abstract..

When a class has one and more abstract method then you should declare the class has abstract.

When a class is abstract then it can’t be initiated , but you can declare the reference variable.

When one class is extending another class, then subclass has to override all the abstract method of super class, otherwise subclass must be declared as abstract.

Abstract class can contain both abstract method and concrete method.

Abstract class can also contain only abstract method.

Abstract class can also contain only concrete methods

Abstract class can’t be initiated i.e. we can’t make the object of abstract class.

When you are not providing body for the method then we should declare that method as abstract.

When a class is abstract, you can’t create object for that class.

Abstract class must have subclass.

Can I declare a class as abstract without abstract method?

Yes, we can have abstract class without abstract method.

Can I declare instance variable and static variables inside the abstract class?

Yes, you can declare.

Can I write constructor inside abstract class?

Yes

Can I have static and non-static block inside the abstract class?

Yes

Can I override static methods?

No

Can I override private methods?

No, private method can’t override.

If I want to stop method overriding what I have to do?

Declare the method as final.

How can I stop the inheritance?

By declare the class as final.

Always you have to call instance method with object only. But you can call static method with class name or with reference variables which contains null or with an object why it is?

Instance method may use instance variable and concrete object only. So without creating object you cannot access instance method.

Static methods wouldn’t use any instance variables so, object is not require to invoke static methods.

Why multiple inheritances are allows with java interfaces and why it is not allowed with java classes?

Difference between abstract class and interface: ---------

|  |  |
| --- | --- |
| Abstract class  Abstract class can have instance variable, static variable, constructor, abstract methods, concrete methods, instance blocks and static blocks.  Subclass has to extend abstract class using extend keyword.  We can declare the variable of abstract class as private, default, public, protected.  We can declare the abstract class variable as final and non-final variable  We can declare the abstract class variable as static and non-static  We can declare the abstract class methods as protected, default, public.  Abstract class doesn’t allow multiple inheritances. | Interface  Interface can have only static variable and abstract methods.  Subclass has to implement interface using implement keyword.  Always interface variable are public  Always interface variable are final.  Always interface variable are static.  Always interface methods are public  Interface allow multiple inheritance. |

Similarity between abstract class and interface: ----------

Both abstract class and interface cannot be initiated.

We can declare the reference variable for both abstract class interface class.

When you extend abstract class or when you implement interface then you have to override all the abstract method in subclass, otherwise you have to declare subclass as abstract.

When I want to use abstract class and interface?

Selection of abstract class and interface depends on your requirement. Exp: -- you can write abstract method and concrete method then you have to select abstract class. When you want to write only abstract methods then you can select abstract class and interface. If you want to achieve multiple inheritances, then you have to select interface only.

Which of the following keyword are allowed for interface?

Private------------------- X

Protected ----------------- X

Default ------------------- Y

Public ---------------------- Y

Static ----------------------- X

Final ------------------------ X

Abstract -------------------- X

For concrete class: ----------- default, public, final, abstract.

For abstract class: ----------- default, public

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Class level var. | Method level var. | constructor | Class level block | Method level block | methods | class | Interface |
| Private | √ | X | √ | X | X | √ | X | X |
| (Default) | √ | X | √ | X | X | √ | √ | √ |
| Protected | √ | X | √ | X | X | √ | X | X |
| Public | √ | X | √ | X | X | √ | √ | √ |
| Static | √ | X | X | √ | X | √ | X | X |
| Final | √ | √ | X | X | X | √ | √ | X |
| Abstract | X | X | X | X | X | √ | √ | X |

Private, public, protected, static, final, abstract -------------

Which of the above keyword is allowed for instance method: ---------private, protected, public, final, abstract

Which of the above keyword is allowed for a static method: ------ except abstract all are allowed

Which of the above keyword is allowed for abstract methods: --------- abstract, static

Interface

Interface is a special JAVA class which is fully abstracted, which contains only two type of members: ---

Final static variable

Public abstract variable

Syntax: -- interface interface\_name

{

Data type var1,var2,-----;

Final static data type var3;

Public abstract return type method \_name(parameter);

Return type method\_name(parameter);

}

Exp:---

Interface i1

{

int a=10;

int b=12;

Void m1();

}

Interface is a complete abstract class which contains public, final, static variable and public abstract methods.

Interface cannot be instantiated but we can declare the reference variable when a class is implementing interface then that class must override all the abstract methods of interface otherwise class must be abstract.

When a class is implementing interface then all final static variable of interface will be inherited to subclass.

All the variables declared inside the interface are final and static.(we can also access the static variable with class name)

All the methods declared inside the interface are public and abstract.

Interface can’t be initiated i.e. Cannot create an object. But we can create reference variable.

Interface must be a subclass.

Can I declare interface as static?

No

Can I declare interface as final?

No

Can I declare top level interface as private?

No

Can I declare abstract method as static?

No

Can I declare abstract method as final?

No

Can I declare abstract class as final?

No

Can I write constructor inside the interface?

No

Can I write static and non-static block inside the interface?

No

Can I write concrete method inside the interface?

No

How can I achieve multiple inheritances in java?

By interface only

Why multiple inheritances are allowed through interface only? Why not through class?

Refer diagram.

Can I write constructor inside the interface?

Constructor is not allowed inside interface

Can I write static block inside the interface?

No

Can I write instance block inside the interface?

No

Can I write instance variable, concrete methods inside the interface?

Yes

Can I write static method inside the interface?

No

Package

Package is a collection of .class file.

Package is a folder or set of folders in the three formats: --------

Package will have view to organize the files of your project in a better way.

Package will have view to solve the naming complex.

When you are writing java classes and compile, by default all the classes will be placed in default package (package without name).

If you want to place your classes in some package with some name, we have to use the package statement in the source file.

(x) (x)-------> (y) (x)---------->(y)------ ->(z)

Package x; package x.y; package x.y.z;

Declaration of package:-------

Syntax: ---------- package package\_name;

Package x;

Package x.y;

Package x.y.z;

In the above example x is a folder inside that one more folder y. inside y another folder called z .here separate the folder name with dot (.).

Package declaration statement must be the 1st statement of the java source file.

Two package declaration statements is not allowed inside one source file.

Private member are allowed within the class and not allowed outside the class.

Public member are allowed any where.

Default member are allowed within the class, subclass &non\_subclass which are available in same package.

Default member are not allowed out side the package.

Protected member are allowed within the class, subclass & non subclass available in same package and subclass available in different package. Protected members are not allowed in non subclass which is available in different package.

|  |  |  |
| --- | --- | --- |
| Class | Extends | Class |
| Class | Implement | Interface |
| Interface | Extends | Interface |
| Interface | No operation | Class |

Check T/F: ---------------

Java is pure OOPL: ------------------------------------- F

Java supports pointer: ---------------------------------- F

Java support operator overloading: ------------------ F

Const and goto keyword is allowed in java: --------- F

Method overloading is not allowed in java: ---------- F

Multiple inheritances is allowed in java: -------------- T

Method inside method is not allowed: --------------- T

Constructor can not be inherited: ----------------------T

Private method can’t be inherited: -------------------- T

Private method can not be overridden: ----------------T

Static method can not be overridden: -------------------T

Static method can not be inherited : --------------------F

Protected methods can’t be inherited: --------------------F

Protected methods can’t be overridden: ----------------- F

Block can’t be inherited: ------------------------------------ T