

What is the Java?

It is an object oriented programming language similar to C++. This is the open source and can run on all platforms. It is concurrent language you can execute many statements instead of sequential executions also it is class based and object oriented programming language. Java allow us write once run anywhere i.e compiled code can execute on any operating system.

Why use the java or feature of java

Portable: This is the platform independent language which application written in one platform and can easily portable to another platform.

Object oriented: everything is considered to be an object. Means in java we cannot create any program without inheritance and without class so we can say it is pure object oriented.

Secured: This code is converted in byte code after compilation which is not readable by a human and java does not use the explicit pointer so user cannot access the memory space and its address directly so it is secured language

Dynamic: Java allows us to allocate memory dynamically at run time to which memory wastage is reduced and improve application performance.

Robust: java has strong memory management system. It help in eliminating error and it check code during compile and runtime.

Multithreaded: java support multiple threads for execution, including a set of synchronization primitives

Distributed: this language provide a feature which help to create distributed application using RMI (Remote method invocation) a program can invoke method of another program across network and get the output.

If we want to work with java we have to know how to create program in it and execute it.

If we want to create the java program then we have the some following important steps.

1) Installed the JDK: before that we have to know what the JDK is

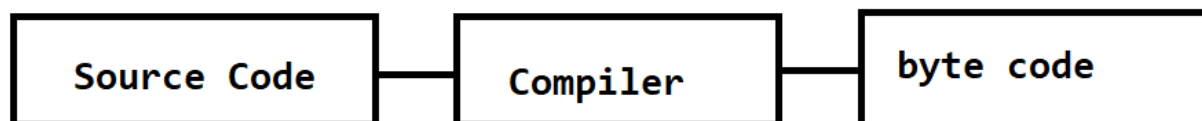
JDK stands java development kit

Definition: It is a software development environment that offers collection of tools and libraries necessary for java application development. JDK contain compiler, JVM, Applet viewer etc

What is the compiler?

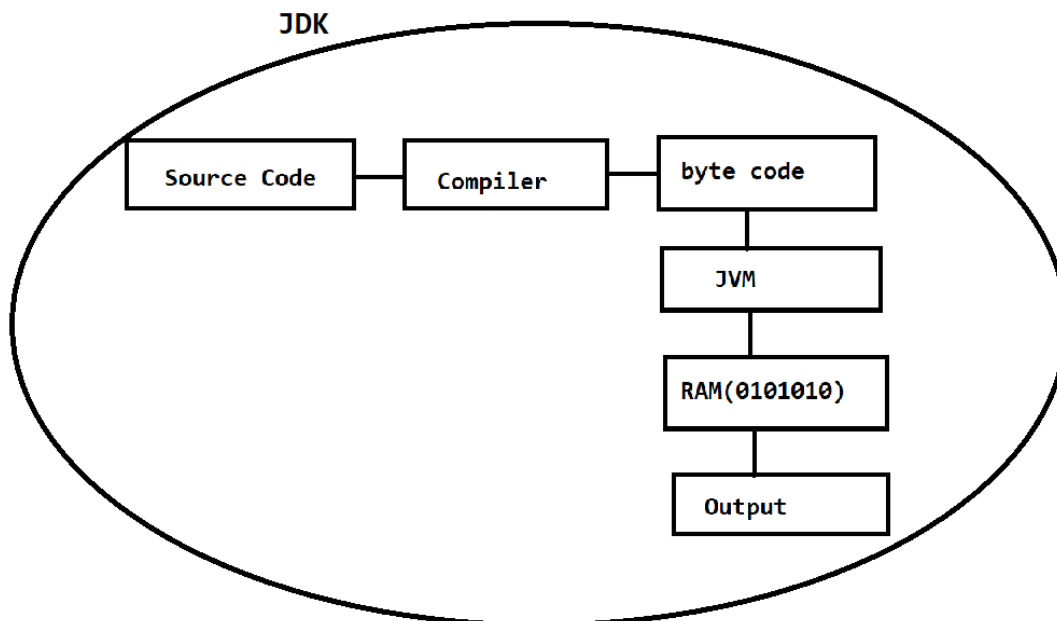
Compiler is program or application software which is used for convert the one programming language to another programming. In java case of java it is used for convert the source code to byte code

Working of compiler



What is the byte code?

Byte code is machine understandable code or it is format which easily converts in machine code with the help of JVM.



What is the JVM?

JVM stands for Java Virtual Machine it is specification that provide runtime environment in which byte code can be executed and translate in machine code

What are the roles of JVM or what it does?

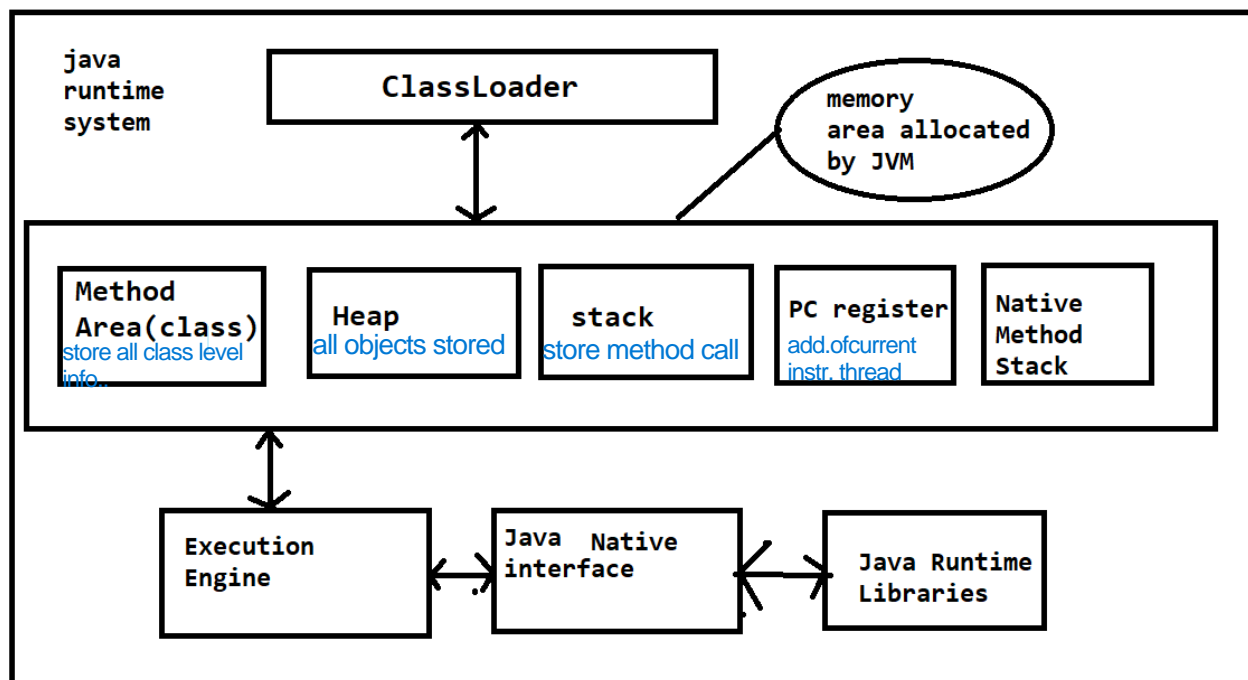
1) Load the byte code 2) verify the byte code 3) execute the byte code 4) provide the runtime environment for execution of code.

JVM provide the definition for the

1) Memory area 2) class file format 3) register set 4) garbage collector 5) fatal error reporting file

Explain the JVM Architecture

Following diagram shows the JVM Architecture



Class Loader: class loader is subsystem of JVM which load the class file whenever we run the java program as well as verify the byte code. There are major three types of class loader

1) Bootstrap Class Loader: this is the first class loader which is super of extension class loader. It loads the rt.jar which contains all class files in java standard edition like as java.lang package, java.net package, java.util package.

2) Extension Class Loader: this is the child of Bootstrap Class Loader which is responsible for load the external library and its .class files in your application

3) System/Application class loader: it is used for load the class files from class path by default class path to current directory means jdk\bin folder

Method Area or Class Area: In the method area all class level information stored like as class name, immediate parent class name, methods of class, and variables of class including static variables. The only one method for per JVM and it is also shared resource.

Heap Section: information of all objects is stored in the heap area. There is also one heap area for per JVM and it is shared resource.

Stack Area: for every thread JVM create runtime stack which is stored here every block of this stack is stack frame or active record which store the method call. All local variables of this method are stored corresponding stack frame after thread termination stack frame will be destroyed by JVM it is not shared resource.

PC Register: It store the address of current instruction of thread and every thread having separate pc register

Native Method Stack: for every thread a separate native stack is created it store native method information

Execution Engine: Execution engine execute the .class file (byte code). It read the byte code line by line uses data and information in various memory areas and execute instructions

Execution engine divide in three parts

a) Interpreter: read the byte code line by line and then execute it. The disadvantage is that when one method is called multiple time every time interpreter get executed.

b) JIT (Just in Time): it is used for increase the efficiency of interpreter. It compile the entire byte code and change it to native code so whenever we call method multiple time no need to interpretation multiple time.

c) Garbage collector: it is used for destroy the unused objects.

JNI (Java Native Interface): it is an interface with the native method library and provides the native libraries (C, C++) required for execution and it enables JVM to call C/C++ libraries.

Why java developed the byte code

Byte code is platform independent code means we can execute the byte code on any operating system so java is platform independent language and it is very strong feature of java

2) Create the sample application:

If we want to create the any program in java we must be write minimum single class. Means we cannot run the java application without class

Generalize format of java application

```
access specifier class classname
{ public static void main(String x[])
{
    write here your logics
}
}
```

Example:

```
public class First
{
    public static void main(String x[])
    {
        System.out.println("good morning");
    }
}
```

Description of above code

public class First: public is access specifier and class is keyword for class declaration and First is class name and user can give any name to his class.

public static void main(String x[]): this is the main method of java same like as main method in c or c++.

System.out.println(): this is the output statement of java it is used for display the output on output screen like as printf()

Meaning: System is class out is static reference of PrintStream class System and PrintStream maintain the HAS-A relationship between them println () is overloaded method for display the output on output screen.

Note: we can create the java program by using notepad, eclipse, spring tool suite or IntelliJ etc

But here we use the notepad.

Example:

```
public class TenAug
{ public static void main(String x[])
  { System.out.println("Good Evening");
  }
}
```

3) Save the Application: if we want to save the application of java then save in bin folder where jdk installed and give class name and file name same with .java extension as per our example we have to give the file name as TenAug.java

4) Compile the Application: if we want to compile the application of java we have the some following steps.

a) Open the command prompt

Start menu → search → command prompt

b) Go where java file save: copy the path where java file save and paste on command prompt using the following command shown in screen short.

```
C:\Users\Admin>cd C:\Program Files\Java\jdk1.8.0_291\bin
C:\Program Files\Java\jdk1.8.0_291\bin>_
```

c) type the command javac filename.java shown in following screen short.

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac TenAug.java
C:\Program Files\Java\jdk1.8.0_291\bin>
```

Note: when we compile the file then java compiler create the new file automatically with extension of .class file and in this file contain your byte code.

```
C:\Program Files\Java\jdk1.8.0_291\bin>dir TenAug.*
Volume in drive C has no label.
Volume Serial Number is D892-2B3C

Directory of C:\Program Files\Java\jdk1.8.0_291\bin

20-10-2021  18:15                418 TenAug.class
20-10-2021  18:12                121 TenAug.java
                2 File(s)              539 bytes
                0 Dir(s)  80,080,384,000 bytes free
```

byte code (pointing to TenAug.class)

source code (pointing to TenAug.java)

5) Run the application: if we want to run the java application then we have the command name as java filename

E.g java TenAug

Output shown in following screen short

```
C:\Program Files\Java\jdk1.8.0_291\bin>java TenAug
Good Evening

C:\Program Files\Java\jdk1.8.0_291\bin>
```

Q. Why main method is static?

Because Java main method present in class and java main method call by using JVM and JVM need to **call** the java **main method using class name** so it is static. Suppose java **not** declare main method **as static** then JVM need to create the **object of the class** in which main method exist but there is **problem** in every **class contain** by **default constructor** called as implicit constructor **but** if user declare the parameterized constructor in its class and when JVM create the object of the class then JVM need to pass parameter to constructor but JVM cannot pass the

parameter to user constructor so java avoid to create the object of class in which main method is present by JVM so it is static because static method can call by using class name (refer java interview video series)

Q. can we overload the main method in java?

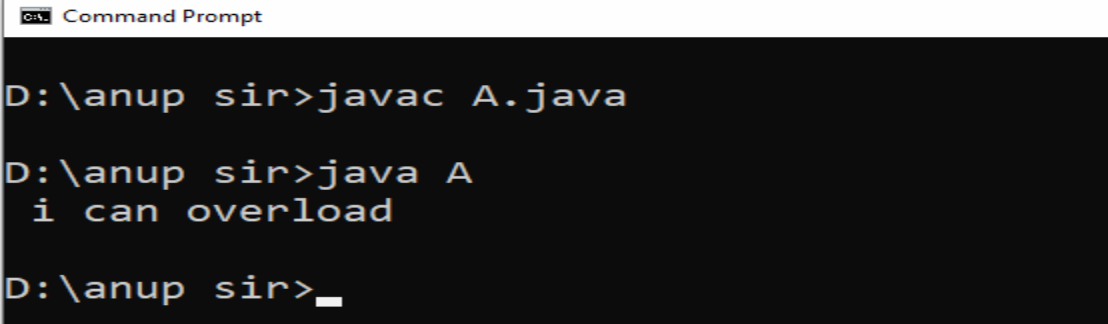
Yes we can overload the main method in java when we overload the main method in java then program compile get but not execute if we not overload the method public static void main(String x[]) because program execution done by using JVM and JVM knows the only one signature of main method called as public static void main(String args[])

```
class A
{
    public static void main(int x[])
    {
    }
}
```

If we write the main method mention **as** above then program get compile but not execute because compiler think user may be overload the main method but when we execute the java program then JVM is enable and JVM knows only one signature of main method called as public static void main(String x[])

So if we want to execute the java program property then your code should be

```
class A
{
    public static void main(int x[])
    {
    }
    public static void main(String x[])
    {
        System.out.println(" i can overload");
    }
}
```



```
D:\anup sir>javac A.java

D:\anup sir>java A
i can overload


D:\anup sir>_
```

What is the meaning of string x[] present in main method ?

String x[] indicate the command line argument in java

Command line argument means parameter pass in main method of string array called as command line arguments.

```
class A
{
    public static void main(String x[])
    {
    }
}
```



command line arguments

Why use the command line arguments or what is the benefit of command line arguments?

It is used for accept the input at program run time and it is infinite string array present in main method means we can accept the n number of input through the command line argument but the first input of command line is at position of zero because it is array.

Now we want to create application to accept the two numbers from keyboard and calculate its addition

```
public class A
{
    public static void main(String x[])
    {
        int a,b,c;
        a=x[0]; //accept first input
        b=x[1]; // accept second input
        c=a+b;
        System.out.printf("Addition is %d\n",c);
    }
}
```

If we think about the above code then we have two compile time errors

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac A.java
A.java:6: error: incompatible types: String cannot be converted to int
        a=x[0];
        ^
A.java:7: error: incompatible types: String cannot be converted to int
        b=x[1];
        ^
2 errors
C:\Program Files\Java\jdk1.8.0_291\bin>
```

Why these errors occur?

Because we have the string value for input and we want to accept the integer value as input and string cannot store in integer so compiler generate the error to us in incompatible types.

```
public class A
{
    public static void main(String x[])
    {
        int a,b,c;
        a=x[0];
        b=x[1];
        c=a+b;
        System.out.printf("Addition is %d\n",c);
    }
}
```

x[0] is type of string and try to store a variable and a variable is type of integer so we have the error incompatible

If we want to solve this error in java we have to use the type casting technique

What is the type casting?

Convert one type value to another type called as type casting.

Means as per our example we have to convert the string to integer

How to convert String to integer in java or String to other type

If we want to convert the string to integer in java we have the following statement.

Syntax:

int variable = Integer.parseInt(String);

For float conversion we have to write like as

Float variable=Float.parseFloat(String);

For double conversion

Double variable=Double.parseDouble (String);

Etc

So if we correct the above code for resolve the above error then your code should be.

```
public class A
{
    public static void main(String x[])
    {
        int a,b,c;
        a=Integer.parseInt(x[0]);
        b=Integer.parseInt(x[1]);
        c=a+b;
        System.out.printf("Addition is %d\n",c);
    }
}
```

we solve error
successfully

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac A.java
C:\Program Files\Java\jdk1.8.0_291\bin>
```

When we try to run this program then we get the error at program run time shown in following screen short.

```
C:\Program Files\Java\jdk1.8.0_291\bin>javac A.java

C:\Program Files\Java\jdk1.8.0_291\bin>java A
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 0
    at A.main(A.java:6)
```

We get error ArrayIndexOutOfBoundsException

Because when we run the program and if we use the command line argument then we have to pass input on same line where we run the program

Following diagram shows the correct way of input in the command line argument

```
public class A
{
    public static void main(String x[])
    {
        int a,b,c;
        a=Integer.parseInt(x[0]);
        b=Integer.parseInt(x[1]);
        c=a+b;
        System.out.printf("Addition is %d\n",c);
    }
}
```

The screenshot shows three runs of the program in a command prompt:

- Run 1: `javac A.java` (successful compilation).
- Run 2: `java A` with empty input fields. The output is: `Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 0 at A.main(A.java:6)`. A red arrow points from `x[0]` in the code to the error message.
- Run 3: `java A 100 200` with input fields containing 100 and 200. The output is: `Addition is 300`. A red arrow points from `x[1]` in the code to the input field containing 200.
- Run 4: `java A 100` with input fields containing 100 and an empty field. The output is: `Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 1 at A.main(A.java:7)`. A red arrow points from `x[1]` in the code to the error message.

Not found input for
x[0] so JVM
raise error
ArrayIndexOutOfBoundsException
0

proper input
as per program

not found
input for x[1]
so JVM
raise error
ArrayIndexOutOfBoundsException
1

What is the JRE?

JRE Stands for Java Runtime Environment it is software package which bundle the libraries (jar) and the java virtual machine and other components to run application written in the java. JRE is physical existence of JVM and JVM is logical entity or abstract machine which is present under the JRE. If we want to execute any java program we have to install the JRE.

JRE contain the following bundles or packages

1. DLL 2. Java extension files 3. Font files 4. Core libraries

What is the diff between JDK JRE and JVM?

JDK is software development kit whereas JRE is software bundle that allow java program to run where as JVM is an environment for executing byte code.

JDK contain tools for developing, debugging etc JRE contain class libraries and other supporting files whereas software development tool are not included in JVM and JVM work as tool in JRE

JDK comes with the installer on the other hand JRE contain environment to execute the code where JVM is tool present in JDK and JRE

JDK contain compiler, applet viewer etc tools

Why use JDK

JDK contain tools required to write Java Programs and JRE to execute them

JDK include compiler, Java Application launcher and Applet viewer

Means we can use the JDK for develop the java source debug it and convert it in byte code means for compilation purpose.

Why Use JRE?

It is used for import the packages in application like as java.util,java.math and run time libraries means JRE specially design for execute and test the java application

Why Use JVM

JVM is platform for executing java source code internally provide the memory for java objects and release the memory of objects and JVM comes with JIT which help to minimize the code interpretation and provide the clean code for converting in machine code.

Is JVM is platform dependent or independent

JVM is platform dependent means every operating system having different JVM as per the operating system.

Why JVM is platform dependent

Because JVM separate for every operating system and every operating system having different machine code generation algorithms and technique so Java design the JVM as per the operating system machine code generation technique so JVM is platform dependent

How to Accept the Input on new line using Java

If we want to accept the input on new line using java we have the Scanner class.

Steps to works with Scanner class

1) Add the java.util package in application

What is the package?

Package is collection of classes and interfaces in java it is like as header file in c
If we want to add the package in program then we have the import keyword
It is like as #include in c

If we want to add the package in java we have the four ways

- a) Wild card import
- b) Single type import
- c) Inline package import
- d) Static package import

Note: inline and static package we will discuss in package chapter in detail.

Wild card import: this is the package import where we can import the all member from package in application it is denoted by using *

Syntax: import packagename.*;

e.g import java.util.*; means we can import or add all member from java.util package in application.

Single type import: single type import means we can add the specific member from package in application.

Syntax: import packagename.classname;

Example: import java.util. Scanner;

2) Create the object of Scanner class: so if we want to use the Scanner class we have to create its object

Syntax: Scanner ref = new Scanner (System. in);

3) Use its methods or function for accept the input: Scanner class provides the some inbuilt method to us for accept the input

int nextInt(): this method is used for accept the input of type integer

float nextFloat(): this method is used for accept the input of type float

double nextDouble(): this method is used for accept the input of type double.

long nextLong(): this method is used for accept the input of type long

String nextLine(): this method is used for accept the input of type string.

etc

Example

We want to accept the name id and per of student and display using Scanner

```
import java.util.*; //step1
public class StudentA
{   public static void main(String x[])
    {   Scanner xyz = new Scanner(System.in); //step2
        String name;
        int id;
        float per;
        System.out.println("Enter the name id and percentage of student");
        name=xyz.nextLine(); //step3
        id =xyz.nextInt();
        per=xyz.nextFloat();
```

```
System.out.printf("Name is %s\n",name);  
System.out.printf("Id is %d\n",id);  
System.out.printf("Percentage is %f\n",per);  
}  
}
```

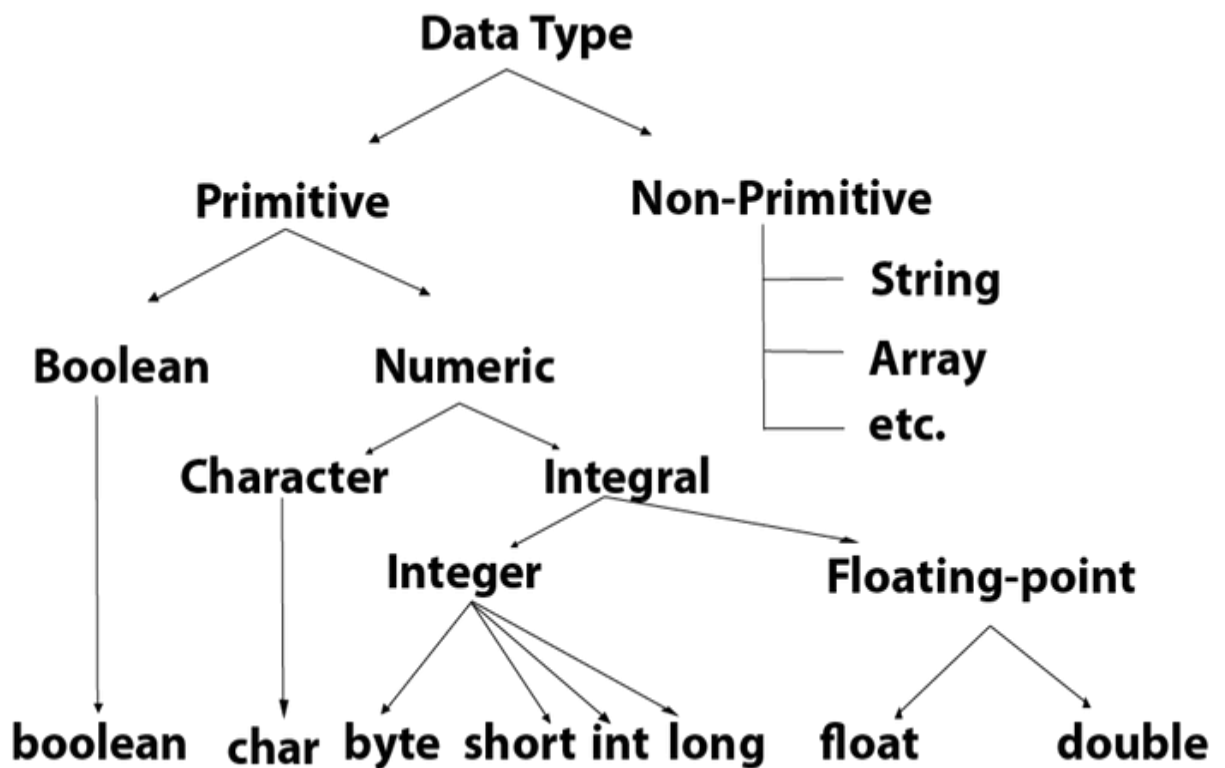
Data types in java

1. **Primitive data types:** The primitive data types include boolean, char, byte, short, int, long, float and double.
2. **Non-primitive data types:** The non-primitive data types include Classes , Interfaces , and Arrays

Primitive Data Type

In Java language, primitive data types are the building blocks of data manipulation. These are the most basic data types available in Java language.

- 1) boolean data type
- 2) byte data type
- 3) char data type
- 4) short data type
- 5) int data type
- 6) long data type
- 7) float data type
- 8) double data type



Data Type	Default Value	Default size
boolean	false	1 bit
char	'\u0000'	2 byte
byte	0	1 byte
short	0	2 byte
int	0	4 byte
long	0L	8 byte
float	0.0f	4 byte
double	0.0d	8 byte

Boolean Data Type

The Boolean data type is used to store only two possible values: true and false. This data type is used for simple flags that track true/false conditions.

The Boolean data type specifies one bit of information, but its "size" can't be defined precisely.

Example: **boolean one = false**

Byte Data Type

The byte data type is an example of primitive data type. It is an 8-bit signed two's complement integer. Its value-range lies between -128 to 127 (inclusive). Its minimum value is -128 and maximum value is 127. Its default value is 0. The byte data type is used to save memory in large arrays where the memory savings is most required. It saves space because a byte is 4 times smaller than an integer. It can also be used in place of "int" data type.

byte a = 10, byte b = -20

Short Data Type

The short data type is a 16-bit signed two's complement integer. Its value-range lies between -32,768 to 32,767 (inclusive). Its minimum value is -32,768 and maximum value is 32,767. Its default value is 0.

The short data type can also be used to save memory just like byte data type. A short data type is 2 times smaller than an integer.

Example:

short s = 10000, short r = -5000

Int Data Type

The int data type is a 32-bit signed two's complement integer. Its value-range lies between - 2,147,483,648 (-2^{31}) to 2,147,483,647 ($2^{31} - 1$) (inclusive). Its minimum value is - 2,147,483,648 and maximum value is 2,147,483,647. Its default value is 0.

The int data type is generally used as a default data type for integral values unless if there is no problem about memory.

Example:

int a = 100000, **int** b = -200000

Long Data Type

The long data type is a 64-bit two's complement integer. Its value-range lies between -9,223,372,036,854,775,808 (-2^{63}) to 9,223,372,036,854,775,807 ($2^{63} - 1$) (inclusive). Its minimum value is - 9,223,372,036,854,775,808 and maximum value is 9,223,372,036,854,775,807. Its default value is 0. The long data type is used when you need a range of values more than those provided by int.

Example:

long a = 100000L, **long** b = -200000L

Float Data Type

The float data type is a single-precision 32-bit IEEE 754 floating point. Its value range is unlimited. It is recommended to use a float (instead of double) if you need to save memory in large arrays of floating point numbers. The float data type should never be used for precise values, such as currency. Its default value is 0.0F.

Example:

float f1 = 234.5f

Double Data Type

The double data type is a double-precision 64-bit IEEE 754 floating point. Its value range is unlimited. The double data type is generally used for decimal values just like float. The double data type also should never be used for precise values, such as currency. Its default value is 0.0d.

Example: double d1 = 12.3

Char Data Type

The char data type is a single 16-bit Unicode character. Its value-range lies between '\u0000' (or 0) to '\uffff' (or 65,535 inclusive).The char data type is used to store characters.

Example: char letterA = 'A'

Operator Precedence Chart in Java

Operator	Precedence
Bracket	()
Prefix increment ,decrement and unary	++ -- + - ~ !
Multiplicative	/ * %
Additive	+ -
Postfix increment and decrement	++ --
Shift	<< >> >>>
Relational	< > <= >= instance of
Equality	== !=
Bitwise AND	&
Bitwise exclusive OR	^
Bitwise inclusive OR	
Bitwise AND	&&
Bitwise OR	
Ternary	? :
Assignment	= += -= *= /= %= &= ^=

MCQ Question

Question1 :

```
public class P {  
    public static void main(String[] args) {  
  
        int a = 10, b = 5, c = 1, result;  
        result = a-++c-++b;  
        //    result = a-(2)-(6)  
        //    result = 10-2=8  
        //    result = 8-6  
        //    result = 2  
        System.out.println (result);  
    }  
}
```

Output: 2

Description of above code:

If we think about the above code and think about the priority of operator then ++ operator having higher priority than – (binary minus) operator so ++ operator get executed before binary minus (-) so your expression look like as after execution of ++ result=a-(++c)-(++b) means result=10-(2)-(6) after solve ++ operator we have the two binary minus operator present in equation so from left to right operator get executed means your equation get executed like as result=10-2-6 solve first result=8-6 solve later so final result is result=2.

Question2:

```
public class Second  
{    public static void main(String x1[]) {  
        int x = 0, y = 0 , z = 0 ;  
        x = (++x + y-- ) * z++;  
        System.out.println("X is "+x);  
    }  
}
```

}

If we think about above code then its execution like as

Initially x=0,y=0;

x = (++x + y--) * z++;

++x means first it gets incremented later gets initialised //++x=1

y-- means it gets initialised first later gets decremented //y-- =-1

z++ means first it gets incremented later gets initialised //z++ =1

x=(1+ -1) * 1

x=0

Question 3:

```
class Numbers{  
    public static void main(String args[]){  
        int a=20, b=10;  
        if((a < b) && (b++ < 25)){  
            System.out.println("This is any language logic");  
        }  
        System.out.println(b);  
    }  
}
```

Description of above code

If we think about the code initially value of a=10 and b=20 and when if statement get executed then (a<b) get executed the value of a is 20 and the value of b is 10 so the first condition get failed and we use the && operator in if condition so the rule of && is when first condition get failed then second condition not check by && so b++ not executed so the value of b is 10

Question 4

```
class MyClass  
{
```



```
public static void main(String s[])
{
    int i = 4;
    int j = 21;
    int k = ++i * 7 + 2 - j--;
    System.out.println("k = " + k + "\t J=" + j);
}
}
```

Output: K= 16 J=20

Explanation of above code

If we think about the above code we have the expression `int k=++i *7+2-j--`—
Here we initialized the value `i=4` and `j=21` in this express we use the `++i` this is the pre increment and pre increment having higher priority than `*`, `+`, `-` and post increment so `++i` get executed very first so after executing `++i` your expression like as `int k=5*7+2-j--` after so the `++i` we have the `*` (mutliplicaton) is higher priority operator so `*` multiplication executed after multiplication our code is like as `int k=35+2-j--`—after solve multiplication we have the `+` and `-` operator so `+` and `-` having same priority from left to right so `+` get executed first after `+` execution our equation is `int k=37-j--` after solve `+` we have the two operator `-` and `j--` - Here `j--` is post increment so post increment having less priority than arithmetic operator so minus (`-`) operator solve first so your equation like as `int k= 37-21` so `k=16` and after initialized `k` value `j` will decrement so the value of `j` is 20

Example

```
class IncDec
{
    public static void main(String s[])
    {
        int a = 1;
        int b = 2;
        int c;
        int d;
```

```
c = ++b;  
d = a++;  
c++;  
System.out.println ("a = " + a);  
System.out.println ("b = " + b);  
System.out.println ("c = " + c);  
System.out.println ("d = " + d);  
}  
}
```

Output

```
a = 2  
b = 3  
c = 4  
d = 1
```

Description of above code

In above code a is initialized as 1 and b is initialized as 2 c and d is not initialized if we think about the statement c = ++b then it is pre incremented means first increase the value of b means c=3 and if we think about the statement d = a++ here a++ is post increment means first value of a is initialized in d means d=1 and then a++ executed means 1 increase by 1 from 1 to 2 so a=2 and again c++ means c is also increase by 1 means c was 3 so is c is increment by 1 so c=4

When we print the a it will print 2 when we print the b it will print 3 and when we print c it will print 4 and d is 1

Example

```
package org.techhub;  
  
public class Demo  
{  
  
    public static void main(String[] args)
```

```
{ int i, j, k, l = 0;
  k = l++;
  j = ++k;
  i = j++;
  System.out.println("l is " + i);
}
```

Output

```
l is 1
```

Code Description

If we think about the above code then we have the four variables i, j, k, l=0
If we think about k=l++ here we have the post increment so the initial value of l is 0 this value initialize in k first and after that l++ execute means after execution of this statement k=0 and l=1 and if we think about the j=++k statement here ++k is pre increment statement means first ++k execute and then value initialized in j means k =1 first and then value initialized in j means j is also 1 and if we think about the i=j++ here j++ is post increment means first value of j is initialized in i and then j++ execute means the value of i=1 and j=2 so output is 1

Example

```
package org.techhub;
public class Demo {
    public static void main(String[] args) {
        int dailywage = 4;
        int number_of_days = 5;
        int salary;
        salary = number_of_days++ * --dailywage * dailywage++ * number_of_days--;
        salary -= dailywage;
        System.out.println(dailywage + " " + number_of_days + " " + salary);
    }
}
```

```
}  
}
```

Output

```
4 5 -4
```

Description of above code

If we think about the above code we have the three variables `dailywage=4`
`number_of_days=5` and `salary`

If we think about the statement `salary=number_of_days++ * --dailywage`
`*dailywage++ * number_of_days--` in this expression top most priority operator
is `--dailywage` and `--dailywage` execute very first after executing `--dailywage`
your expression look like as

`salary= number_of_days++ * dailywage * dailywage++` if we put the values in this
expression then expression look like as `salary=5 * 3 * 3` so we have the two
operator in expression `++` and `*` but `++` in the form of post increment and post
increment having less priority than `*` so first multiplication get executed and
result of multiplication stored in `salary` so `salary=45` after solve the multiplication
`++` operator get executed `number_of_days++` before increment
`number_of_days=5` so it will increase by 1 and `number_of_days=6` and after that
`dailywage++` execute so initially `dailywage =4` but previous it is decrease by 1 with
pre decrement operator so `dailywage=3` at the time of multiplication but now
`dailywage++` execute so it will from 3 to 4 means now current value of
`dailywage=4` and after that `number_of_days--` execute so number of days
previous `number_of_days=6` here again `number_of_days` decrease by 1 so it will
again 5

and if we think about the statement `salary = - dailywage` so `dailywage` was 4
previous and when we initialize it in `salary` using negative sign then it will stored

as -4 in salary and if we think about this statement `System.out.println(dailywage + " " + number_of_days + " " + salary);`

So we get the output 4 5 -4

Example

```
package org.techhub;
class Demo
{ public static void main(String s[])
  { int i = 34.0;
    int j = 7;

    int k = i % j;
    System.out.println("k = " + k );
  }
}
```

Output: compile time error

If we think about the above code we try to store `int i=34.0` in integer and 34.0 is double value so we cannot store in integer directly so it will generate the compile time error at run time.

Example

```
package org.techhub;
class Demo
{
  public static void main(String s[])
  {
    int x = 42;
    double y = 42.25;
    System.out.println("x mod 10 = " + x % 10 );
    System.out.println("y mod 10 = " + y % 10 );
  }
}
```

```
}  
}
```

Output

```
x mod 10 = 2  
y mod 10 = 2.25
```

Code description

If we think about the above code we try to apply the % mod operator on double value but after jdk 1.7 version of java % (mod) can apply on floating value so the result is mention above

Example

Which of the following statements is true?

- 1. When $a = 5$ the value of $a \% 2$ is same as $a - 4$
- 2. When $a = 3$ the value of $a * 3 * 3$ is greater than $(a + 10) * 3$
- 3. When $a = 7$ the value of $a * 7 * 3$ is greater than $(a * a + 7 * a + 3)$

Output: 1 and 3

Code Description

Statement 1: $a = 5$. $a \% 2 = 5 \% 2 = 1$ and $a - 4 = 5 - 4 = 1$. Both values are equal. So statement 1 is true.

Statement 2: $a = 3$. $a * 3 * 3 = 27$ and $(a + 10) * 3 = (3 + 10) * 3 = 13 * 3 = 39$.

But 27 is not greater than 39. So statement 2 is false.

Statement 3: $a = 7$. $a * 7 * 3 = 147$ and $(a * a + 7 * a + 3) = (7 * 7 + 7 * 7 + 3) = (49 + 49 + 3) = 101$. 147 is greater than 101. So statement 3 is true.

Extra Tip: Operator Precedence (highest to lowest)

* / % left to right (associativity)

+ - left to right (associativity)

Example

```
package org.techhub;
public class Demo {
    public static void main(String[] args) {
        System.out.println(10 * 5 + 100 * (25 * 11) / (25 * 10) * 10 - 5 + 7 % 2);
        int zx = (10 * 5 + 100 * (25 * 11));
        int yz = ((25 * 10) * 10 - 5 + 7 % 2);
        System.out.println(zx / yz);
    }
}
```

Output

```
1146
11
```

Example

```
package org.techhub;
class Demo{
    public static void main(String args[])
    {
        int var1 = 42;
        int var2 = ~var1;
        System.out.print(var1 + " " + var2);
    }
}
```

Output

```
42 -43
```

Code Description

Unary not operator, ~, inverts all of the bits of its operand. 42 in binary is 00101010 in using ~ operator on var1 and assigning it to var2 we get inverted value of 42 i.e 11010101 which is -43 in decimal.

Example

```
package org.techhub;
class Demo {
    public static void main(String args[]) {
        int a = 3;
        int b = 6;
        int c = a | b;
        int d = a & b;
        System.out.println(c + "\t" + d);
    }
}
```

Output

7	2
---	---

And operator produces 1 bit if both operand are 1. Or operator produces 1 bit if any bit of the two operands is 1.

Example

```
package org.techhub;
class Demo {
    public static void main(String args[])
    {
        byte x = 64;
        int i;
```



```
    byte y;  
    i = x << 2;  
    y = (byte) (x << 2);  
    System.out.print(i + "\t" + y);  
}  
}
```

Output

256 0

Example

```
package org.techhub;  
class Demo {  
    public static void main(String args[])  
    {  
        int x;  
        x = 10;  
        x = x >> 1;  
        System.out.println(x);  
    }  
}
```

Output

5

Example

```
package org.techhub;  
class Demo {  
    public static void main(String args[])  
    {  
        int a = 1;
```

```
int b = 2;
int c = 3;
a |= 4;
b >>= 1;
c <<= 1;
a ^= c;
System.out.println(a + " " + b + " " + c);
}
}
```

Output

3 1 6

Example

```
public class UnaryOperators {
    public static void main(String[] args) {
        int i = 5;
        System.out.print(i++);
        System.out.print(++i);
        System.out.print(i--);
        System.out.print(--i);
    }
}
```

Output

5775

Example

```
public class ShiftOperators {
    public static void main(String[] args) {
        System.out.print(8<<2);
        System.out.print(",");
    }
}
```

```
        System.out.print(8>>1);
    }
}
```

Output:

32, 4

Example

```
public class BitwiseNotOperator{
    public static void main(String[] args) {
        int i = 50;
        System.out.print(~i);
        System.out.print(",");
        System.out.print(~--i);
        System.out.print(",");
        System.out.print(~++i);
    }
}
```

Output

-51,-50,-51

Example

```
public class RelationalOperators {
    public static void main(String[] args) {
        int a = 4;
        int b = 5;
        System.out.print(a>b);
        System.out.print(",");
        System.out.print(a<b);
    }
}
```

Output

false , true

Example

```
public class LogicalOperators {  
    public static void main(String[] args) {  
        int a = 5;  
        int b = 10;  
        boolean flag = a>b;  
        System.out.println(!flag);  
    }  
}
```

Output

True

Example

```
public class TernaryOperators {  
    public static void main(String[] args) {  
        int a = 10;  
        String result = a > 5 ? "Hello 1" : "Hello 2";  
        System.out.println(result);  
    }  
}
```

Output

Hello 1

Example

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

```
{  
    byte b = 12;  
    int y = b;  
    b = b + 10;  
    System.out.println(b);  
}  
}
```

Output

The program will lead to compile time error as explicit casting is required in the line, `b = b + 10`.

Explanation

When you compile the above program, a compile time error occurs in the line, `b = b + 10` as explicit casting is required to assign an int value to a byte type. In the preceding program the byte type variable is declared and initialized to the value 12. Then the value of byte variable is assigned to the int variable, `y` as assigning byte type value to an int variable is possible. However the compilation error of loss of precision will occur while incrementing the byte type value with 10, i.e. int value. Therefore A, C, and D are incorrect options

Example

Which of the following are valid declarations of the main () method?

- A. `static main(String args[]){ }`
- B. `public static String main(String args[]) {... }`
- C. `public static void main(String args[]) {....}`
- D. `final static void main(String args[]) {....}`

Output:

The correct option is C.

Explanation: The following is a valid declaration of the static method: public static void main(String args[]) { //implementation of the main method }

Therefore, the correct option is C since the return type of the main method is void and is declared as static

Example

Ram as a developer was asked to create a program using switch...case within for loop. Ram created the following program:

```
class Ram {  
public static void main(String args[])  
{  
    int z=3;  
    for(int i=0; i<2;i++)  
    {  
        z++;  
        switch(z)  
        {  
            case 3:  
                System.out.print(z=z+1 + " ");  
            case 5:  
                System.out.print(z=z+2 + " ");  
                break;  
            default :  
                System.out.print(z=z+8 + " ");  
            case 6:  
                System.out.print( z=z+4 + " ");  
        }  
    }  
}
```

```
        z--;  
    }  
}  
}
```

Output

12 16 24 28

Explanation: Option D is correct Initially the variable z is initialized with 3, which becomes 4 inside the for loop therefore, the cases before default becomes false and default statement prints 12 then statement following default is executed because break is not used after default and therefore 16 will be printed. Then, the value of z decreases by 1 and becomes 15. Now, for loop executes once again and z becomes 16 and same process continued and z becomes 24 in default case and 28 in next case. Option B is incorrect because for loop is executed two times. Option A is incorrect because value of z is 4 and none of the case statement is 4 and same is the case in option C.

Example

Imagine, you as a student provided with the following program during a class test

```
class Student {  
    public static void main(String args[]){  
        int z=6, k;  
        for(int i=0; i<2;i++) {  
            z++;  
            switch(z) {  
            case 3: System.out.print(z=z+1 + " ");  
            case 5: System.out.print(z=z+2 + " ");  
            break;  
            default : {
```

```
for (int x=10; x>3; x++) {  
    System.out.print(x=k+x + " ");  
}  
}  
case 6: System.out.print( z=z+4 + " ");  
}  
z--;  
}  
}  
}
```

What would be the output of the preceding program?

- A. Program will display 8 10 as an output
- B. Program will not compile successfully
- C. Program runs infinity endless
- D. Program will display 8 10 10as an output

Output: Option B is correct.

Explanation: Option B is correct. Program will not compile because k is not initialized. Option C will be correct when k is initialized but this time it is incorrect. Options A and D are incorrect because the program will not compile successfully.

Example

Ram as a student was provided with the following code snippet

```
public void Ram(float c) {  
    switch (c) {  
        case 5:  
        case 7:  
        case 2:  
        default:  
        case 9.5:
```



```
}  
}
```

After viewing the code snippet Ram was asked to notice the problems in preceding code snippet on the basis of the rules regarding switch...case statement. Following are the options from which Ram has to choose the correct answer.

- A. There is no problem in the code snippet
- B. Switch cannot evaluate float value
- C. The default statement cannot be used between case statements
- D. All cases must be in increasing order

Output

Option B is correct.

Explanation: Option B is correct because switch...case statement can evaluate to a char, byte, short, int, or enum. Therefore Option A is incorrect because float value is being used as a case, which is not allowed. Option C is incorrect because default can be used anywhere in switch...case block. Option D is incorrect because it is not necessary that cases must be in increasing order.

Example

Sheela as a faculty given following options to her students and asked them to choose the correct options:

- A. A switch statement can only evaluate to float and double values
- B. A switch...case block must have break statements after every case
- C. Switch case must be similar to switch expression type
- D. A switch...case can be nested like nested if...else

Option D is correct.

Explanation: Option D is correct because it is a fact.

Option A is incorrect because a switch...case cannot evaluate float and double.

Option B is incorrect because there is no need to have break statement after every case. Option C is incorrect, for example, your case expression is of char type but you used 65 as case label then internally that 65 is recognized as char A.

Therefore, case expression and case label can be varied but must take attention before using them

Example

Shyam during an interview was provided with following code and asked to review the program

```
public class Sam
{
    public static void main(String args[])
    {
        int x=0, i=0;
        for (int y=0; y>=i; ++y,i++) {
            System.out.println(y);
            System.out.println(i);
        }
    }
}
```

After reviewing the code he was asked to predict the correct options among the following:

A. Program will print 0 0 for first time

- B. Program results in an endless loop
- C. Program will not compile because declaration is not allowed inside the for loop
- D. Program will successfully compile and print 0 0 on execution and then terminates

Option B is correct.

Explanation: Option B is correct because the value of y will always be equals to i and therefore loop will never terminates so, option A is incorrect.

Option C is incorrect because declaration can be done inside the for loop.

Option D is incorrect because program will successfully compile but when executes it becomes an endless loop.

Example

Rani during an interview was shown the following program:

```
class Rani {  
    public static void main(String args[]) {  
        int x = 0;int y=9;  
        for ( ; x<y; ) { x++; y++;} // (a)  
        for (x; x==y; --x) continue; // (b)  
        for (x=0; x<5; ) { x++; } // (c)  
        for ( ; ; ) ; // (d)  
    }  
}
```

What would be the output of the preceding program from the following options?

- A. Program will successfully compile and executes but does not print any value
- B. Program will successfully compile and becomes endless because of loop d
- C. Program will not compile because loop b is syntactically incorrect

D. Program will not compile because loop a has only expression part but missing initialization and Increment/decrement part

Option C is the correct answer.

Explanation: Correct answer is option C because the loop is not initialized. Therefore, options A and B are incorrect. Option D is incorrect because syntax of loop is correct.

Example

Shyam was given the following code snippet during an interview and asked to choose all correct decisional and loop statements:

```
int y=9;
for ( ;true ; ) { break;} // 1
if(y==9) { break; } // 2
switch(y) {default: break;} // 3
do ( ) { // code } while(expression); // 4
while ( ) { //code } // 5
```

Options:

- A. Statement 1 and 3 are correct B. Statement 1, 2, and 3 are correct
C. Statement 1, 3, and 5 are correct D. Statement 1,4, and 5 are correct
E. Statement 1, 2, and 4 are correct

Correct option is A.

Explanation: Correct option is A. Reason for all other options can be verified on the basis of following facts:

The break statement can only be used in looping constructs and if need to use with if then if must be inside a loop the do... while do not have expression part with do i.e. do () .The expression is used in while block. The while block Cannot be used without specifying expression.

Example

Shyam was given the following program by his teacher

```
class Sam {  
    public static void main(String args[])  
    {  
        int y=2;int i;  
        for (i=0; i <= 3; i++) {  
            if (i == 2) {  
                break;  
            }  
            else  
            {  
                y++;  
            }  
        }  
        System.out.println(i + " , " + y);  
    }  
}
```

What would be the output of the preceding code?

- A. Program will display 2 , 2 as an output
- B. Program will display 2 , 3 as an output
- C. Program will display 2 , 4 as an output
- D. Program will display 1 , 2 as an output

Option C is the correct.

Explanation: Option C is the correct output. Inside the for loop condition i==2 is checked and when it Evaluates to false the value of y will increment by 1. This process continues until i is not equal to 2. When the condition i==2 evaluates to true then the loop terminates and the vales of i and y are printed.

Rest all of the options are incorrect as they are representing incorrect output.

Example

Sheela after attending a lecture on for statement was shown the following for statements to choose the correct for statement:

- A. for(int j=2; j*j==4, j<4; j++) B. for (int j=3; j/2==1; j++)
C. for (int j=3, long k=0; j>k; j++) D. int k, j; for (j=3, k=2; k==j-1; k++, j--)

Options B and D are correct for statements.

Explanation: Options B and D are correct for statements.

Option A is incorrect because multiple conditions cannot be used in for statement. Option C is incorrect because different type of initialization variable cannot be declared inside the for loop rather they can be declared outside the for loop.

Example

Shyam works in a xyz company and he designed the following program:

```
class Shyam {  
    public static void main (String args[]) {  
        int x=2; int y=6;  
        if( x!=y || (y*=x)!=x) {  
            System.out.println(" Not equal");  
        }  
        else{  
            System.out.println(" Equal");  
        }  
    }  
}
```

What happens when he compile and run the preceding program?

- A. Program will display Equal
- B. Program will display Not Equal
- C. Program will not compile successfully because if statement is not correct
- D. Program will compile but not executes

Option B is the correct answer.

Explanation: Option B is correct because the first expression in if statement evaluates to true therefore Second expression is not checked and the statement in else block is displayed. Option A is incorrect because if statement evaluates to false. Options C and D are incorrect because the program compiles and executes successfully

Example

Rems and Sam while preparing for Java certification created the following program:

```
public class Rose{
    public static void main(String[] args)
    {
        char x = 'a';
        switch(x)
        {
            case 66: System.out.println( "B" +" ");break;
            case 72: System.out.println( "H"+ " ");break;
            case 97: System.out.println("a"+ " ");
            case 89: System.out.println( "Y" + " ");break;
            default: System.out.println( "default");break;
        }
    }
}
```

```
}  
}
```

What would be the output of this program? Choose the correct option from the following options:

- A. Program will display a Y default
- B. Program will display a Y
- C. Program will not compile successfully because break cannot be used with default case.
- D. Program will display A

Option B is the correct answer.

Explanation:

Option B is correct because ASCII equivalent of small a is 97 and because break is not used so, statement following this case is also displayed. Option A is incorrect because break statement is used in case 89 (ASCII equivalent of Y).

Option C is incorrect because break can be used with default statement.

Option D is incorrect because ASCII equivalent of capital A is 65, which is not a case in this program.

GIRI'S TECH HUB PVT.LTD PUNE - 9175444433, 9049361265

GIRI'S TECH HUB – PUNE -9175444433, 9049361265

GIRI'S TECH HUB PVT.LTD PUNE - 9175444433, 9049361265

GIRI'S TECH HUB – PUNE -9175444433, 9049361265

GIRI'S TECH HUB PVT.LTD PUNE - 9175444433, 9049361265

GIRI'S TECH HUB – PUNE -9175444433, 9049361265

GIRI'S TECH HUB PVT.LTD PUNE - 9175444433, 9049361265

GIRI'S TECH HUB – PUNE -9175444433, 9049361265