# QUESTION BANK WITH ANSWERS
# PROGRAMMING IN CORE JAVA
# IV SEM BCA
# 20BCASD41

## UNIT I
## PART A

## Multiple Choice Questions

1. Who invented Java Programming Language?

**A. James Goosling**

B. Newell simon

C. John Mc.Carthy

D. Niklus Wirth

2. In which year was Java invented?

A. 1989

B. 1990

**C.1991**

D.1992

3. Java compiler translates source code into _____

A. Machine code

**B.Byte code**

C. Assembly code

D. Binary code

4. _____generates machine code that can be directly executed by the machine that is running the Java program.

A. Java Compiler

**B.Java interpreter**

C. Java Editor

D. Java Applet viewer

5. _____ translates source code to byte code in Java

A. Java interpreter

**B. Java Compiler**

C. Java Virtual Machine

D. D. Java Editor

6. In a Java environment, _____ command creates a .class file corresponding to the Java program

A. java

B. gedit

**C.javac**

D.compile

7. Which process in a Java program execution will convert the program to byte code?
A. compilation
B. interpretation
C. execution
D. Editing
8. The acronym of JDK is
A. Java Developers Kit
**B. Java Development Kit**
C. Java Designer Kit
D. Java Deployment Kit
9. The full form of API is
A. Applet Programming Interface
**B. Application Programming Interface**
C. Application Processing Interface
D. Application Process Interaction
10. Executing a stand-alone Java program is a_____process
A. one step
B. **Two step**
C. Interpreted
D. compiled
11. _____are small Java programs developed for Internet applications
**A. Applets**
B. Drivers
C. Java Menus
D. Java Machine
12. How many classes are allowed in any Java program?
A. only one
B. only two
C. Only three
**D. any number**
Questions based on Understanding
13. How many classes can contain main method in any Java program?
**A. only one**
B. only two
C. Only three
D. any number
14. Smallest individual units in a Java program are known as _____
**A. tokens**
B. keywords
C. numbers
D. strings
15. How many keywords are present in Java language?
A. 30
B. 40
C. 45
**D. 60**
16. Programmer designed tokens are called _____
**A. identifiers**
B. keywords

C.separators
D.operators
17. What is the maximum length of an identifier in Java?
A. 8 characters
B. 16 characters
C. 20 characters
**D. No limit**
18. _____are symbols used to indicate where groups of code are divided and arranged
A. operators
**B.Separators**
C.literals
D.identifiers
19. Which of the following is/are separator(s) in Java?
A.()
**B.**{}
C.[]
**A. All the above**
20. Mention the separator used in every line of a Java program.
A. {
B. [
**C.;**
D.,
21. JVM stands for _____
A. Java Visualization Machine
**B.Java Virtual Machine**
C.Java Visual Machine
D.Java Vigilant Machine
22. _____are parameters that are supplied to the application program at the time of invoking it for execution.
**A.Command line arguments**
**B.**Arrays
C.String
D.Reference parameters
23. A_____is an identifier that denotes a storage location used to store a data value.
A. Literal
**B.variable**
C.keyword
D.Reserve word
24. How do you assign values to variables in Java?
**A.using assignment statement**
B.using read method
C.using symbolic constants
D.A or B
25. Variables declared and used inside methods are called _____
A. instance variables
**B.local variables**
C.class variables
D.global variables

## QUESTION AND ANSWER

### 1. List and Explain the features of java programming language

Java has following features.

**Compiled and Interpreted**

First Java compiler translates source code into what is known as *bytecode* instructions, Bytecodes are not machine instructions and therefore, in the second stage, Java interpreter generates machine code that can be directly executed by the machine that is running the Java program.

**Platform-Independent and Portable**

The most significant contribution of Java over other languages is its portability. Jaya programs can be easily moved from one computer system to another, anywhere and anytime.

**Object-Oriented**

Java is a true object-oriented language. Almost everything in Java is an *object.* All program code and data reside within objects and classes.

**Robust and Secure**

Java is a robust language. It provides many safeguards to ensure reliable code. It has strict compile time and run time checking for data types.

**Distributed**

Java is designed as a distributed language for creating applications on networks. It has the ability to share both data and programs.

**Simple, Small and Familiar**

Java is a small and simple language. Java does not use pointers, pre processor header files, go to statement and many others. It also eliminates operator overloading and multiple inheritance. Java is modelled on C and C++ languages. So it is familiar to programmers.

**Multithreaded and Interactive**

Multithreaded means handling multiple task-simultaneously. Java supports multithreaded programs

**High Performance**

Java performance is impressive for an interpreted language, mainly due to the use of intermediate bytecode. the incorporation of multithreading enhances the overall execution speed of Java programs.

### 2. Explain the structure of a java program with an example.

A Java program may contain one or more sections.

**Documentation Section**

The documentation section comprises a set of comment lines giving the name of the program, the author and other details,. Java permits both the single-line comments and multi-line comments.

/**….....*/ known as *documentation comment.* This form of comment is used for generating documentation automatically. LUI

**Package Statement**

The first statement allowed in a Java file is a package statement. This statement declares a package name and informs the compiler that the classes defined here belong to this package. For example: package student; The package statement is optional.

**Import Statements**

The next thing after a package statement (but before any class definitions may be a number of import statements. For example import student test. This statement instructs the interpreter to load the test class contained in the package student.

**Interface Statements**

An interface is like a class but includes a group of method declarations. This is also an optional section and is used only when we wish to implement the multiple inheritance features in the program.

**Class Definitions**

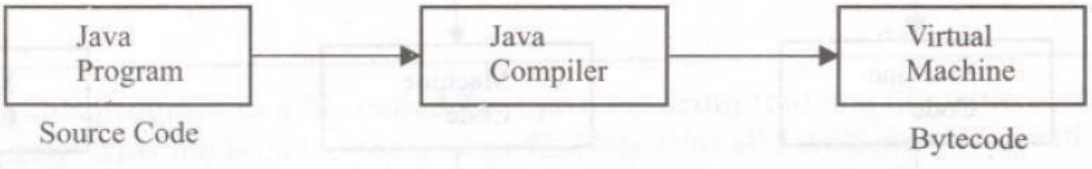
A Java program may contain multiple class definitions. Classes are the primary and essential elements of a Java program.

**Main Method Class**

Since every Java stand-alone program requires a main method as its starting point, this class is the essential part of a Java program. A simple Java program may contain only this part.

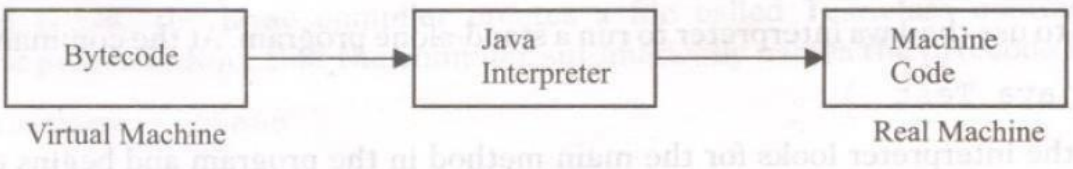
**3.Explain the contribution of Java Virtual Machine in executing a Java program**.

All language compilers translate source code into *machine code* for a specific computer. Java compiler also does the same thing. The Java compiler produces an intermediate code known as bytecode for a machine that does not exist. This machine is called the *Java Virtual Machine* and it exists only inside the computer memory. It is a simulated computer within the computer and does all major functions of a realcomputer.




*Process of compilation*

The virtual machine code is not machine specific. The machine specific code known as machine code) is generated by the Java interpreter by acting as an intermediary between the virtual machine and the real machine.




*Process of converting bytecode into machine code*

**3. List and Explain the different data types used in Java.**

Every variable in Java has a data type. Data types specify the size and type of values that can be stored. Java language is rich in its *data types.* The variety of data types available, allow the programmer to select the type appropriate to the needs of the application.

**Integer Types**

Integer types can hold whole numbers such as 123, -96, and 5639. The size of the values that can be stored depends on the integer data type we choose.

**Floating Point Types**

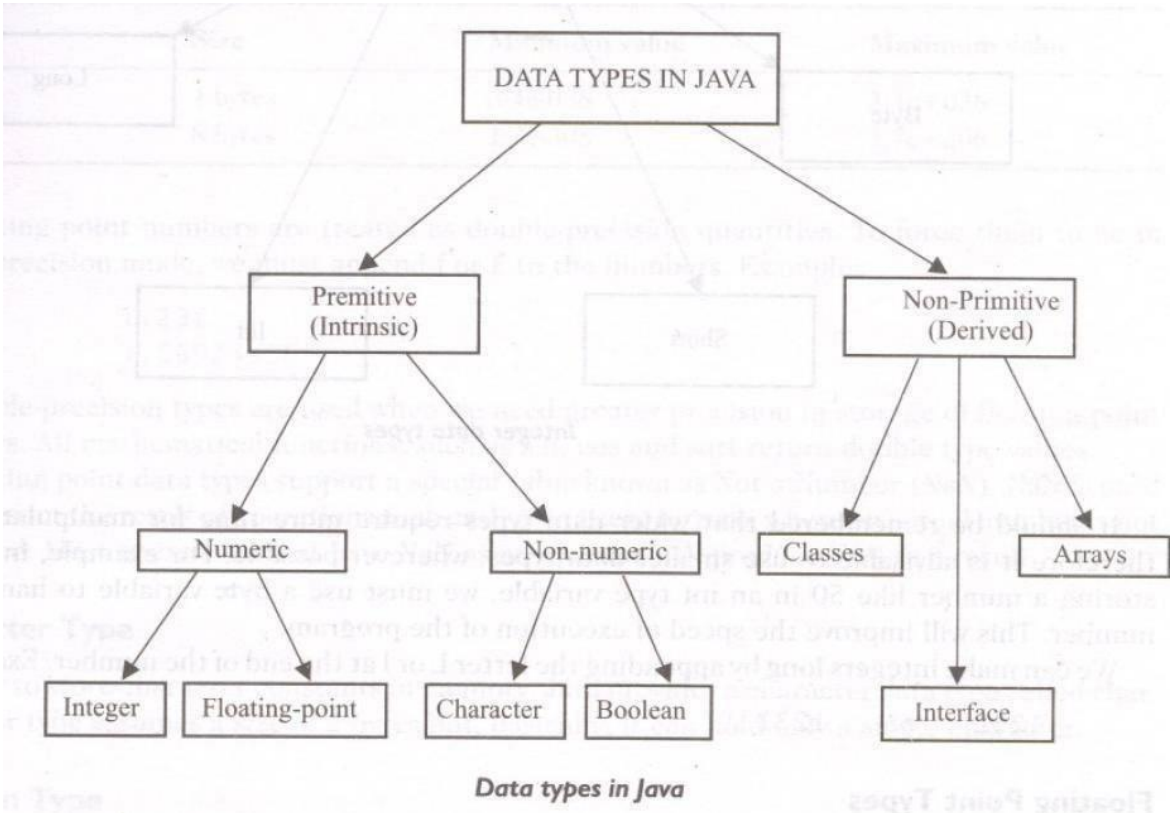
Integer types can hold only whole numbers and therefore we use another type known *as floating point* type to hold numbers containing fractional parts such as 27.59 and -1.

**Character Type**

In order to store character constants in memory, Java provides a character data type called char. The char type assumes a size of 2 bytes but, basically, it can hold only a single character.

**Boolean Type**

Boolean type is used when we want to test a particular condition during the execution of the program.
There are only two values that a boolean type can take: true or false.




*Data types in Java*

**5. Define a variable. How do you declare the variable in Java? Explain with an example.**

A *variable* is an identifier that denotes a storage location used to store a data value. Unlike constants that remain unchanged during the execution of a program, a variable may take different values at different times during the execution of the program

Variable names may consist of alphabets, digits, the underscore ( -) and dollar characters, subject to the following conditions:

1. They must not begin with a digit.
2. Uppercase and lowercase are distinct.

3. It should not be akeyword.
4. White space is not allowed.
5. 5. Variable names can be of anylength.

The general form of declaration of a variableis:
A variable must be declared before it is used in the program. A variable can be used to store a value of any data type.

```
type variable1, variable2, ....., variableN;
```

Variables are separated by commas, A declaration statement must end with a semicolon, Some valid declarations are:

int count;
float x, y;
doublepi;

**6. How do different types of operators in Java help in writing a program for any type of application?**
Java operators can be classified into a number of related categories as below:
1. Arithmetic operators
2. Relational operators
3. Logical operators
4. Assignment operators
5. Increment and decrement operators
6. Conditional operators.
7. Bitwise operators
8. Special operators

**ARITHMETIC OPERATORS**
Java provides all the basic arithmetic operators. These can operate on any built-in numeric data type of Java. We cannot use these operators on boolean type Arithmetic operators are used as shown below: a-b a*b a%b a+b a/b -a*b Here a and b may be variables or constants and are known as *operands.*

**LOGICAL OPERATORS**
In addition to the relational operators, Java has three logical operators, Logical Operators
The logical operators && and | | are used when we want to form compound conditions by combining two or more relations. An example is: a > b && x 10

**ASSIGNMENT OPERATORS**
Assignment operators are used to assign the value of an expression to a variable,

**INCREMENT AND DECREMENTOPERATORS**
Java has two very useful operators not generally found in many other languages. These are the increment and decrement operators: ++ and--

**CONDITIONAL OPERATOR**
The character pair ?: is a ternary operator available in Java. This operator is used to construct conditional expressions of the form
Where *expl, exp2,* and *exp3* are expressions.

**BITWISE OPERATORS**

Java has a distinction of supporting special operators known as bitwise operators for manipulation of data at values of bit level.

**SPECIAL OPERATORS**

Java supports some special operators of interest such as instance of operator and member selection operator(.).

**7. Explain Command Line Arguments in Java with an example.**

**COMMAND LINE ARGUMENTS**

There may be occasions when we may like our program to act in a particular way depending on the input provided at the time of execution. This is achieved in Java programs by using what are known as command line arguments. Command line arguments are parameters that are supplied to the application program at the time of invoking it for execution.

We can write Java programs that can receive and use the arguments provided in the command line.

```
/*
 *  This program uses command line
 *  arguments as input.
 */
Class ComLineTest
{
   public static void main(String args[ ])
   {
      int count, i=0;
      String string;
      count = args.length;
      System.out.println("Number of arguments  =  " + count);
      while(i < count)
      {
         string = args[i];
         i = i + 1;
         System.out.println(i+ " : " + "Java is " +string+ "!");
      }
   }
}
```

Compile and run the program with the command line as follows: javaComLineTest Simple Object_Oriented Distributed Robust Secure Portable Multithreaded Dynamic . The number of arguments is obtained by statement count = args.length; The output of the program would be as follows: Number of arguments = 8

1 : Java isSimple!
2 : Java is Object_Oriented!
3 : Java isDistributed!
4 : Java is Robust!
5 : Java is Secure!
6 : Java isPortable!
7 : Java is Multithreaded!
8 : Java isDynamic!

Note how the output statement concatenates the strings while printing.

**8. Explain the different types of tokens in Java .**

Java language includes five types of tokens. They are:

- Reserved Keywords
- Identifiers
- Literals
- Operators
- Separators

. **Keywords**

Keywords are an essential part of a language definition. They implement specific features of the language. Java language has reserved 60 words as keywords. These keywords, combined with operators and separators according to syntax, form definition of the Java language

**Identifiers**

Identifiers are programmer-designed tokens. They are used for naming classes, methods, variables, objects, labels, packages and interfaces in a program.

.

**Literals**

Literals in Java are a sequence of characters (digits, letters, and other characters) that represent constant values to be stored in variables. Java language specifies five major types of literals. They are:

Integer literals

- Floating point literals
- Character literals
- String literals
- Boolean literals
- **Operators**

An operator is a symbol that takes one or more arguments and *operates* on them to produce a result.

**Separators**

Separators are symbols used to indicate where groups of code are divided and arranged. They basically define the shape and function of our code. Some of the separators are parentheses( ), braces{ },brackets[ ], semicolon; comma , period.

**9. Explain how symbolic constants are used in Java with an example**

We often use certain unique constants in a program. These constants may appear repeatedly in a number of places in the program.

Modifiability

We may like to change the value of "pi" from 3.142 to 3.14159 to improve the accuracy of calculations or the number 50 to 100 to process the test results of another class. In both the cases, we will have to search throughout the program and explicitly change the value of the constant wherever it has been used. If any value is left unchanged, the program may produce disastrous outputs.

Understandability

When a numeric value appears in a program, its use is not always clear, especially when the same value means different things in different places Valid examples of constant declaration are:

Final int STRENGTH = 100; final int PASS MARK = 50;

Note that:

1. Symbolic names take the same form as variable names.
2. After declaration of symbolic constants, they should not be assigned any other value within the program by using an assignment statement. For example, STRENGTH = 200; is illegal.
3. Symbolic constants are declared for types.
4. They can NOT be declared inside a method

**10. How do you work with arithmetic expressions in Java? Explain**

An arithmetic expression is a combination of variables, constants, and operators arranged as per the syntax of the language. Java can handle any complex mathematical expressions. Expressions

| Algebraic expression | Java expression |
|---|---|
| a b–c | a*b-c |
| (m+n)(x+y) | (m+n)*(x+y) |
| $\frac{ab}{c}$ | a*b/c |
| $3x^2+2x+1$ | 3*x*x+2*x+1 |
| $\frac{x}{y}+c$ | x/y+c |

**EVALUATIONOF EXPRESSIONS**

Expressions are evaluated using an assignment statement of the form

*Variable* is any valid Java variable name. When the statement is encountered, the *expression* is evaluated first and the result then replaces the previous value of the variable on the left- hand side. All variables used in the expression must be assigned values before evaluation is attempted. Examples of evaluation statements are x= a*b-c; y = b/c*a; z= a-b/c+d;

**11. Compare and contrast Java and C++.**

- Java does not support operator overloading.
- Java does not have template classes as in C++.
- Java does not support multiple inheritance of classes. This is accomplished using a new feature called" interface".
- Java does not support global variables. Every variable and method is declared with in a class and forms part of that class.
- Java does not use pointers.
- Java has replaced the destructor function with a finalize( )function.
- There are no header files in Java.

**12.List and the explain the usage of different types of variables that are used in a Java program.**

Java variables are actually classified into three kinds:

- *instance* variables
- *class* variables
- local variables

Instance and class variables are declared inside a class. Instance variables are created when the objects are instantiated and therefore they are associated with the objects. They take different *values* for each object. On the other hand, class variables are' global to a class and belong to the entire set of objects that class creates. Only one memory location is created for each class variable.

Variables declared and used inside methods are called *local variables.* They are called so because they are not available for use outside the method definition. Local variables can also be declared inside program blocks that are defined between an opening brace { and a closing brace }. These variables are visible to the program only from the beginning of its program block to the end of the program block.

When the program control leaves a block, all the variables in the block will cease to exist. The area of the program where the variable is accessible is called its *scope.*

In Java, every variable has a default value. If we don't initialize a variable when it is first created, Java provides default value to that variable type automatically.

| Type of variable | Default value |
|---|---|
| byte | Zero : (byte) 0 |
| Short | Zero : (short) 0 |
| int | Zero : 0 |
| long | Zero : 0L |
| float | 0.0f |
| double | 0.0d |
| char | null character |
| boolean | false |
| reference | null |