

Q.1. Write the features of Java Programming.

Ans:

The primary objective of **Java programming** language creation was to make it portable, simple and secure programming language. Apart from this, there are also some excellent features which play an important role in the popularity of this language. The features of Java are also known as java *buzzwords*.

A list of most important features of Java language is given below-

1. Simple
2. Object-Oriented
3. Portable
4. Platform independent
5. Secured
6. Robust
7. Architecture neutral
8. Interpreted
9. High Performance
10. Multithreaded
11. Distributed
12. Dynamic

Q.2. Why we say Java is a Portable Language.

Ans:

Java is Considered as Platform independent because of many different reasons which are listed below –

1. Output of a Java compiler is **Non Executable Code i.e Bytecode**.
2. Bytecode is a **highly optimized set of instructions**
3. Bytecode is executed by Java run-time system, which is called the Java Virtual Machine (**JVM**).

Q. 3. Short Notes on JVM.

Ans:

JVM (Java Virtual Machine) is an abstract machine. It is called a virtual machine because it doesn't physically exist. It is a specification that provides a runtime environment in which Java bytecode can be

executed. It can also run those programs which are written in other languages and compiled to Java bytecode.

JVMs are available for many hardware and software platforms. JVM, JRE, and JDK are platform dependent because the configuration of each OS is different from each other. However, Java is platform independent. There are three notions of the JVM: *specification*, *implementation*, and *instance*.

The JVM performs the following main tasks:

- Loads code
- Verifies code
- Executes code
- Provides runtime environment

Q.4 Write about Java Variables.

Ans:

A variable is a container which holds the value while the Java program is executed. A variable is assigned with a data type.

Variable is a name of memory location. There are three types of variables in java: local, instance and static.

1) Local Variable

A variable declared inside the body of the method is called local variable. We can use this variable only within that method and the other methods in the class aren't even aware that the variable exists.

A local variable cannot be defined with "static" keyword.

2) Instance Variable

A variable declared inside the class but outside the body of the method, is called instance variable. It is not declared as **static**.

It is called instance variable because its value is instance specific and is not shared among instances.

3) Static variable

A variable which is declared as static is called static variable. It cannot be local. We can create a single copy of static variable and share among all the instances of the class. Memory allocation for static variable happens only once when the class is loaded in the memory.

Q.5. Write about Java Data Types.

Ans:

Data types specify the different sizes and values that can be stored in the variable. There are two types of 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**.

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

There are 8 types of primitive data types:

- boolean data type
- byte data type
- char data type
- short data type
- int data type
- long data type
- float data type
- double data type

Q.6. What we mean Keywords? Mention Java Keywords.

Ans:

Java keywords are also known as **reserved words**. Keywords are particular words which acts as a key to a code. These are predefined words by Java so it cannot be used as a variable or object name.

List of Java Keywords

A list of Java keywords or reserved words are given below:

1. **abstract:** Java abstract keyword is used to declare abstract class. Abstract class can provide the implementation of interface. It can have abstract and non-abstract methods.
2. **boolean:** Java boolean keyword is used to declare a variable as a boolean type. It can hold True and False values only.
3. **break:** Java break keyword is used to break loop or switch statement. It breaks the current flow of the program at specified condition.

4. **byte:** Java byte keyword is used to declare a variable that can hold an 8-bit data values.
5. **case:** Java case keyword is used to with the switch statements to mark blocks of text.
6. **catch:** Java catch keyword is used to catch the exceptions generated by try statements. It must be used after the try block only.
7. **char:** Java char keyword is used to declare a variable that can hold unsigned 16-bit Unicode characters
8. **class:** Java class keyword is used to declare a class.
9. **continue:** Java continue keyword is used to continue the loop. It continues the current flow of the program and skips the remaining code at the specified condition.
10. **default:** Java default keyword is used to specify the default block of code in a switch statement.
11. **do:** Java do keyword is used in control statement to declare a loop. It can iterate a part of the program several times.
12. **double:** Java double keyword is used to declare a variable that can hold a 64-bit floating-point numbers.
13. **else:** Java else keyword is used to indicate the alternative branches in an if statement.
14. **enum:** Java enum keyword is used to define a fixed set of constants. Enum constructors are always private or default.
15. **extends:** Java extends keyword is used to indicate that a class is derived from another class or interface.
16. **final:** Java final keyword is used to indicate that a variable holds a constant value. It is applied with a variable. It is used to restrict the user.
17. **finally:** Java finally keyword indicates a block of code in a try-catch structure. This block is always executed whether exception is handled or not.
18. **float:** Java float keyword is used to declare a variable that can hold a 32-bit floating-point number.
19. **for:** Java for keyword is used to start a for loop. It is used to execute a set of instructions/functions repeatedly when some conditions become true. If the number of iteration is fixed, it is recommended to use for loop.
20. **if:** Java if keyword tests the condition. It executes the if block if condition is true.
21. **implements:** Java implements keyword is used to implement an interface.
22. **import:** Java import keyword makes classes and interfaces available and accessible to the current source code.
23. **instanceof:** Java instanceof keyword is used to test whether the object is an instance of the specified class or implements an interface.
24. **int:** Java int keyword is used to declare a variable that can hold a 32-bit signed integer.

25. **interface:** Java interface keyword is used to declare an interface. It can have only abstract methods.
26. **long:** Java long keyword is used to declare a variable that can hold a 64-bit integer.
27. **native:** Java native keyword is used to specify that a method is implemented in native code using JNI (Java Native Interface).
28. **new:** Java new keyword is used to create new objects.
29. **null:** Java null keyword is used to indicate that a reference does not refer to anything. It removes the garbage value.
30. **package:** Java package keyword is used to declare a Java package that includes the classes.
31. **private:** Java private keyword is an access modifier. It is used to indicate that a method or variable may be accessed only in the class in which it is declared.
32. **protected:** Java protected keyword is an access modifier. It can be accessible within package and outside the package but through inheritance only. It can't be applied on the class.
33. **public:** Java public keyword is an access modifier. It is used to indicate that an item is accessible anywhere. It has the widest scope among all other modifiers.
34. **return:** Java return keyword is used to return from a method when its execution is complete.
35. **short:** Java short keyword is used to declare a variable that can hold a 16-bit integer.
36. **static:** Java static keyword is used to indicate that a variable or method is a class method. The static keyword in Java is used for memory management mainly.
37. **strictfp:** Java strictfp is used to restrict the floating-point calculations to ensure portability.
38. **super:** Java super keyword is a reference variable that is used to refer parent class object. It can be used to invoke immediate parent class method.
39. **switch:** The Java switch keyword contains a switch statement that executes code based on test value. The switch statement tests the equality of a variable against multiple values.
40. **synchronized:** Java synchronized keyword is used to specify the critical sections or methods in multithreaded code.
41. **this:** Java this keyword can be used to refer the current object in a method or constructor.
42. **throw:** The Java throw keyword is used to explicitly throw an exception. The throw keyword is mainly used to throw custom exception. It is followed by an instance.
43. **throws:** The Java throws keyword is used to declare an exception. Checked exception can be propagated with throws.
44. **transient:** Java transient keyword is used in serialization. If you define any data member as transient, it will not be serialized.
45. **try:** Java try keyword is used to start a block of code that will be tested for exceptions. The try block must be followed by either catch or finally block.

- 46. **void:** Java void keyword is used to specify that a method does not have a return value.
- 47. **volatile:** Java volatile keyword is used to indicate that a variable may change asynchronously.
- 48. **while:** Java while keyword is used to start a while loop. This loop iterates a part of the program several times. If the number of iteration is not fixed, it is recommended to use while loop.

Q. 7. What is Type conversion and casting in Java? Explain.

Ans:

Converting one primitive datatype into another is known as type casting (type conversion) in Java. We can cast the primitive datatypes in two ways namely, Widening and, Narrowing.

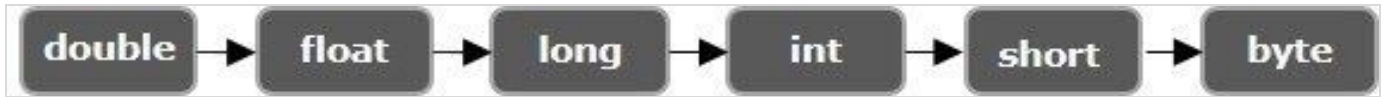
Widening – Converting a lower datatype to a higher datatype is known as widening. In this case the casting/conversion is done automatically therefore, it is known as implicit type casting. In this case both datatypes should be compatible with each other.



Example:

```
public class WideningExample
{
    public static void main(String args[])
    {
        char ch = 'C';
        int i = ch;
        System.out.println(i);
    }
}
```

Narrowing – Converting a higher datatype to a lower datatype is known as narrowing. In this case the casting/conversion is not done automatically, we need to convert explicitly using the cast operator “()” explicitly. Therefore, it is known as explicit type casting. In this case both datatypes need not be compatible with each other.



Example:

```
import java.util.Scanner;

public class NarrowingExample
{
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);

        System.out.println("Enter an integer value: ");

        int i = sc.nextInt();

        char ch = (char) i;

        System.out.println("Character value of the given integer: "+ch);
    }
}
```

