Table of Contents

[1. What is variable: 2](#_Toc38102673)

[2. Java primitive datatypes: java has 8 primitive data types. 2](#_Toc38102674)

[3. Characteristics of java language: why it is so famous ? 2](#_Toc38102675)

[4. Conditions: 2](#_Toc38102676)

[5. Relational Expressions: 3](#_Toc38102677)

[6. Difference between == and .equals() method in Java 3](#_Toc38102678)

[7. What is a string? 4](#_Toc38102679)

[8. What is constructor? 5](#_Toc38102680)

[9. Access Modifiers: 5](#_Toc38102681)

[10. Data Hiding (Encapsulation): 6](#_Toc38102682)

[11. Inheritance: 6](#_Toc38102683)

[12. Overriding: 6](#_Toc38102684)

# What is variable:

Ans: Variable is any character(s) which is used to hold some value. It is used with the java datatype to hold its value.

Ex: int x = 0; 🡪 here x is variable holding value as ‘0’.

String name = “Virginia”; 🡪 here name is a variable holding some value.

## Java primitive datatypes: java has 8 primitive data types.

|  |  |
| --- | --- |
| DataType | Size it can hold in the memory |
| byte | 8 bits |
| short | 16 bits |
| int (numbers) | 32 bits |
| long (numbers) | 64 bits |
| float (decimals values) | 32 bits |
| double (too hold big decimal values) | 64 bits |
| char (for ex: ‘c’, ‘b’, ‘2’,’1’) | 16 bits |
| boolean (true or false) | 1. bit |

## Characteristics of java language: why it is so famous ?

Ans:

1. Platform Independent:

Java is platform independent language, write once and execute anywhere (if the particular device has JVM/JRE installed).

# Conditions:

In programming world, conditions are computations, each condition computes some expression after the condition matches in true/false terms.

# Relational Expressions:

|  |  |
| --- | --- |
| Expression | Definition |
| == | Equals operator, which Compare 2 values/objects, and checks if objects are equal. If objects are equal it returns true otherwise false. |
| != | Not equals operator, it checks if 2 objects are not equal. If objects are not equal it returns true otherwise false. |
| > | Greater than, to check if one object is greater than another (returns true). |
| < | Less than, to check if one object is less than other |
| >= | Greater than equals to, to check if one object is greater than or equals to another object. |
| <= | less than equals to, to check if one object is less than or equals to another object. |

# Difference between == and .equals() method in Java

In general both equals() and “==” operator in Java are used to compare objects to check equality but here are some of the differences between the two:

Main difference between .equals() method and == operator is that one is method and other is operator.

We can use == operators for reference comparison (address comparison) and .equals() method for content comparison. In simple words, == checks if both objects point to the same memory location whereas .equals() evaluates to the comparison of values in the objects.

If a class does not override the equals method, then by default it uses equals(Object o) method of the closest parent class that has overridden this method. See this for detail

Example:

**public class** DifferenceEqualsAndEqualOperator {  
  
 **public static void** main(String[] args) {  
 String name = **new** String(**"foo bar"**); *//String object 1* String anotherName = **new** String(**"foo bar"**); *//String object 2* System.***out***.println(**"With equal operator : "**);  
 System.***out***.println(name == anotherName);  
  
 System.***out***.println(**"With String Equals method :**

**"**+name.equals(anotherName));  
  
  
 }  
}

# What is a string?

Solution: String is an array of characters which holds alphanumeric values or any text. String is immutable in nature, that means value of the String won’t be changed regardless of how many operations you would apply. Value of String will be changed only on the new assignment.

For example:

String names = “Foo Bar”; Java creates array of characters internally for string values.

Array – collection of uniform objects or data (think Excel)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Char(s) | F | O | O |  | B | A | R |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 |

Ternary Operator:

String name = “US”;

# What is constructor?

Solution :

Constructor is a method which has the same name as the class name, it doesn’t return any values. Constructor is used to setup/assign/initialize the values of the member variables of the particular class.

Note: Default constructor is automatically created by JVM behind the scenes.

# Access Modifiers:

Solution:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Access Modifier | Class | Package | SubClass | Project |
| public | X | X | X | X |
| protected | X | X | X |  |
| default | X | X |  |  |
| private | X |  |  |  |

General rules of class:

It is always good to use:

1. private for data/member variables.
2. public for methods.
3. private for inner classes.

# Data Hiding (Encapsulation):

It is the concept of protecting the data/member variables form accidental manipulations, private access modifier is used to protect the data/member variables of any class.

When we protect the data/member variables, still objects need to access the members of the class?

Solution: use setter methods to set the values of the member variables and get methods to retrieve the values of the member variables:

# Inheritance:

When one class acquires properties of another class it’s called Inheritance.

Ex: Imagine the class which does not belong to me/my project, I got it from opensource/someone else, I won’t be able to modify the contents (member variables and methods) of that class, since someone wrote that class. So, I need to use inheritance (meaning I would need to extend the given class in new class and I would need to create new method, which I would like to change).

# Overriding:

When a method in the subclass has the same signature as a method in the super class, then the subclass method overrides and takes precedence/priority and it will be executed.

# Interfaces:

Interfaces are used to declare set of rules, Interface is like a class in which you can declare empty methods and provide the implementations later in the classes.

## Abstract class:

Abstract class is a class with fully defined methods and empty methods. The limitation of using abstract class is that java allows only single inheritance, you cannot extend multiple classes at a time. (Tip : use interfaces if you want to use multiple implementations of same methods.)