Java Training Agenda

Core Java

Build Tool- Maven

Version tool - Git

Testing – Junit

Logging- Log4J

Java 8 features

Database – MySQL / Oracle

JDBD

Spring Core

Spring MVC

Web Services

1. SOAP / XML Service
2. RESTful web services
   1. Spring Restful services

Spring boot with Restful service

Spring data JPA

Mockito

Spring boot testing

Code Quality – Sonarcube

Day-5

Constructor

Whenever object is build the constructor will be invoked.

What is the need of constructor?

Constructor is used to initialize the values to the variables

Rules:

Constructor is a function

Function name and class name should be the same (Case sensitive)

Does not return. Even void.

Access modifier is public, never private.

Types of Constructors:

1. Default constructor
2. Non parameter constructor
3. Parameterized constructor
4. Copy constructor

Ex: public class Employee {

Int empNo;

Int age;

String empName;

}

Sop (employee.empNo) == default value == 0

Sop (employee.empName) == default value == null

Sop (employee.age) == default value == 0

If we want non default values of the variables, we need to assign the values to the variables first. It can be achieved by

1. employee.setProperty()
2. Using the constructor

Ex for non-parameterized constructor:

Public Employee () {

this.empNo = 100;

this.age = 25;

this.empName = “Bharath”;

}

If we don’t write constructor in our program, JVM will create the default constructor.

Employee employee = new Employee () // Invoking the non-parameterized constructor

Ex for parameterized constructor:

Public Employee (int empNo, int age, String empName) {

this.empNo = empNo;

this.age = age;

this.empName = empName;

}

Employee employee1 = new Employee (10, “Chinna, 25000f) // Invoking the 2 constructor

Public Employee (int empNo, int age) {

this.empNo = empNo;

this.age = age;

this.empName =”Bha”;

}

Employee employee2= new Employee (10, 25000f) // Invoking the 3 constructor

Here we mention the 3 constructors with the same name Employee. But when invoking we invoked with the respective parameters. This is called the static polymorphism or static constructor overloading.

Any Java file that we create that should have a default or non-parameterized constructor. So the users can assign their own data to the variables.

Constructor will be called after the object creation in the main class.

Destructor

Final

What is the Final Keyword in Java?

Java final keyword is a non-access specifier that is used to restrict a class, variable, and method. If we initialize a variable with the final keyword, then we cannot modify its value.

If we declare a method as final, then it cannot be overridden by any subclasses. And, if we declare a class as final, we restrict the other classes to inherit or extend it.

In other words, the final classes cannot be inherited by other classes.

This was a brief introduction to a final keyword in Java. Now we will discuss the implementation of the final keyword with a variable, method, and class in Java. We will discuss the following topics in detail:

**1.** final variable  
**2.** final class  
**3.** final method

### Final Variable in Java

Once we declare a variable with the final keyword, we can’t change its value again. If we attempt to change the value of the final variable, then we will get a compilation error.

Generally, we can consider a final variable as a constant, as the final variable acts like a constant whose values cannot be changed.

This rule was for the normal variables, what if we declare the reference variables as final? The restriction with the final reference variable is that we cannot change “what the object is referring to” but we can modify the object itself.

We can’t just declare a variable as final without initializing it. That is, we have to give it an initial value while declaring a final variable. If we declare the final variable without initialization, then it is a blank final variable.

But it is mandatory to initialize a final variable either during declaration or after declaration. If we leave it uninitialized, then we will get a compilation error.

**Syntax of defining a final variable:**

final int number = 10;                             //final variable  
final float value;                                       //blank final variable  
static final double rate = 2.5;             //final and static variable

#### **Initializing Final Variable in Java**

As we know that a final variable cannot be left uninitialized, the following are the ways to initialize a final variable:

**1.** The most common approach is to initialize a final variable during its declaration. But if you don’t initialize it while declaring it, then we call it as a blank final variable.

The next two ways are the ways to initialize a blank final variable.

**2.** We can initialize a blank final variable inside an instance-initializer block or inside the constructor of the class. In case, there are many constructors in the class, then you need to initialize the final variable inside each constructor, else there will be an error.

**3.** We can initialize a blank final static variable inside a static block.

Aggregation

* Has a relation

Inheritance

* Is a relation

Class diagram in Java

1. Name of class
2. Variables
   1. + Public
   2. – Private
3. Methods

String

String is a data type and not a primitive type

String is collection of character’s which are enclosed in the double quotes. Whereas characters are defined by enclosing with in a single quotes.

Ways to declare the String

1. String varName = “value”;
2. String varName = new(“value”)

String objects are immutable.

Assignment

Employee

empNo, empName, salary

Address

doorNo, city, state

1. One to one
   1. Find out whether the employee is staying in “Chennai”
      1. Emp
      2. Add
      3. Service
      4. Main

Output

* + - Yes (Boolean)
    - String (This employee is staying in Chennai)
    - Employee (Object)

1. One to Many
   1. Find out whether employee is staying in “Bangalore”