Java Interview Questions

30 April 2022 14:18

© Rajat Kumar
https://www.linkedin.com/in/imRajat/
https://github.com/im-Rajat

Following : https://www.interviewbit.com/java-interview-questions/

Java Basic Interview Questions

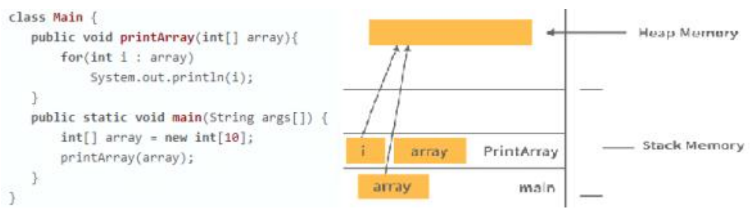
Java Intermediate Interview Questions

Q) What is Java?
- Java is the high-level programming language
- It is based on the principles of object-oriented programming
- And can be used to develop large-scale applications

Q) Why is Java a platform independent language?
- does not depend on any hardware or software due to the fact that the compiler compiles the code
- and then converts it to platform-independent byte code which can be run on multiple systems.
- The only condition to run that byte code is for the machine to have a runtime environment (JRE) installed in it

Q) Why is Java not a pure object oriented language?
- Java supports primitive data types - byte, boolean, char, short, int, float, long, and double and hence it is not a pure object oriented language

Q) Difference between Heap and Stack Memory in java. And how java utilizes this?
- Stack memory is the portion of memory that was assigned to every individual program. And it was fixed. (Assigned during compile time)
- heap memory is the portion that was not allocated to the java program but it will be available for use by the java program when it is required (mostly during the runtime of the program)
- Java Utilizes this memory as -
    - When we write a java program then all the variables, methods, etc are stored in the stack memory.
    - And when we create any object in the java program then that object was created in the heap memory. And it was referenced from the stack memory.



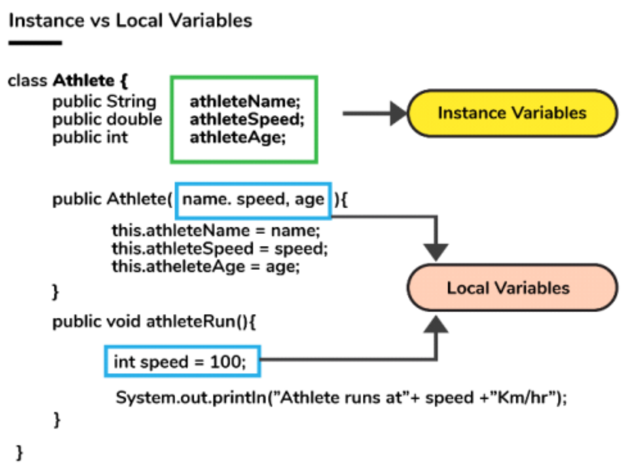
```
class Main {
    public void printArray(int[] array){
        for(int i : array)
            System.out.println(i);
    }
    public static void main(String args[]) {
        int[] array = new int[10];
        printArray(array);
    }
}
```

Q) Is java a completely object-oriented programming language?
- Java is not a pure object-oriented programming language, because it has direct access to primitive data types

Q) How is Java different from C++?
- C++ is only a compiled language, whereas Java is compiled as well as an interpreted language.
- C++ allows users to use pointers in the program. Whereas java doesn’t allow it. Java internally uses pointers.
- C++ supports the concept of Multiple inheritances whereas Java doesn't support this. And it is due to avoiding the complexity of name ambiguity that causes the diamond problem.

Q) Pointers are used in C/ C++. Why does Java not make use of pointers?
- Pointers are quite complicated and unsafe to use by beginner programmers.
- Java focuses on code simplicity, and the usage of pointers can make it challenging. Pointer utilization can also cause potential errors.
- Moreover, security is also compromised if pointers are used because the users can directly access memory with the help of pointers.

Q) What do you understand by an instance variable and a local variable?
- Instance variables are those variables that are accessible by all the methods in the class.
- Local variables are those variables present within a block, function, or constructor and can be accessed only inside them


Instance vs Local Variables

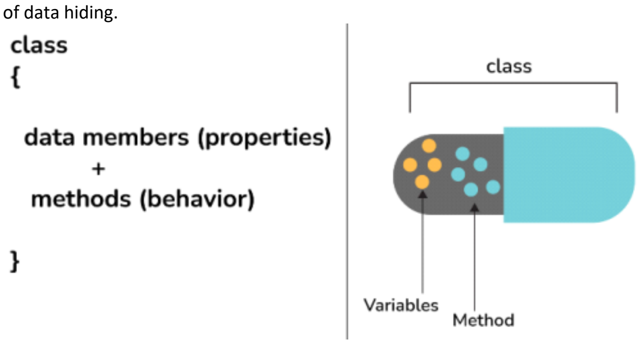
```
class Athlete {
        public String   athleteName;
        public double   athleteSpeed;
        public int      athleteAge;

        public Athlete( name. speed, age ){
                this.athleteName = name;
                this.athleteSpeed = speed;
                this.atheleteAge = age;
        }
        public void athleteRun(){
                int speed = 100;
                System.out.println("Athlete runs at"+ speed +"Km/hr");
        }
}
```

Q) What are the default values assigned to variables and instances in java?

- There are no default values assigned to the variables in java. We need to initialize the value before using it. Otherwise, it will throw a compilation error of (Variable might not be initialized).
- But for instance, if we create the object, then the default value will be initialized by the default constructor depending on the data type.

Q) What do you mean by data encapsulation?

- Data Encapsulation is an Object-Oriented Programming concept of hiding the data attributes and their behaviors in a single unit.
- It helps developers to follow modularity while developing software by ensuring that each object is independent of other objects by having its own methods, attributes, and functionalities.
- It is used for the security of the private properties of an object and hence serves the purpose of data hiding.



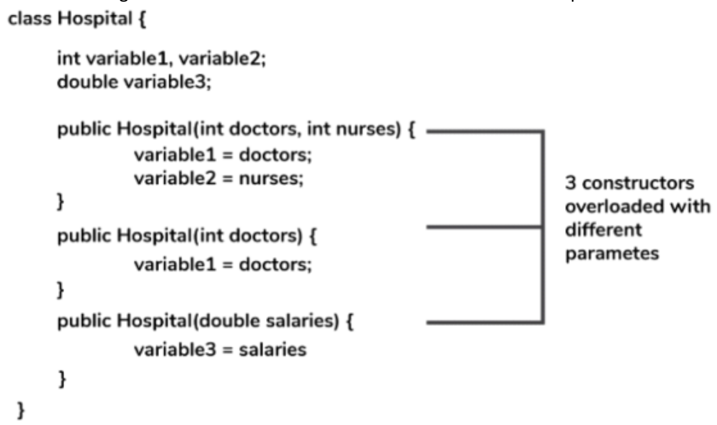
Q) difference between equals() method and equality operator (==) in Java?

equals()	==
This is a method defined in the Object class.	It is a binary operator in Java.
This method is used for checking the equality of contents between two objects as per the specified business logic.	This operator is used for comparing addresses (or references), i.e checks if both the objects are pointing to the same memory location.

- Object class is considered as the parent class of all the java classes. The implementation of the equals method in the Object class uses the == operator to compare two objects. This default implementation can be overridden as per the business logic.

Q) explain the concept of constructor overloading.

- Constructor overloading is the process of creating multiple constructors in the class consisting of the same name with a difference in the constructor parameters



```
class Hospital {

      int variable1, variable2;
      double variable3;

      public Hospital(int doctors, int nurses) {
                variable1 = doctors;
                variable2 = nurses;
      }
      public Hospital(int doctors) {
                variable1 = doctors;
      }
      public Hospital(double salaries) {
                variable3 = salaries
      }
}
```

Q) Define Copy constructor in java.

- Copy Constructor is the constructor used when we want to initialize the value to the new object from the old object of the same class.

```
class InterviewBit{
    String department;
    String service;
    InterviewBit(InterviewBit ib){
        this.departments = ib.departments;
        this.services = ib.services;
    }
}
```

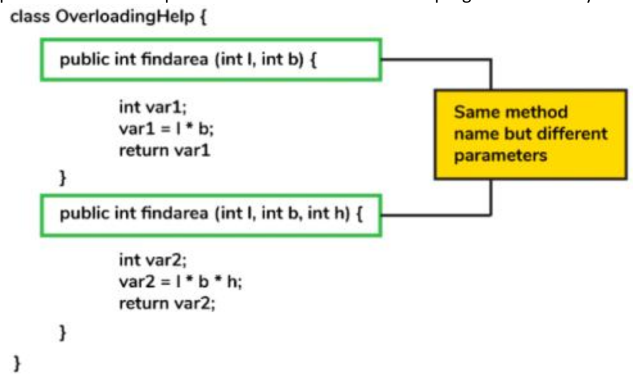
Q) Can the main method be Overloaded?

- Yes, It is possible to overload the main method. We can create as many overloaded main methods we want. However, JVM has a predefined calling method that JVM will only call the main method with the definition of - public static void main(string[] args)

```
class Main {
    public static void main(String args[]) {
        System.out.println(" Main Method");
    }
    public static void main(int[] args){
        System.out.println("Overloaded Integer array Main Method");
    }
    public static void main(char[] args){
        System.out.println("Overloaded Character array Main Method");
    }
    public static int main(double[] args){
        System.out.println("Overloaded Double array Main Method");
    }
    public static void main(float args){
        System.out.println("Overloaded float Main Method");
    }
}
```

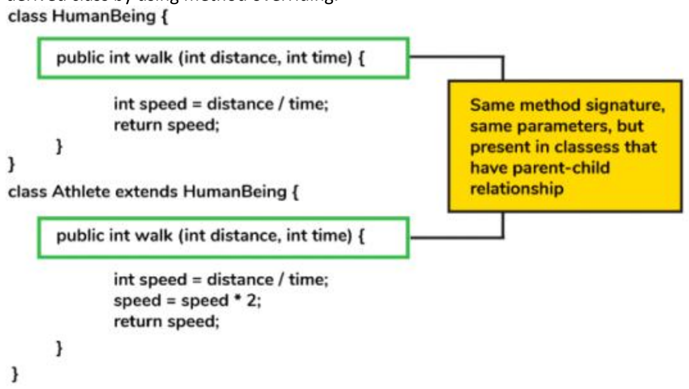
Q) method overloading vs overriding?

- In Java, **method overloading** is made possible by introducing different methods in the same class consisting of the same name. Still, all the functions differ in the number or type of parameters. It takes place inside a class and enhances program readability.



```
class OverloadingHelp {
    public int findarea (int l, int b) {
        int var1;
        var1 = l * b;
        return var1
    }
    public int findarea (int l, int b, int h) {
        int var2;
        var2 = l * b * h;
        return var2;
    }
}
```

- **Method overriding** is the concept in which two methods having the same method signature are present in two different classes in which an inheritance relationship is present.
- A particular method implementation (already present in the base class) is possible for the derived class by using method overriding.



```
class HumanBeing {
    public int walk (int distance, int time) {
        int speed = distance / time;
        return speed;
    }
}
class Athlete extends HumanBeing {
    public int walk (int distance, int time) {
        int speed = distance / time;
        speed = speed * 2;
        return speed;
    }
}
```

Same method signature, same parameters, but present in classess that have parent-child relationship

- Both class methods have the name walk and the same parameters, distance, and time. If the derived class method is called, then the base class method walk gets overridden by that of the derived class.

Q) Explain the use of final keyword in variable, method and class?

- **final variable :**
    - When a variable is declared as final in Java, the value can't be modified once it has been assigned.
    - If any value has not been assigned to that variable, then it can be assigned only by the constructor of the class.
- **final method :**

    - A method declared as final cannot be overridden by its children's classes.
    - A constructor cannot be marked as final because whenever a class is inherited, the constructors are not inherited. Hence, marking it final doesn't make sense. Java throws compilation error saying - modifier final not allowed here
- **final class :**
    - No classes can be inherited from the class declared as final. But that final class can extend other classes for its usage.

Q) Finalize?
- Prior to the garbage collection of an object, the finalize method is called so that the clean-up activity is implemented. Example:
```
public static void main(String[] args) {
String example = new String("InterviewBit");
example = null;
System.gc(); // Garbage collector called
}
public void finalize() {
// Finalize called
}
```

Q) Is it possible that the 'finally' block will not be executed? If yes then list the case.
- Yes. It is possible that the 'finally' block will not be executed. The cases are-
    - Suppose we use System.exit() in the above statement.
    - If there are fatal errors like Stack overflow, Memory access error, etc

Q) When can you use super keyword?
- The super keyword is used to access hidden fields and overridden methods or attributes of the parent class.
- Following are the cases when this keyword can be used:
    - Accessing data members of parent class when the member names of the class and its child subclasses are same.
    - To call the default and parameterized constructor of the parent class inside the child class.
    - Accessing the parent class methods when the child classes have overridden them.
- Examples :
```
public class Parent{
        protected int num = 1;

        Parent(){
            System.out.println("Parent class default constructor.");
        }

        Parent(String x){
            System.out.println("Parent class parameterised constructor.");
        }

        public void foo(){
            System.out.println("Parent class foo!");
        }
    }

    public class Child extends Parent{
        private int num = 2;

        Child(){
            System.out.println("Child class default Constructor");

            super();    // to call default parent constructor
            super("Call Parent");    // to call parameterised constructor.
        }

        void printNum(){
            System.out.println(num);
            System.out.println(super.num); //prints the value of num of parent class
        }

        @Override
        public void foo(){
            System.out.println("Parent class foo!");
            super.foo();    //Calls foo method of Parent class inside the Overriden foo method of Child class.
        }
    }
```

Q) Can the static methods be overloaded?
- Yes! There can be two or more static methods in a class with the same name but differing input parameters.

Q) Why is the main method static in Java?
- The main method is always static because static members are those methods that belong to the classes, not to an individual object. So if the main method will not be static then for every object, It is available. And that is not acceptable by JVM. JVM calls the main method based on the class name itself. Not by creating the object.

Q) Can the static methods be overridden?
- No! Declaration of static methods having the same signature can be done in the subclass but run time polymorphism cannot take place in such cases.
- Overriding or dynamic polymorphism occurs during the runtime, but the static methods are loaded and looked up at the compile time statically. Hence, these methods cant be overridden.

Q) Difference between static methods, static variables, and static classes in java.

- **Static Methods and Static variables** are those methods and variables that belong to the class of the java program, not to the object of the class. This gets memory where the class is loaded. And these can directly be called with the help of class names.
    - For example - We have used mathematical functions in the java program like - max(), min(), sqrt(), pow(), etc. And if we notice that, then we will find that we call it directly with the class name. Like - Math.max(), Math.min(), etc. So that is a static method. And Similarly static variables we have used like (length) for the array to get the length. So that is the static method.
- **Static classes** - A class in the java program cannot be static except if it is the inner class. If it is an inner static class, then it exactly works like other static members of the class

Q) What is the main objective of garbage collection?

- The main objective of this process is to free up the memory space occupied by the unnecessary and unreachable objects during the Java program execution by deleting those unreachable objects.
- **Heap Memory** is cleaned in garbage collection process.

Q) What is a ClassLoader?

- Java Classloader is the program that belongs to JRE (Java Runtime Environment). The task of ClassLoader is to load the required classes and interfaces to the JVM when required.
- Example- To get input from the console, we require the scanner class. And the Scanner class is loaded by the ClassLoader

Q) Shallow Copy?

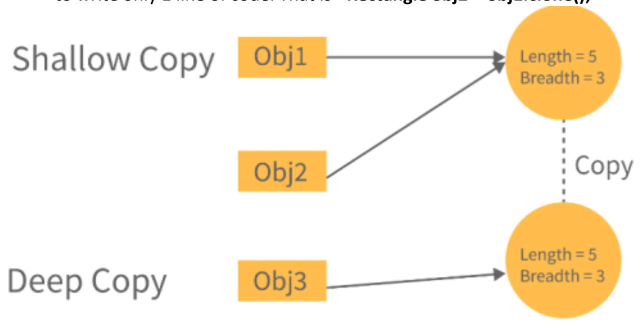
- The shallow copy only creates a new reference and points to the same object.
  ```
  class Rectangle{
  int length = 5;
      int breadth = 3;
  }
  ```

-
  ```
  Rectangle obj1 = new Rectangle();
  Rectangle obj2 = obj1;
  ```
- if we change the values in shallow copy then they affect the other reference as well.

Q) Deep Copy?

- In a deep copy, we create a new object and copy the old object value to the new object.

-
  ```
  Rectangle obj3 = new Rectangle();
  Obj3.length = obj1.length;
  Obj3.breadth = obj1.breadth;
  ```
- if we change the code to deep copy, then there will be no effect on object2 if it is of type deep copy.
- The **clone()** will do this deep copy internally and return a new object. And to do this we need to write only 1 line of code. That is - **Rectangle obj2 = obj1.clone();**


Shallow Copy
Obj1
Obj2
Length = 5
Breadth = 3
Copy
Deep Copy
Obj3
Length = 5
Breadth = 3