1. **What is Java?**

**Ans.** Java is the high-level programming language that was developed by James Gosling in the year 1982.

It is based on the principles of object-oriented programming and can be used to develop large-scale applications.

1. **Why is Java a platform independent language?**

**Ans.** [**Java language**](https://www.interviewbit.com/blog/features-of-java/) was developed in such a way that it does not depend on any hardware or software due to the fact that the [**compiler**](https://www.interviewbit.com/online-java-compiler/) compiles the code and then converts it to platform-independent byte code which can be run on multiple systems.

1. **Can java be said to be the complete object-oriented programming language?**

**Ans.** It is not wrong if we claim that java is the complete object-oriented programming language. Because Everything in Java is under the classes. And we can access that by creating the objects.

But also if we say that java is not a completely object-oriented programming language because it has the support of primitive data types like int, float, char, boolean, double, etc.

Now for the question: **Is java a completely object-oriented programming language?** We can say that - Java is not a pure object-oriented programming language, because it has direct access to primitive data types. And these primitive data types don't directly belong to the Integer classes.

1. **What are Java bytecodes?**

**Ans. j**ava bytecode is **the result of the compilation of a Java program, an intermediate representation of that program which is machine independent**.

The Java bytecode gets processed by the Java virtual machine (JVM) instead of the processor.

1. **How is a java program executed by JVM?**

**Ans.** In Java, programs are not compiled into executable files; **they are compiled into bytecode (as discussed earlier), which the JVM (Java Virtual Machine) then executes at runtime**.

Java source code is compiled into bytecode when we use the javac compiler. The bytecode gets saved on the disk with the file extension

1. **What do you understand by an instance variable and a local variable?**

**Ans. Instance variables** are those variables that are accessible by all the methods in the class.

They are declared outside the methods and inside the class. These variables describe the properties of an object and remain bound to it at any cost.

All the objects of the class will have their copy of the variables for utilization. If any modification is done on these variables, then only that instance will be impacted by it, and all other class instances continue to remain unaffected.

**Example:**

**class** **Athlete** {

**public** String athleteName;

**public** **double** athleteSpeed;

**public** **int** athleteAge;

}

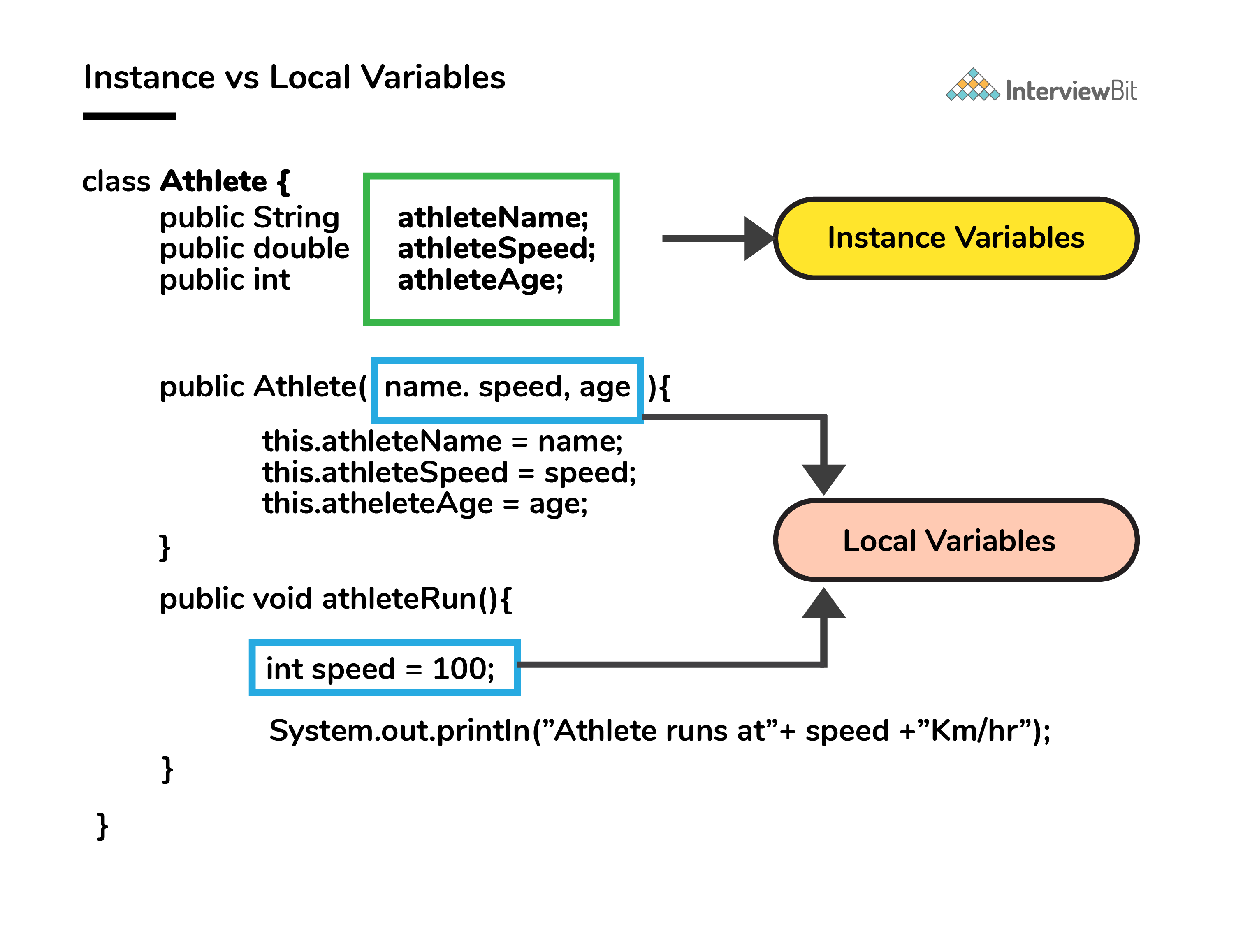
**Local variables** are those variables present within a block, function, or constructor and can be accessed only inside them. The utilization of the variable is restricted to the block scope. Whenever a local variable is declared inside a method, the other class methods don’t have any knowledge about the local variable.

**Example:**

**public** **void** **athlete**() {

String athleteName;

**double** athleteSpeed;



1. **When can you use super keyword?**

**Ans.** 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.
* The following example demonstrates all 3 cases when a super keyword is used.

1. **What is the main objective of garbage collection?**

**Ans.** When Java programs run on the JVM, objects are created on the heap, which is a portion of memory dedicated to the program. Eventually, some objects will no longer be needed. **The garbage collector finds these unused objects and deletes them to free up memory**.

1. **Can the static methods be overridden?**

**Ans.** No! Declaration of static methods having the same signature can be done in the subclass but run time polymorphism can not 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.

1. **How would you differentiate between a String, String Buffer, and a StringBuilder?**

**Ans.** Strings, which are widely used in Java programming, are a sequence of characters. In Java programming language, strings are treated as objects. The Java platform provides the String class to create and manipulate strings.

Whereas, StringBuffer class is a thread-safe, mutable sequence of characters.

* A string buffer is like a String, but can be modified.
* It contains some particular sequence of characters, but the length and content of the sequence can be changed through certain method calls.
* They are safe for use by multiple threads.
* Every string buffer has a capacity.

1. **What do we get in the JDK file? What are the differences between JVM, JRE and JDK in Java?**

* **Ans.** Strings in switch statement.
* Binary integer literals.
* Allowing underscores in numeric literals.
* Catching multiple exception types and rethrowing exceptions with improved type checking.
* Automatic resource management in try -statement

## Difference between JDK, JRE and JVM

Here are the major differences between JDK vs. JRE vs. JVM:

|  |  |  |
| --- | --- | --- |
| **JDK** | **JRE** | **JVM** |
| The full form of JDK is Java Development Kit. | The full form of JRE is Java Runtime Environment. | The full form of JVM is Java Virtual Machine. |
| JDK is a software development kit to develop applications in Java. | It is a software bundle which provides Java class libraries with necessary components to run Java code. | JVM executes Java byte code and provides an environment for executing it. |
| JDK is platform dependent. | JRE is also platform dependent. | JVM is highly platform dependent. |
| It contains tools for developing, debugging, and monitoring java code. | It contains class libraries and other supporting files that JVM requires to execute the program. | Software development tools are not included in JVM. |
| It is the superset of JRE | It is the subset of JDK. | JVM is a subset of JRE. |
| The JDK enables developers to create Java programs that can be executed and run by the JRE and JVM. | The JRE is the part of Java that creates the JVM. | It is the Java platform component that executes source code. |
| JDK comes with the installer. | JRE only contain environment to execute source code. | JVM bundled in both software JDK and JRE. |

1. **What are the differences between constructor and method of a class in Java?**

**Ans. Constructor is used to initialize an object whereas method is used to exhibits functionality of an object**. Constructors are invoked implicitly whereas methods are invoked explicitly. Constructor does not return any value where the method may/may not return a value

1. **What is an array in Java?**

**Ans.** An array is **a collection of elements of the same type placed in contiguous memory locations that can be individually referenced by using an index to a unique identifier**. Five values of type int can be declared as an array without having to declare five different variables (each with its own identifier).

1. **What are the types of an array?**

**Ans. Types of Arrays**

* One dimensional array.
* Multi-dimensional array.

1. **Can we declare array size as negative?**

**Ans. No, you cannot use a negative integer as size**, the size of an array represents the number of elements in it, –ve number of elements in an array makes no sense

**A ={“bootcamp”, “java”, -1, -2}**

1. **What is the difference between Array and Array List?**

**Ans. An array is a fixed-length data structure.** **ArrayList is a variable-length data structure**. It can be resized itself when needed. It is mandatory to provide the size of an array while initializing it directly or indirectly.

1. **What is Inheritance in Java?**

**Ans.** Inheritance is an important pillar of OOP(Object-Oriented Programming). It is **the mechanism in java by which one class is allowed to inherit the features(fields and methods) of another class**

1. **Why do we need to use inheritance?**

**Ans.** Reusability: Inheritance supports the concept of “reusability”, i.e. **when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our new class from the existing class**. By doing this, we are reusing the fields and methods of the existing class

1. **What is super class and subclass?**

**Ans.** Super Class: The class whose features are inherited is known as super class(or a base class or a parent class). Sub Class: The class that inherits the other class is known as subclass(or a derived class, extended class, or child class)

1. **Can we assign superclass to subclass?**

**Ans.** Can I cast a superclass to subclass in Java?

**You can try to convert the super class variable to the sub class type by simply using the cast operator**. But, first of all you need to create the super class reference using the sub class object and then, convert this (super) reference type to sub class type using the cast operator

1. **Can we extend (inherit) final class?**

**Ans. You cannot extend a final class**. If you try it gives you a compile time error

1. **Can a final method be overridden?**

**Ans. No, the Methods that are declared as final cannot be Overridden or hidden**. For this very reason, a method must be declared as final only when we're sure that it is complete.

1. **Can we inherit private members of base class to its subclass?**

**Ans. A subclass does not inherit the private members of its parent class**. However, if the superclass has public or protected methods for accessing its private fields, these can also be used by the subclass. A nested class has access to all the private members of its enclosing class—both fields and methods.

1. **What is order of calling constructors in case of inheritance?**

**Ans.** Order of execution of constructors in inheritance relationship is **from base /parent class to derived / child class**. We know that when we create an object of a class then the constructors get called automatically.

1. **What are the types of inheritance in Java? Briefly explain each and every inheritance in detail?**

**Ans. Types of Inheritance in Java**

* Single Inheritance.
* Multiple Inheritance.(interfaces)
* Multi-Level Inheritance.
* Hierarchical Inheritance.
* Hybrid Inheritance

1. **Why multiple inheritance is not supported in java through class?**

**Ans.** The reason behind this is to prevent ambiguity.

Consider a case where class B extends class A and Class C and both class A and C have the same method display().

Now java compiler cannot decide, which display method it should inherit. To prevent such situation, multiple inheritances is not allowed in java.

Class a class b

**Class c**

1. **How does Multiple inheritance implement in Java?**

**Ans.** The only way to implement multiple inheritance is to **implement multiple interfaces in a class**. In java, one class can implements two or more interfaces. This also does not cause any ambiguity because all methods declared in interfaces are implemented in class.

1. **How many ways to implement relationships among classes in Java?**

**Ans.** In Java, we have **two** types of relationship: Is-A relationship: Whenever one class inherits another class, it is called an IS-A relationship. Has-A relationship: Whenever an instance of one class is used in another class, it is called HAS-A relationship

1. **What is Class in Object-oriented programming?**

**Ans.** In object-oriented programming, a class is **an extensible program-code-template for creating objects, providing initial values for state (member variables) and implementations of behavior (member functions or methods)**.

**Class is a blue print of creating a objects.**

1. **What is Encapsulation or data hiding in Java?**

**Ans.** By definition, encapsulation describes **the idea of bundling data and methods that work on that data within one unit, like a class in Java**. This concept is also often used to hide the internal representation, or state of an object from the outside. This is called information hiding

1. **What is Polymorphism in Java or OOP?**

**Ans.** Polymorphism is **the ability of an object to take on different forms**. In Java, polymorphism refers to the ability of a class to provide different implementations of a method, depending on the type of object that is passed to the method

1. **What is the difference between static and dynamic binding in Java?**

**Ans.** The **static binding uses Type information for binding while Dynamic binding uses Objects to resolve to bind**. Overloaded methods are resolved (deciding which method to be called when there are multiple methods with the same name) using static binding while overridden methods use dynamic binding, i.e, at run time