Java Questions: Definition and real time example and if possible give declarations. Short and clear answer.

1. **What is JVM?**Ans. JVM (Java Virtual Machine) is an abstract machine. It is a specification that provides runtime environment in which java bytecode can be executed. JVMs are available for many hardware and software platforms (i.e. JVM is platform dependent).

**A screenshot of a computer

Description automatically generated**

* **2.Types of Variables**: In Java, there are three types of variables: local, instance, and static.
* **Local Variables**: These are defined within a block or method or constructor. They are created when the block is entered, or the function is called and destroyed after exiting from the block or when the call returns from the function.
* **Instance Variables**: These are non-static and are declared in a class outside any method, constructor, or block. They are created when an object of the class is created and destroyed when the object is destroyed.

**Static Variables**: Also known as Class variables, they are declared similarly as instance variables but are declared using the ‘static’ keyword. They are created when the program starts and destroyed when the program stops.public class Main {

// static variable

static int a = 10; // This is a static variable. It belongs to the class and not to any instance(object).

// instance variable

int b = 20; // This is an instance variable. It belongs to an instance of the class.

void myMethod() {

// local variable

int c = 30; // This is a local variable. Its scope is limited to this method.

System.out.println("Static variable: " + a);

System.out.println("Instance variable: " + b);

System.out.println("Local variable: " + c);

}

public static void main(String[] args) {

Main obj = new Main();

obj.myMethod();

}

}

1. **Non-primitive Data Types**: Non-primitive data types in Java are classes, interfaces, and arrays.
2. **Integer vs int**: Integer is a class in Java that wraps a value of the primitive type int. int is a primitive data type.
3. **Data Types in Java**:
   * byte: 8-bit
   * short: 16-bit
   * int: 32-bit
   * long: 64-bit
   * float: 32-bit
   * double: 64-bit
   * boolean: 1-bit
   * char: 16-bit
4. **float vs double**: float is a single-precision 32-bit IEEE 754 floating point, while double is a double-precision 64-bit IEEE 754 floating point.
5. **Array**: An array is a container object that holds a fixed number of values of a single type.
6. **Array without Size**: No, you cannot declare an array in Java without specifying its size.
7. **Declare Array**: int[] arr = new int[5];
8. **BREAK statement**: The break statement is used to terminate the loop or switch statement and transfer execution to the statement immediately following the loop or switch.
9. **super Keyword**: super is a keyword in Java which is used as a reference to the object of the parent class.
10. **final Keyword**: The final keyword in Java is used to restrict the user from modifying the value, method, or class.
11. **Class**: A class is a blueprint from which individual objects are created.
12. **Base Class**: The base class of all classes in Java is the Object class.
13. **THIS Keyword**: this keyword can be used to refer to the current object in a method or constructor.
14. **Object in Java**: An object is an instance of a class.
15. **Characteristics of an Object**: An object in Java has three characteristics: state, behavior, and identity.
16. **Creating Objects in Java**: You can create objects in Java using the new keyword, new instance() or using methods like clone(), deserialization, or factories.
17. **Static Variable**: A static variable is a variable that belongs to the class and not to object(instance). It is initialized only once at the start of the execution.

**20.Static Block**: A static block in Java is a block of code enclosed in braces {} and preceded by the static keyword. [It is executed only once when the class is loaded into memory1](https://www.geeksforgeeks.org/static-blocks-in-java/). For example, if you have a class MyClass with a static block, the code inside this block will be executed when the class is first loaded.

21.[**Ways to Pass Arguments in Java**: There are two main ways to pass arguments in Java: pass by value and pass by reference](https://www.geeksforgeeks.org/static-blocks-in-java/)[2](https://www.geeksforgeeks.org/parameter-passing-techniques-in-java-with-examples/). Pass by value means that a copy of the variable is made and the method receives that copy. Alterations to the copy do not affect the original variable. Pass by reference means the method receives a reference to the original variable. So, changes to this reference will affect the original variable.

1. [**Packages in Java**: A package in Java is a namespace that organizes a set of related classes and interfaces](https://www.geeksforgeeks.org/static-blocks-in-java/)[3](https://www.geeksforgeeks.org/packages-in-java/). Conceptually you can think of packages as being similar to different folders on your computer.
2. [**Abstraction in Java**: Abstraction is a process of hiding the implementation details and showing only the functionality to the user](https://www.geeksforgeeks.org/static-blocks-in-java/)[4](https://refreshjava.com/java/abstraction-in-java). For example, when you send an email, you just type the text and hit send. You don’t need to understand exactly how the email gets sent.
3. [**Inheritance in Java**: Inheritance in Java is a mechanism where one class acquires the properties (fields and methods) of another](https://www.geeksforgeeks.org/static-blocks-in-java/)[5](https://www.geeksforgeeks.org/inheritance-in-java/). For example, consider a class Animal that has a method eat(). A Dog class could inherit from Animal and it would also have the eat() method.
4. [**Constructor**: A constructor in Java is a block of code similar to a method that’s called when an instance of an object is created](https://www.geeksforgeeks.org/static-blocks-in-java/)[6](https://www.geeksforgeeks.org/constructors-in-java/). For example, if you have a class Bike, the constructor Bike() is called when a new Bike object is created.
5. [**Different Types of Constructors**: There are three types of constructors in Java: default (no-arg), parameterized, and copy](https://www.geeksforgeeks.org/static-blocks-in-java/)[7](https://www.tutorialspoint.com/java/java_constructors.htm).
6. **Default and Parameterized Constructors**: A default constructor is a constructor with no parameters. [If you don’t define a constructor in your class, Java creates one for you](https://www.geeksforgeeks.org/static-blocks-in-java/)[8](https://howtodoinjava.com/java/oops/java-constructors/). [A parameterized constructor has one or more parameters](https://www.geeksforgeeks.org/static-blocks-in-java/)[8](https://howtodoinjava.com/java/oops/java-constructors/).
7. [**Polymorphism in Java**: Polymorphism in Java is a concept by which we can perform a single action in different ways](https://www.geeksforgeeks.org/static-blocks-in-java/)[9](https://stackoverflow.com/questions/28961957/example-of-runtime-polymorphism-in-java). For example, you might have a shape class with a method draw(). If you have subclasses circle and rectangle that both override draw(), the correct version of the method is called based on what type of shape you’re working with.
8. [**Method Overloading**: Method overloading in Java is a feature that allows a class to have more than one method having the same name, if their argument lists are different](https://www.geeksforgeeks.org/static-blocks-in-java/)[10](https://www.geeksforgeeks.org/method-overloading-in-java/). For example, you might have a calculateArea method that calculates the area of a rectangle if given two parameters (width and height), but calculates the area of a square if given one parameter (side length).
9. [**Method Overriding**: Method overriding in Java is a feature that allows a subclass to provide a specific implementation of a method that is already provided by its superclass](https://www.geeksforgeeks.org/static-blocks-in-java/)[9](https://stackoverflow.com/questions/28961957/example-of-runtime-polymorphism-in-java). For example, the Animal class might have a method makeSound(). The Dog class could override this method to return “bark”.
10. [**Abstract Class in Java**: An abstract class in Java is a class that cannot be instantiated and is always used as a base class](https://www.geeksforgeeks.org/static-blocks-in-java/)[4](https://refreshjava.com/java/abstraction-in-java). For example, Animal could be an abstract class with an abstract method makeSound(). You can’t create an Animal, but you can create a Dog that extends Animal and implements makeSound().
11. [**Interface in Java**: An interface in Java is a reference type, similar to a class, that can contain only constants, method signatures, default methods, static methods, and nested types](https://www.geeksforgeeks.org/static-blocks-in-java/)[4](https://refreshjava.com/java/abstraction-in-java). For example, Animal could be an interface with a method makeSound(). The Dog class would implement the Animal interface and define makeSound().
12. [**Implementation Class of Interface**: An implementation class in Java is a class that implements an interface and provides definitions for the methods declared in the interface](https://www.geeksforgeeks.org/static-blocks-in-java/)[4](https://refreshjava.com/java/abstraction-in-java). For example, Dog would be an implementation class of the Animal interface.
13. **Constructor vs Method**: A constructor is a special method used to initialize an object when it’s created. [It has the same name as the class and doesn’t have a return type1](https://www.tutorialspoint.com/difference-between-constructors-and-methods-in-java). [A method is a function defined in a class that describes the behaviors of an object of that class1](https://www.tutorialspoint.com/difference-between-constructors-and-methods-in-java). For example:

**Java**

public class MyClass {

// Constructor

MyClass() {

System.out.println("Constructor invoked");

}

// Method

void myMethod() {

System.out.println("Method invoked");

}

}

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

1. [**Abstract Class vs Interface**: An abstract class is a class that cannot be instantiated and can contain both abstract and non-abstract methods](https://www.tutorialspoint.com/difference-between-constructors-and-methods-in-java)[2](https://www.geeksforgeeks.org/difference-between-abstract-class-and-interface-in-java/). [An interface is a contract that specifies a set of methods that a class must implement](https://www.tutorialspoint.com/difference-between-constructors-and-methods-in-java)[2](https://www.geeksforgeeks.org/difference-between-abstract-class-and-interface-in-java/). For example:

**Java**

public abstract class AbstractClass {

public abstract void abstractMethod();

public void concreteMethod() {

System.out.println("Concrete method in abstract class");

}

}

public interface MyInterface {

void interfaceMethod();

}

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

1. [**Static vs Non-static Methods**: A static method belongs to the class itself and can be called without an instance of the class3](https://www.geeksforgeeks.org/difference-between-static-and-non-static-method-in-java/). [A non-static method belongs to an instance of a class and requires an object of the class to be invoked3](https://www.geeksforgeeks.org/difference-between-static-and-non-static-method-in-java/). For example:

**Java**

public class MyClass {

// Static method

static void staticMethod() {

System.out.println("Static method");

}

// Non-static method

void nonStaticMethod() {

System.out.println("Non-static method");

}

}

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

1. [**Wrapper Class in Java**: A wrapper class in Java provides the mechanism to convert a primitive data type into an object and vice versa](https://www.tutorialspoint.com/difference-between-constructors-and-methods-in-java)[4](https://www.javatpoint.com/wrapper-class-in-java). For example:

**Java**

int primitive = 5;

Integer wrapper = Integer.valueOf(primitive); // Boxing

int backToPrimitive = wrapper.intValue(); // Unboxing

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

1. [**Path and ClassPath in Java**: The Path is an environment variable used by the operating system to locate binary files like java and javac](https://www.tutorialspoint.com/difference-between-constructors-and-methods-in-java)[5](https://www.geeksforgeeks.org/difference-between-path-and-classpath-in-java/). [The ClassPath is a parameter in the JVM or the Java compiler that specifies where to find user-defined classes and packages](https://www.tutorialspoint.com/difference-between-constructors-and-methods-in-java)[5](https://www.geeksforgeeks.org/difference-between-path-and-classpath-in-java/).
2. [**Type Casting in Java**: Type casting in Java is a method to convert a value from one data type to another](https://www.tutorialspoint.com/difference-between-constructors-and-methods-in-java)[6](https://www.javatpoint.com/type-casting-in-java). For example:

**Java**

double doubleNumber = 9.78;

int intNumber = (int) doubleNumber; // Type casting double to int

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

1. **Access Modifiers for Inner Classes**: Inner classes in Java can be declared as public, private, protected, or package-private (default). [The access modifier determines the visibility of the inner class from other classes and packages](https://www.tutorialspoint.com/difference-between-constructors-and-methods-in-java)[7](https://www.javatpoint.com/access-modifiers).
2. [**Difference between Error and Exception**: An Error indicates serious problems that a reasonable application should not try to catch, such as OutOfMemoryError8](https://www.geeksforgeeks.org/errors-v-s-exceptions-in-java/). [An Exception is an event that occurs during the execution of a program that disrupts the normal flow of instructions](https://www.tutorialspoint.com/difference-between-constructors-and-methods-in-java)[8](https://www.geeksforgeeks.org/errors-v-s-exceptions-in-java/).
3. **Types of Exceptions**: There are two types of exceptions in Java: checked and unchecked. [Checked exceptions are exceptions that need to be declared in a method or constructor’s throws clause if they can be thrown by the execution of the method or constructor](https://www.tutorialspoint.com/difference-between-constructors-and-methods-in-java)[9](https://www.geeksforgeeks.org/types-of-exception-in-java-with-examples/). [Unchecked exceptions are exceptions that do not need to be declared in a method or constructor’s throws clause](https://www.tutorialspoint.com/difference-between-constructors-and-methods-in-java)[9](https://www.geeksforgeeks.org/types-of-exception-in-java-with-examples/).
4. [**Throw Keyword**: The throw keyword in Java is used to explicitly throw an exception](https://www.tutorialspoint.com/difference-between-constructors-and-methods-in-java)[10](https://www.programiz.com/java-programming/throw-throws). For example:

**Java**

throw new ArithmeticException("Division by zero!");

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

1. [**Throws Keyword**: The throws keyword in Java is used in the signature of method to indicate that this method might throw one of the listed type exceptions](https://www.tutorialspoint.com/difference-between-constructors-and-methods-in-java)[10](https://www.programiz.com/java-programming/throw-throws).
2. [**Difference between Throw and Throws**: The throw keyword is used within a method to throw a single exception, whereas the throws keyword is used in the method signature to declare the exceptions that can be thrown by the method](https://www.tutorialspoint.com/difference-between-constructors-and-methods-in-java)[11](https://howtodoinjava.com/java/exception-handling/throw-vs-throws/).
3. [**Finally Block**: The finally block in Java is used to put important code that must be executed whether an exception is handled or not](https://www.tutorialspoint.com/difference-between-constructors-and-methods-in-java)[9](https://www.geeksforgeeks.org/types-of-exception-in-java-with-examples/).
4. **Handling More Than One Exception**: Yes, you can handle more than one type of exception in a single catch block by using pipe (|) to separate them. For example:

**Java**

try {

// Some code here

} catch (IOException | SQLException ex) {

// Exception handler

}

AI-generated code. Review and use carefully. [More info on FAQ](https://www.bing.com/new#faq).

1. [**ClassNotFoundException vs NoClassDefFoundError**: ClassNotFoundException is an exception that occurs when you try to load a class at run time using Class.forName() or loadClass() methods and the required classes are not found in the classpath](https://www.tutorialspoint.com/difference-between-constructors-and-methods-in-java)[8](https://www.geeksforgeeks.org/errors-v-s-exceptions-in-java/)[NoClassDefFoundError is an error that occurs when the JVM or a ClassLoader instance tries to load a class definition and the class definition is no longer available](https://www.tutorialspoint.com/difference-between-constructors-and-methods-in-java)[8](https://www.geeksforgeeks.org/errors-v-s-exceptions-in-java/).
2. [**Input Mismatch Exception**: InputMismatchException is a java.util exception that is thrown by a Scanner class when a user enters a value of an unexpected data type](https://www.tutorialspoint.com/difference-between-constructors-and-methods-in-java)[9](https://www.geeksforgeeks.org/types-of-exception-in-java-with-examples/).
3. **Collection Interface**: The Collection interface in Java is used to represent a group of objects as a single unit. [It is the root interface of the Java Collections Framework, which includes lists, sets, queues, etc](https://www.tutorialspoint.com/difference-between-constructors-and-methods-in-java)
4. **What are the important Methods in Collections Class?**
   1. Important methods in the Collections class include sort(), reverse(), copy(), min(), max(), etc.
5. **What are the Important Methods in Collection Interface?**
   1. Important methods in the Collection interface include add(), addAll(), remove(), removeAll(), clear(), size(), and contains().
6. **What is the difference between Collections & Collection?**
   1. Collection is an interface representing a group of objects known as its elements. Collections is a utility class in Java containing various static methods for manipulating data in a collection.
7. **What is List?**
   1. List is an ordered collection that allows us to store and access elements sequentially.
8. **What is Map?**
   1. Map is a collection interface that provides a way to store data in a key-value format.
9. **What is Set?**
   1. Set is an interface that represents an unordered collection of objects in which duplicate values cannot be stored.
10. **Why ArrayList is better than Arrays?**
    1. ArrayList is better than arrays because it is dynamic, can grow as needed, and provides more functionality than arrays.
11. **What is the difference between ArrayList and LinkedList?**
    1. ArrayList uses a dynamic array to store elements, while LinkedList uses a doubly linked list. ArrayList is better for storing and accessing data, while LinkedList is better for manipulating data.
12. **Which one is most preferred for addition and deletion? ArrayList or LinkedList?**
    1. LinkedList is preferred for addition and deletion because it provides constant time for these operations.
13. **Which one is most preferred For searches? ArrayList or LinkedList?**
    1. ArrayList is preferred for searches because it provides constant time for search operation.
14. **What is the difference between Iterator and Enumeration?**
    1. Both Iterator and Enumeration are interfaces in Java used to traverse through the elements of a collection. The main difference is that Iterator has a remove() method while Enumeration does not.
15. **What are the type of inheritance? Which type of inheritance does not support in java?**
    1. There are four types of inheritance in Java: single, multilevel, hierarchical, and hybrid. However, Java does not support multiple inheritance with classes due to the “Diamond Problem”. It does, however, support multiple inheritance with interfaces.
16. What is Maven?
17. What is Jira? Is Jira is bug tracking tool? What is scrumb? What is Azile? Explain.