1. What is Java?Java is a high-level, object-oriented programming language developed for creating secure, portable, and platform-independent applications. It was designed to follow the principle “Write Once, Run Anywhere.” Java programs run on the Java Virtual Machine (JVM), which makes them platform independent. Java is widely used for web applications, desktop software, mobile applications, and enterprise systems. It supports features like inheritance, polymorphism, multithreading, and automatic memory management through garbage collection.

2. Who developed Java?Java was developed by James Gosling and his team at Sun Microsystems in 1995. Initially, the language was called Oak because of an oak tree outside Gosling’s office. Later, it was renamed Java. The main goal of Java development was to create a portable and platform-independent programming language for electronic devices. After Sun Microsystems, Java was acquired and maintained by Oracle Corporation.

3. Define OOPs. Object-Oriented Programming System (OOPs) is a programming approach based on objects and classes. It helps in organizing programs into reusable and modular units. OOPs focuses on real-world concepts such as objects, inheritance, encapsulation, abstraction, and polymorphism. In Java, OOPs improves code reusability, security, and maintainability. Programs become easier to understand and modify. Objects communicate with each other using methods, making software development more structured and efficient for large-scale applications.

4. What are the features of OOPs?The main features of OOPs are encapsulation, inheritance, polymorphism, and abstraction. Encapsulation hides data and protects it from unauthorized access. Inheritance allows one class to acquire properties of another class. Polymorphism enables methods to perform different tasks using the same name. Abstraction hides implementation details and shows only essential features. OOPs also supports code reusability, modularity, data security, and easy maintenance, making programming more efficient and flexible.

5. What is platform independence in Java?Platform independence means a Java program can run on any operating system without modification. Java source code is converted into bytecode by the compiler. This bytecode runs on the Java Virtual Machine (JVM), which is available for different platforms like Windows, Linux, and macOS. Therefore, Java follows the principle “Write Once, Run Anywhere.” Platform independence makes Java highly portable and suitable for developing applications that can run on multiple systems easily.

6. Differentiate JDK, JRE, and JVM.JDK (Java Development Kit) is a complete package used for developing Java applications. It includes JRE and development tools like compiler and debugger. JRE (Java Runtime Environment) provides libraries and files required to run Java programs. JVM (Java Virtual Machine) is a virtual machine that executes Java bytecode. JVM makes Java platform independent. In short, JDK is for development, JRE is for execution, and JVM is responsible for running Java programs.

1. Define class in Java.A class in Java is a blueprint or template used to create objects. It contains variables called data members and functions called methods that define the behavior of objects. A class helps in implementing Object-Oriented Programming concepts such as encapsulation and inheritance. It does not occupy memory until objects are created. For example, a Student class may contain student name, roll number, and methods like display() and calculateMarks().

2. Define object in Java. An object in Java is an instance of a class. It represents a real-world entity and contains data and behavior. Objects are created using the new keyword. Each object has its own memory and can access class methods and variables. For example, if Student is a class, then s1 and s2 can be objects of that class. Objects help in implementing Object-Oriented Programming by allowing data hiding, reusability, and modular programming.

3. What is a constructor?A constructor in Java is a special method used to initialize objects. It has the same name as the class and does not have any return type. Constructors are automatically called when an object is created. They help in assigning initial values to variables. Java provides default constructors if no constructor is written. Constructors improve code readability and simplify object initialization. Constructors can also be overloaded to initialize objects in different ways.

4. Differentiate constructor and method.A constructor is used to initialize objects, while a method is used to perform operations or actions. Constructors have the same name as the class, but methods can have any valid name. Constructors do not have a return type, whereas methods may return values. Constructors are automatically called during object creation, while methods must be called explicitly. A constructor runs only once during object creation, but methods can be called many times.

5. What is method overloading?Method overloading is a feature in Java where multiple methods have the same name but different parameter lists. The difference may be in the number, type, or order of parameters. It is an example of compile-time polymorphism. Method overloading improves readability and allows the same method to perform different tasks. For example, an add() method may add integers or floating-point numbers depending on the arguments passed during method calling.

6. What is constructor overloading?Constructor overloading means defining multiple constructors in the same class with different parameter lists. Each constructor initializes objects in different ways. It allows flexibility while creating objects. One constructor may take no arguments, while another may accept parameters like name and age. Constructor overloading is an example of compile-time polymorphism. It improves code reusability and simplifies object creation by allowing different initialization options within the same class.

1. Define inheritance. Inheritance is an Object-Oriented Programming feature in Java where one class acquires the properties and methods of another class. The existing class is called the superclass or parent class, while the new class is called the subclass or child class. Inheritance promotes code reusability and reduces duplication. It is implemented using the extends keyword in Java. Inheritance also supports method overriding and polymorphism, making programs more organized, flexible, and easier to maintain.

2. What are the types of inheritance in Java?Java supports several types of inheritance. These include single inheritance, multilevel inheritance, and hierarchical inheritance. Single inheritance means one child class inherits one parent class. Multilevel inheritance occurs when a class inherits another derived class. Hierarchical inheritance means multiple child classes inherit the same parent class. Java does not support multiple inheritance through classes to avoid ambiguity problems, but it can be achieved using interfaces. Inheritance improves code reusability and flexibility.

3. What is polymorphism?Polymorphism is an important feature of Object-Oriented Programming where one method or object can perform different tasks. The word polymorphism means “many forms.” In Java, polymorphism is of two types: compile-time polymorphism and runtime polymorphism. Method overloading is an example of compile-time polymorphism, while method overriding is an example of runtime polymorphism. Polymorphism increases flexibility, improves code reusability, and allows the same interface to represent different implementations in Java programs.

4. What is dynamic method dispatch?Dynamic method dispatch is a mechanism in Java through which a call to an overridden method is resolved at runtime rather than compile time. It is achieved using inheritance and method overriding. A superclass reference variable can refer to a subclass object, and the method of the subclass gets executed. Dynamic method dispatch supports runtime polymorphism. It makes Java programs more flexible and allows methods to behave differently depending on the object being referenced.

5. Define interface in Java. An interface in Java is a collection of abstract methods and constants used to achieve abstraction and multiple inheritance. It is declared using the interface keyword. Methods inside an interface are public and abstract by default. A class implements an interface using the implements keyword. Interfaces provide a common structure that different classes can follow. They improve flexibility, code reusability, and standardization in Java programming by separating implementation from design.

6. Difference between abstract class and interface. An abstract class can contain both abstract and non-abstract methods, while an interface mainly contains abstract methods. A class can inherit only one abstract class but can implement multiple interfaces. Abstract classes are declared using the abstract keyword, whereas interfaces use the interface keyword. Abstract classes may contain constructors and instance variables, but interfaces cannot have constructors. Interfaces are mainly used for complete abstraction and multiple inheritance in Java programming

1. What is an exception in Java?An exception in Java is an abnormal event that occurs during program execution and disrupts the normal flow of the program. Exceptions usually happen because of errors such as division by zero, invalid file access, or array index out of bounds. Java provides a mechanism called exception handling to manage such situations. Exceptions are represented as objects in Java. Proper exception handling prevents program termination and improves the reliability and stability of applications.

2. Differentiate Error and Exception. Errors and exceptions both represent abnormal conditions in Java, but they differ in nature. Errors are serious problems caused by the system, such as memory failure, and are generally not recoverable. Exceptions are conditions that can be handled by the programmer using exception handling techniques. Errors belong to the Error class, while exceptions belong to the Exception class. Exceptions allow programs to continue execution safely, whereas errors usually terminate program execution completely.

3. What is exception handling?Exception handling in Java is a mechanism used to detect, manage, and handle runtime errors so that program execution can continue normally. It uses keywords like try, catch, finally, throw, and throws. Exception handling prevents sudden program termination and improves software reliability. When an exception occurs, Java transfers control to the appropriate catch block. It helps programmers identify errors, display meaningful messages, and maintain smooth execution of applications efficiently.

4. What is the use of try block?The try block in Java is used to enclose code that may generate exceptions during program execution. It allows Java to monitor the enclosed statements for runtime errors. A try block must be followed by at least one catch block or a finally block. If an exception occurs inside the try block, control immediately transfers to the matching catch block. The try block helps in implementing safe and controlled exception handling in Java applications.

5. What is catch block?A catch block in Java is used to handle exceptions generated inside the try block. It contains code that executes when a specific exception occurs. The catch block prevents abnormal program termination by handling errors properly. Multiple catch blocks can be used with a single try block to handle different exceptions separately. The catch block receives the exception object as a parameter, which helps in identifying and displaying information about the error.

6. What is finally block?The finally block in Java contains code that always executes whether an exception occurs or not. It is generally used for cleanup activities such as closing files, releasing resources, or disconnecting database connections. The finally block is placed after try and catch blocks. Even if an exception is not handled, the finally block usually executes before program termination. It ensures important operations are completed and improves resource management in Java applications.

1. What is a thread?A thread is the smallest unit of execution within a process. It allows a program to perform multiple tasks simultaneously. Threads share the same memory space and resources of a process, making execution faster and more efficient. In Java, threads are used for multitasking operations such as animations, file downloading, and background processing. Each thread executes independently, improving program performance and responsiveness, especially in applications that require concurrent task execution.

2. Define multithreading. Multithreading is a process in which multiple threads execute simultaneously within a single program. It helps perform several tasks at the same time, improving CPU utilization and application performance. Java provides built-in support for multithreading through the Thread class and Runnable interface. Multithreading is useful in applications like web servers, games, and multimedia software. It reduces execution time, increases responsiveness, and allows efficient use of system resources in Java applications.

3. Difference between process and thread.A process is an independent program in execution with its own memory and resources, while a thread is a smaller execution unit within a process. Processes are heavyweight and require more memory, whereas threads are lightweight and share process resources. Communication between processes is slower compared to threads. Multiple threads can exist inside a single process. Threads improve efficiency and multitasking, while processes provide better isolation and security between running applications.

4. What are thread states?A thread in Java passes through different states during execution. The main thread states are New, Runnable, Running, Blocked/Waiting, and Terminated. A thread is in the New state when created, Runnable when ready to run, Running when executing, Waiting or Blocked when paused temporarily, and Terminated after completion. These states help Java manage thread execution efficiently. Understanding thread states is important for synchronization and controlling multithreaded program behavior properly.

5. How is a thread created in Java?Threads in Java can be created in two ways: by extending the Thread class or by implementing the Runnable interface. After creating the thread object, the start() method is called to begin execution. The start() method internally invokes the run() method. Using Runnable is preferred because it supports multiple inheritance through interfaces. Thread creation allows programs to perform multiple tasks concurrently and improves application performance and responsiveness in Java.

6. What is thread priority?Thread priority in Java determines the importance of a thread and influences the order in which threads are scheduled for execution. Java provides priorities from 1 to 10. The default priority is 5. Higher-priority threads are generally executed before lower-priority threads. Thread priority is set using the setPriority() method. However, execution also depends on the operating system scheduler. Proper priority management helps improve performance and efficient execution of important tasks in multithreaded programs.

1. What is a file in Java?A file in Java is a collection of related data stored permanently on a storage device such as a hard disk. Java provides the File class in the java.io package to create, read, write, rename, and delete files. Files help store program data permanently even after program execution ends. Java supports text and binary files. File handling is important for applications that need data storage, retrieval, and processing in an organized manner.

2. Define stream in Java.A stream in Java is a flow of data used for input and output operations. It transfers data between a program and external devices such as files, keyboards, or networks. Java streams are mainly divided into input streams and output streams. Input streams read data, while output streams write data. Streams provide efficient data handling and support file processing. Java offers byte streams and character streams for handling different types of data effectively.

3. Differentiate byte stream and character stream. Byte streams handle data in the form of bytes and are mainly used for binary files like images and audio files. Character streams handle data in the form of characters and are used for text files. Byte streams use InputStream and OutputStream classes, while character streams use Reader and Writer classes. Character streams support Unicode and are more suitable for text processing. Both streams are important for input and output operations in Java.

4. What is serialization?Serialization in Java is the process of converting an object into a byte stream so that it can be stored in a file or transferred over a network. It is achieved using the Serializable interface. Serialized objects can later be restored using deserialization. Serialization helps in saving object states permanently and supports distributed computing. It is widely used in networking, file handling, and database operations where object data needs to be preserved or transmitted.

5. What is Collection Framework?The Collection Framework in Java is a set of classes and interfaces used to store, manage, and manipulate groups of objects efficiently. It provides dynamic data structures such as lists, sets, queues, and maps. The framework includes interfaces like List, Set, and Queue, along with classes like ArrayList and HashSet. It simplifies programming by providing reusable algorithms and methods for searching, sorting, insertion, and deletion operations in Java applications.

6. What is List interface?The List interface in Java is part of the Collection Framework and is used to store ordered collections of elements. It allows duplicate elements and maintains insertion order. Elements can be accessed using indexes. Common classes implementing the List interface are ArrayList, LinkedList, and Vector. The List interface provides methods for adding, removing, searching, and updating elements. It is useful when ordered data storage and retrieval are required in Java applications.