1. **What is Java?**
   - Java is a high-level, object-oriented programming language developed by Sun Microsystems, which is now owned by Oracle Corporation.

2. **What is the difference between JDK, JRE, and JVM?**
   - JDK (Java Development Kit) is a software development kit used for developing Java applications. JRE (Java Runtime Environment) is a runtime environment required to run Java applications. JVM (Java Virtual Machine) is an abstract machine that provides a runtime environment in which Java bytecode can be executed.

3. **Explain the main features of Java.**
   - Java is platform-independent, object-oriented, secure, portable, robust, multithreaded, and dynamic.

4. **What are the primitive data types in Java?**
   - The primitive data types in Java are byte, short, int, long, float, double, char, and boolean.

5. **What is a class in Java?**
   - A class in Java is a blueprint for creating objects. It defines the properties and behaviors of objects.

6. **What is an object in Java?**
   - An object in Java is an instance of a class. It has state and behavior.

7. **Explain inheritance in Java.**
   - Inheritance is a mechanism in Java by which one class can inherit properties and behaviors from another class.

8. **What is method overloading?**
   - Method overloading is a feature of Java that allows a class to have multiple methods with the same name but different parameters.

9. **What is method overriding?**
   - Method overriding is a feature of Java that allows a subclass to provide a specific implementation of a method that is already defined in its superclass.

10. **What is polymorphism?**
    - Polymorphism is a feature of Java that allows objects of different classes to be treated as objects of a common superclass.

11. **Explain encapsulation in Java.**
    - Encapsulation is a mechanism in Java that binds data and methods that manipulate the data together as a single unit.

12. **What is an interface?**

- An interface in Java is a reference type that can contain only abstract methods, default methods, static methods, and constant variables.

13. **What is an abstract class?**
    - An abstract class in Java is a class that cannot be instantiated and may contain abstract methods.

14. **What is a constructor in Java?**
    - A constructor in Java is a special type of method that is used to initialize objects.

15. **What is the difference between a constructor and a method?**
    - A constructor is used to initialize objects, while a method is used to perform some action.

16. **Explain the access modifiers in Java.**
    - Access modifiers in Java control the visibility of classes, variables, methods, and constructors.

17. **What is the default access modifier in Java?**
    - The default access modifier in Java is package-private, which means that the member is accessible only within its own package.

18. **What is the difference between `==` and `.equals()` method in Java?**
    - The `==` operator is used to compare references, while the `.equals()` method is used to compare the contents of objects.

19. **What is the `final` keyword in Java?**
    - The `final` keyword in Java is used to restrict the user from changing the value of a variable, from overriding a method, or from inheriting from a class.

20. **What is the difference between `final`, `finally`, and `finalize` in Java?**
    - `final` is a keyword used to apply restrictions on class, method, and variable. `finally` is a block used to execute important code such as closing a connection, whether an exception is thrown or not. `finalize` is a method used to perform cleanup operations on an object before it is garbage collected.

21. **What is a package in Java?**
    - A package in Java is a group of related classes and interfaces.

22. **What is a static variable in Java?**
    - A static variable in Java is a variable that belongs to the class and not to any instance of the class.

23. **What is a static method in Java?**
    - A static method in Java is a method that belongs to the class and not to any instance of the class.

24. **What is method chaining in Java?**
    - Method chaining in Java is a technique where multiple method calls are chained together in a single statement.

25. **Explain the `this` keyword in Java.**
   - The `this` keyword in Java refers to the current instance of the class.

26. **What is the purpose of the `super` keyword in Java?**
   - The `super` keyword in Java is used to refer to the superclass of the current object.

27. **What is method hiding in Java?**
   - Method hiding in Java occurs when a static method in a subclass has the same signature as a static method in the superclass.

28. **What is a constructor chaining in Java?**
   - Constructor chaining in Java is a process of calling one constructor from another constructor of the same class or from the constructor of the superclass.

29. **What is the `instanceof` operator in Java?**
   - The `instanceof` operator in Java is used to test whether an object is an instance of a particular class or interface.

30. **What are wrapper classes in Java?**
   - Wrapper classes in Java are classes that encapsulate primitive data types.

31. **What is autoboxing and unboxing in Java?**
   - Autoboxing is the process of converting a primitive type into its corresponding wrapper class object. Unboxing is the process of converting a wrapper class object into its corresponding primitive type.

32. **What is a thread in Java?**
   - A thread in Java is a lightweight process that executes a sequence of instructions independently of other threads.

33. **Explain the difference between `Thread` and `Runnable` in Java.**
   - A `Thread` is a class in Java that represents a single thread of execution. A `Runnable` is an interface in Java that defines a single method called `run()`, which contains the code that will be executed by a thread.

34. **How do you create a thread in Java?**
   - You can create a thread in Java by extending the `Thread` class or by implementing the `Runnable` interface.

35. **What is synchronization in Java?**
   - Synchronization in Java is the process of controlling the access of multiple threads to shared resources.

36. **What is the `synchronized` keyword in Java?**

- The `synchronized` keyword in Java is used to create synchronized blocks of code or synchronized methods.

37. **What is the purpose of the `volatile` keyword in Java?**
- The `volatile` keyword in Java is used to indicate that a variable's value will be modified by different threads.

38. **What is deadlock in Java?**
- Deadlock in Java occurs when two or more threads are blocked forever, waiting for each other to release resources.

39. **What is the `wait()`, `notify()`, and `notifyAll()` methods in Java?**
- `wait()`: This method is used to make a thread wait until another thread invokes the `notify()` or `notifyAll()` method for this object.
- `notify()`: This method is used to wake up a single thread that is waiting on this object's monitor.
- `notifyAll()`: This method is used to wake up all threads that are waiting on this object's monitor.

40. **What is the difference between `wait()` and `sleep()` methods in Java?**
- The `wait()` method is used to make a thread wait for a signal from another thread, while the `sleep()` method is used to make a thread pause for a specified amount of time.

41. **What is a deadlock in Java?**
- Deadlock in Java occurs when two or more threads are blocked forever, waiting for each other to release resources.

42. **What is the purpose of the `volatile` keyword in Java?**
- The `volatile` keyword in Java is used to indicate that a variable's value will be modified by different threads.

43. **What is the difference between `synchronized` block and `synchronized` method in Java?**
- A `synchronized` block allows only one thread to execute the code block at a time for a particular object, while a `synchronized` method is a method that is synchronized, meaning only one thread can execute it at a time for a particular instance of the class.

44. **What is the `Executor` framework in Java?**
- The `Executor` framework in Java provides a way to decouple task submission from task execution.

45. **Explain the `Callable` and `Future` interfaces in Java.**
- The `Callable` interface in Java is similar to the `Runnable` interface, but it can return a result and throw a checked exception. The `Future` interface represents the result of an asynchronous computation.

46. **What is the purpose of the `java.util.concurrent` package in Java?**
- The `java.util.concurrent` package in Java provides a set of high-level concurrency utilities.

47. **What is the `Lock` interface in Java?**
  - The `Lock` interface in Java provides a way to control access to a shared resource by multiple threads.

48. **What is the `ReentrantLock` class in Java?**
  - The `ReentrantLock` class in Java implements the `Lock` interface and provides a reentrant mutual exclusion lock.

49. **What is the `ReadWriteLock` interface in Java?**
  - The `ReadWriteLock` interface in Java provides a way to have multiple threads read a resource concurrently while exclusive access is granted to a single thread for writing.

50. **What is the `Semaphore` class in Java?**
  - The `Semaphore` class in Java is used to control access to a shared resource by a fixed number of threads.

51. **What is the `CountDownLatch` class in Java?**
  - The `CountDownLatch` class in Java is used to make a thread wait until a fixed number of threads have completed their tasks.

52. **What is the `CyclicBarrier` class in Java?**
  - The `CyclicBarrier` class in Java is used to make a group of threads wait at a barrier until all threads have reached it before proceeding.

53. **What is the `BlockingQueue` interface in Java?**
  - The `BlockingQueue` interface in Java is a type of queue that supports blocking operations.

54. **What is the `LinkedBlockingQueue` class in Java?**
  - The `LinkedBlockingQueue` class in Java is an implementation of the `BlockingQueue` interface that uses a linked list to store elements.

55. **What is the `ArrayBlockingQueue` class in Java?**
  - The `ArrayBlockingQueue` class in Java is an implementation of the `BlockingQueue` interface that uses an array to store elements.

56. **What is the `ConcurrentHashMap` class in Java?**
  - The `ConcurrentHashMap` class in Java is a thread-safe implementation of the `Map` interface.

57. **What is the `CopyOnWriteArrayList` class in Java?**
  - The `CopyOnWriteArrayList` class in Java is a thread-safe implementation of the `List` interface.

58. **What is the purpose of the `java.util.concurrent.atomic` package in Java?**
  - The `java.util.concurrent.atomic` package in Java provides classes that support lock-free thread-safe programming on single variables.

59. **What is the `AtomicInteger` class in Java?**
  - The `AtomicInteger` class in Java is a class that provides atomic operations on integer values.

60. **What is the `AtomicLong` class in Java?**
  - The `AtomicLong` class in Java is a class that provides atomic operations on long values.

61. **What is the purpose of the `java.util.concurrent.locks` package in Java?**
  - The `java.util.concurrent.locks` package in Java provides a framework for locking and waiting for conditions that is distinct from the built-in synchronization support.

62. **What is the `ReadWriteLock` interface in Java?**
  - The `ReadWriteLock` interface in Java provides a way to have multiple threads read a resource concurrently while exclusive access is granted to a single thread for writing.

63. **What is the `ReentrantReadWriteLock` class in Java?**
  - The `ReentrantReadWriteLock` class in Java implements the `ReadWriteLock` interface and provides reentrant read-write locks.

64. **What is the purpose of the `java.util.concurrent.atomic` package in Java?**
  - The `java.util.concurrent.atomic` package in Java provides classes that support lock-free thread-safe programming on single variables.

65. **What is the `AtomicInteger` class in Java?**
  - The `AtomicInteger` class in Java is a class that provides atomic operations on integer values.

66. **What is the `AtomicLong` class in Java?**
  - The `AtomicLong` class in Java is a class that provides atomic operations on long values.

67. **What is the purpose of the `java.util.concurrent.locks` package in Java?**
  - The `java.util.concurrent.locks` package in Java provides a framework for locking and waiting for conditions that is distinct from the built-in synchronization support.

68. **What is the `Lock` interface in Java?**
  - The `Lock` interface in Java provides a way to control access to a shared resource by multiple threads.

69. **What is the `ReentrantLock` class in Java?**
  - The `ReentrantLock` class in Java implements the `Lock` interface and provides a reentrant mutual exclusion lock.

70. **What is the `Condition` interface in Java?**
  - The `Condition` interface in Java provides a way for threads to suspend execution until a particular condition is true.

71. **What is the purpose of the `java.util.concurrent.atomic` package in Java?**
  - The `java.util.concurrent.atomic` package in Java provides classes that support lock-free thread-safe programming on single variables.

72. **What is the `AtomicInteger` class in Java?**
  - The `AtomicInteger` class in Java is a class that provides atomic operations on integer values.

73. **What is the `AtomicLong` class in Java?**
  - The `AtomicLong` class in Java is a class that provides atomic operations on long values.

74. **What is the `AtomicReference` class in Java?**
  - The `AtomicReference` class in Java is a class that provides atomic operations on object references.

75. **What is the purpose of the `java.util.concurrent.locks` package in Java?**
  - The `java.util.concurrent.locks` package in Java provides a framework for locking and waiting for conditions that is distinct from the built-in synchronization support.

76. **What is the `Lock` interface in Java?**
  - The `Lock` interface in Java provides a way to control access to a shared resource by multiple threads.

77. **What is the `ReentrantLock` class in Java?**
  - The `ReentrantLock` class in Java implements the `Lock` interface and provides a reentrant mutual exclusion lock.

78. **What is the `ReadWriteLock` interface in Java?**
  - The `ReadWriteLock` interface in Java provides a way to have multiple threads read a resource concurrently while exclusive access is granted to a single thread for writing.

79. **What is the `ReentrantReadWriteLock` class in Java?**
  - The `ReentrantReadWriteLock` class in Java implements the `ReadWriteLock` interface and provides reentrant read-write locks.

80. **What is the `Condition` interface in Java?**
  - The `Condition` interface in Java provides a way for threads to suspend execution until a particular condition is true.

81. **What is the `Semaphore` class in Java?**
  - The `Semaphore` class in Java is used to control access to a shared resource by a fixed number of threads.

82. **What is the `CountDownLatch` class in Java?**
  - The `CountDownLatch` class in Java is used to make a thread wait until a fixed number of threads have completed their tasks.

83. **What is the `CyclicBarrier` class in Java?**
  - The `CyclicBarrier` class in Java is used to make a group of threads wait at a barrier until all threads have reached it before proceeding.

84. **What is the `BlockingQueue` interface in Java?**
  - The `BlockingQueue` interface in Java is a type of queue that supports blocking operations.

85. **What is the `LinkedBlockingQueue` class in Java?**
  - The `LinkedBlockingQueue` class in Java is an implementation of the `BlockingQueue` interface that uses a linked list to store elements.

86. **What is the `ArrayBlockingQueue` class in Java?**
  - The `ArrayBlockingQueue` class in Java is an implementation of the `BlockingQueue` interface that uses an array to store elements.

87. **What is the `PriorityBlockingQueue` class in Java?**
  - The `PriorityBlockingQueue` class in Java is an unbounded blocking queue that uses a priority heap.

88. **What is the `DelayQueue` class in Java?**
  - The `DelayQueue` class in Java is an unbounded blocking queue of elements with a delay time.

89. **What is the `ConcurrentHashMap` class in Java?**
  - The `ConcurrentHashMap` class in Java is a thread-safe implementation of the `Map` interface.

90. **What is the `CopyOnWriteArrayList` class in Java?**
  - The `CopyOnWriteArrayList` class in Java is a thread-safe implementation of the `List` interface.

91. **What is the `CopyOnWriteArraySet` class in Java?**
  - The `CopyOnWriteArraySet` class in Java is a thread-safe implementation of the `Set` interface.

92. **What is the `ConcurrentSkipListMap` class in Java?**
  - The `ConcurrentSkipListMap` class in Java is a concurrent, sorted, navigable map implementation based on a skip list.

93. **What is the `ConcurrentSkipListSet` class in Java?**
  - The `ConcurrentSkipListSet` class in Java is a concurrent, sorted, navigable set implementation based on a skip list.

94. **What is the purpose of the `java.util.concurrent.atomic` package in Java?**
  - The `java.util.concurrent.atomic` package in Java provides classes that support lock-free thread-safe programming on single variables.

95. **What is the `AtomicInteger` class in Java?**

- The `AtomicInteger` class in Java is a class that provides atomic operations on integer values.

96. **What is the `AtomicLong` class in Java?**
- The `AtomicLong` class in Java is a class that provides atomic operations on long values.

97. **What is the `AtomicReference` class in Java?**
- The `AtomicReference` class in Java is a class that provides atomic operations on object references.

98. **What is the purpose of the `java.util.concurrent.locks` package in Java?**
- The `java.util.concurrent.locks` package in Java provides a framework for locking and waiting for conditions that is distinct from the built-in synchronization support.

99. **What is the `Lock` interface in Java?**
- The `Lock` interface in Java provides a way to control access to a shared resource by multiple threads.

100. **What is the `ReentrantLock` class in Java?**
- The `ReentrantLock` class in Java implements the `Lock` interface and provides a reentrant mutual exclusion lock.