Advance JAVA Interview Question

1. **What is String in Java?**
   In Java, a String is an object that represents a sequence of characters. It is a built-in class in Java and is used extensively for text processing and manipulation.
   J**ava String** class provides a lot of methods to perform operations on strings such as compare(), concat(), equals(), split(), length(), replace(), compareTo(), intern(), substring() etc.

```
char[] ch={'j','a','v','a','t','p','o','i','n','t'};
String s=new String(ch);
```

2. **What is an Immutable String in Java?**
   An Immutable String in Java is a String object whose value cannot be changed after it is created. Once a String is created, it cannot be modified. Any operation that seems to modify a String actually creates a new String object.
3. **How do you compare two strings in Java?**
   In Java, you can compare two strings for equality using the `equals()` method or using the `==` operator. The `equals()` method compares the content of the strings, while the `==` operator compares the references of the string objects.

```
javaCopy code
public class StringComparisonExample {
    public static void main(String[] args) {
        String str1 = "Hello";
        String str2 = "Hello";
        String str3 = new String("Hello");

        System.out.println(str1.equals(str2)); // Output: true
        System.out.println(str1 == str2);      // Output: true

        System.out.println(str1.equals(str3)); // Output: true
        System.out.println(str1 == str3);      // Output: false
    }
```

```
}
```

4. **How do you concatenate strings in Java?**
   In Java, you can concatenate strings using the `+` operator or the `concat()` method.

```
javaCopy code
public class StringConcatenationExample {
    public static void main(String[] args) {
        String firstName = "John";
        String lastName = "Doe";

        // Using the + operator
        String fullName1 = firstName + " " + lastName;
        System.out.println(fullName1); // Output: John Doe

        // Using the concat() method
        String fullName2 = firstName.concat(" ").concat(lastName);
        System.out.println(fullName2); // Output: John Doe
    }
}
```

5. **How do you extract substrings from a string in Java?**
   In Java, you can extract substrings from a string using the `substring()` method.

```
javaCopy code
public class SubstringExample {
    public static void main(String[] args) {
        String message = "Hello, World!";
        String subMessage = message.substring(0, 5);
        System.out.println(subMessage); // Output: Hello
    }
}
```

6. **What are some common methods of the String class in Java?**
   Some common methods of the String class in Java are:

   - `length()`: Returns the length of the string.
   - `charAt(index)`: Returns the character at the specified index.
   - `toUpperCase()`: Converts the string to uppercase.

- `toLowerCase()`: Converts the string to lowercase.
- `indexOf(str)`: Returns the index of the first occurrence of the specified substring.
- `startsWith(prefix)`: Checks if the string starts with the specified prefix.
- `endsWith(suffix)`: Checks if the string ends with the specified suffix.

```
javaCopy code
public class StringMethodsExample {
    public static void main(String[] args) {
        String str = "Hello, Java";

        System.out.println("Length: " + str.length()); // Output: 11
        System.out.println("Character at index 4: " + str.charAt(4)); // Output: o
        System.out.println("Uppercase: " + str.toUpperCase()); // Output: HELLO, JAVA
        System.out.println("Lowercase: " + str.toLowerCase()); // Output: hello, java
        System.out.println("Index of 'Java': " + str.indexOf("Java")); // Output: 7
        System.out.println("Starts with 'Hello': " + str.startsWith("Hello")); // Output:
 true
        System.out.println("Ends with 'Java': " + str.endsWith("Java")); // Output: true
    }
}
```

7. **What is the StringBuffer class in Java?**
   The StringBuffer class in Java is used to create mutable strings. It provides methods to modify the content of a string without creating a new object.

```
javaCopy code
public class StringBufferExample {
    public static void main(String[] args) {
        StringBuffer buffer = new StringBuffer("Hello");
        buffer.append(", ");
        buffer.append("Java");
        System.out.println(buffer); // Output: Hello, Java
    }
}
```

8. **What is the StringBuilder class in Java?**
   The StringBuilder class in Java is similar to the StringBuffer class, but it is not thread-safe. It is more efficient than StringBuffer in a single-threaded environment.

```
javaCopy code
public class StringBuilderExample {
    public static void main(String[] args) {
        StringBuilder builder = new StringBuilder("Hello");
        builder.append(", ");
        builder.append("Java");
        System.out.println(builder); // Output: Hello, Java
    }
}
```

9. **What is the difference between String and StringBuffer in Java?**
   The main difference between String and StringBuffer in Java is that String is immutable, whereas StringBuffer is mutable.
10. **What is the difference between StringBuffer and StringBuilder in Java?**
    The main difference between StringBuffer and StringBuilder in Java is that StringBuffer is synchronized and thread-safe, while StringBuilder is not synchronized and not thread-safe.
11. **How do you create an immutable class in Java?**
    To create an immutable class in Java, follow these guidelines:

- Declare the class as final to prevent inheritance.
- Make all fields private and final to ensure they cannot be modified.
- Do not provide setter methods.
- Ensure that any mutable objects used within the class are not directly accessible or mutable.
- Provide only getter methods to access the fields.

```
javaCopy code
public final class ImmutablePerson {
    private final String name;
    private final int age;

    public ImmutablePerson(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public String getName() {
```

```
        return name;
    }

    public int getAge() {
        return age;
    }
}
```

12. **What is the purpose of the `toString()` method in Java?**
    The `toString()` method in Java is used to represent an object as a string. It is often used for debugging or printing the contents of an object.

```
javaCopy code
public class ToStringExample {
    public static void main(String[] args) {
        Person person = new Person("John", 30);
        System.out.println(person); // Output: Person[name=John, age=30]
    }
}

class Person {
    private String name;
    private int age;

    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    @Override
    public String toString() {
        return "Person[name=" + name + ", age=" + age + "]";
    }
}
```

13. **What is the String Tokenizer class in Java?**
    The String Tokenizer class in Java is used to break a string into tokens based on a specified delimiter.

```
javaCopy code
import java.util.StringTokenizer;

public class StringTokenizerExample {
    public static void main(String[] args) {
```

```
        String sentence = "Java programming is fun";
        StringTokenizer tokenizer = new StringTokenizer(sentence);

        while (tokenizer.hasMoreTokens()) {
            System.out.println(tokenizer.nextToken());
        }
    }
}
```

CharAt() Method

14. **What is the purpose of the `charAt()` method in Java? Provide an example of its usage.**

The `charAt()` method is used to get the character at the specified index in a String.

Example:

```
javaCopy code
String str = "Hello";
char ch = str.charAt(2); // Returns 'l'
```

CompareTo() Method

15. **How does the `compareTo()` method work in Java? Give an example.**

The `compareTo()` method compares two strings lexicographically and returns an integer value:

- 0 if both strings are equal.
- A positive value if the current string is lexicographically greater than the other string.
- A negative value if the current string is lexicographically less than the other string.

Example:

```
javaCopy code
String str1 = "apple";
String str2 = "banana";
int result = str1.compareTo(str2); // Returns a negative value
```

Concat() Method

16. **What does the `concat()` method do in Java? Provide an example.**
The `concat()` method is used to concatenate the specified string to the end of the current string and returns a new string.

Example:

```
javaCopy code
String str1 = "Hello";
String str2 = "World";
String result = str1.concat(str2); // Returns "HelloWorld"
```

Contains() Method

17. **How does the `contains()` method work in Java? Show an example.**
The `contains()` method checks if the current string contains the specified character sequence and returns a boolean value.

Example:

```
javaCopy code
String str = "Hello, Java";
boolean containsJava = str.contains("Java"); // Returns true
```

endsWith() Method

18. **What is the purpose of the `endsWith()` method in Java? Give an example.**
The `endsWith()` method checks if the current string ends with the specified suffix and returns a boolean value.

    Example:

```
javaCopy code
String str = "Hello, Java";
boolean endsWithJava = str.endsWith("Java"); // Returns true
```

equal() Method

19. **How do you compare two strings for equality using the `equals()` method in Java?**
    The `equals()` method is used to compare the current string with the specified object and returns a boolean value indicating whether they are equal.

    Example:

```
javaCopy code
String str1 = "Hello";
String str2 = "Hello";
boolean isEqual = str1.equals(str2); // Returns true
```

equalIgnoreCase() Method

20. **How is `equalsIgnoreCase()` different from `equals()`? Provide an example.**

The `equalsIgnoreCase()` method compares two strings for equality while ignoring their case.

Example:

```
javaCopy code
String str1 = "hello";
String str2 = "HeLLo";
boolean isEqualIgnoreCase = str1.equalsIgnoreCase(str2); // Returns true
```

format() Method

21. **Explain the use of the `format()` method in Java with an example.**
The `format()` method is used to create a formatted string using the specified format and arguments.

Example:

```
javaCopy code
String name = "John";
int age = 30;
String formattedString = String.format("My name is %s and I am %d years old.", name, a
ge);
// Returns "My name is John and I am 30 years old."
```

getBytes() Method

22. **What does the `getBytes()` method do in Java? Provide an example.**
The `getBytes()` method converts the string to a byte array using the platform's default charset.

Example:

```
javaCopy code
String str = "Hello, World!";
byte[] byteArray = str.getBytes();
```

getChars() Method

23. **How do you use the `getChars()` method in Java? Give an example.**
he `getChars()` method copies characters from the current string to the destination character array.

Example:

```
javaCopy code
String str = "Hello, Java";
char[] charArray = new char[5];
str.getChars(0, 5, charArray, 0); // Copies "Hello" to charArray
```

indexOf() Method

24. **Explain the use of the `indexOf()` method in Java with an example.**
The `indexOf()` method returns the index of the first occurrence of the specified substring in the current string.

Example:

```
javaCopy code
String str = "Hello, Java";
int index = str.indexOf("Java"); // Returns 7
```

intern() Method

25. **What is the purpose of the `intern()` method in Java? Provide an example.**
The `intern()` method returns the canonical representation of the string from the string pool if it is already present.

Example:

```
javaCopy code
String str1 = "Hello";
String str2 = new String("Hello").intern();
boolean isEqual = str1 == str2; // Returns true
```

isEmpty() Method

26. **How do you check if a string is empty using the `isEmpty()` method in Java?**
The `isEmpty()` method checks if the current string is empty (contains no characters) and returns a boolean value.

Example:

```
javaCopy code
String str = "";
boolean isEmpty = str.isEmpty(); // Returns true
```

join() Method

27. **Explain the purpose of the `join()` method in Java with an example.**
The `join()` method joins the elements of the provided CharSequence sequence using the specified delimiter and returns a new string.

Example:

```
javaCopy code
String[] names = {"John", "Doe", "Alice"};
String joinedString = String.join(", ", names); // Returns "John, Doe, Alice"
```

lastIndex() Method

28. **How does the `lastIndexOf()` method work in Java? Provide an example.**
The `lastIndexOf()` method returns the index of the last occurrence of the specified substring in the current string.

Example:

```
javaCopy code
String str = "Hello, Java";
int lastIndex = str.lastIndexOf("Java"); // Returns 7
```

length() Method

29. **What is the purpose of the `length()` method in Java? Give an example.**
The `length()` method is used to get the length (number of characters) of the current string.

Example:

```
javaCopy code
String str = "Hello, World!";
int length = str.length(); // Returns 13
```

replace() Method

30. **Explain the use of the `replace()` method in Java with an example.**
The `replace()` method replaces all occurrences of the specified old character with the new character in the current string.

Example:

```
javaCopy code
String str = "Hello, World!";
String replacedStr = str.replace('o', 'x'); // Returns "Hellx, Wxrld!"
```

replaceAll() Method

31. **How is `replaceAll()` different from `replace()`? Provide an example.**
The `replaceAll()` method replaces all occurrences of the specified regular expression with the replacement string in the current string.

Example:

```
javaCopy code
String str = "Hello, Java";
String replacedStr = str.replaceAll("a", "x"); // Returns "Hello, Jxvx"
```

split() Method

32. **Explain the use of the `split()` method in Java with an example.**

The `split()` method splits the current string around matches of the given regular expression and returns an array of substrings.

Example:

```
javaCopy code
String str = "apple, banana, orange";
String[] fruits = str.split(", ");
// Returns an array: ["apple", "banana", "orange"]
```

startsWith() Method

33. **What does the `startsWith()` method do in Java? Give an example.**
The `startsWith()` method checks if the current string starts with the specified prefix and returns a boolean value.

Example:

```
javaCopy code
String str = "Hello, Java";
boolean startsWithHello = str.startsWith("Hello"); // Returns true
```

substring() Method

34. **How is the `substring()` method used in Java? Show an example.**
The `substring()` method returns a new string that is a substring of the current string, starting from the specified index.

Example:

```
javaCopy code
String str = "Hello, Java";
```

```
String subStr = str.substring(7); // Returns "Java"
```

toChareArray() Method

35. **What is the purpose of the `toCharArray()` method in Java? Provide an example.**
The `toCharArray()` method converts the current string to a character array.

Example:

```
javaCopy code
String str = "Hello";
char[] charArray = str.toCharArray(); // Returns ['H', 'e', 'l', 'l', 'o']
```

toLowerCase() Method

36. **How do you convert a string to lowercase using the `toLowerCase()` method in Java?**
The `toLowerCase()` method returns a new string with all the characters converted to lowercase.

Example:

```
javaCopy code
String str = "Hello, World!";
String lowerCaseStr = str.toLowerCase(); // Returns "hello, world!"
```

toUpperCase() Method

37. **Explain the use of the `toUpperCase()` method in Java with an example.**
The `toUpperCase()` method returns a new string with all the characters converted to uppercase.

Example:

```
javaCopy code
String str = "Hello, World!";
String upperCaseStr = str.toUpperCase(); // Returns "HELLO, WORLD!"
```

trim() Method

38. **How does the `trim()` method work in Java? Show an example.**
The `trim()` method removes leading and trailing whitespace from the current string.

Example:

```
javaCopy code
String str = "   Hello, Java   ";
String trimmedStr = str.trim(); // Returns "Hello, Java"
```

valueof() Method

39. **What is the purpose of the `valueOf()` method in Java? Give an example.**
The `valueOf()` method converts an object to its string representation.

Example:

```
javaCopy code
int number = 42;
```

```
String str = String.valueOf(number); // Returns "42"
```

Java Regex

1. **What is a regular expression in Java? Provide an example of using regular expressions.**
   Answer: A regular expression is a sequence of characters that defines a search pattern. In Java, regular expressions are used with classes like `Pattern` and `Matcher` from the `java.util.regex` package.

   Example:

```
import java.util.regex.*;

public class RegexExample {
    public static void main(String[] args) {
        String input = "Hello, Java!";
        Pattern pattern = Pattern.compile("Java");
        Matcher matcher = pattern.matcher(input);
        if (matcher.find()) {
            System.out.println("Pattern found!");
        } else {
            System.out.println("Pattern not found!");
        }
    }
}
```

Exception Handling

1. **What is exception handling in Java? How do you handle exceptions in Java?**
   Exception handling in Java allows the program to gracefully handle unexpected situations or errors. It involves using `try`, `catch`, `finally`, and `throw` blocks.

   Example:

```
public class ExceptionHandlingExample {
    public static void main(String[] args) {
        try {
```

```
            int result = 10 / 0; // Throws an ArithmeticException
            System.out.println("Result: " + result); // This line won't be executed
        } catch (ArithmeticException ex) {
            System.out.println("Exception caught: " + ex.getMessage());
        } finally {
            System.out.println("Finally block executed!");
        }
    }
}
```

Java Inner Classes

1. **What are inner classes in Java? How do you use them?**
   Inner classes are classes defined within other classes. They are used to logically group classes that are only used in one place.

   Example:

```
public class OuterClass {
    private int data = 10;

    class InnerClass {
        void display() {
            System.out.println("Data: " + data);
        }
    }
}

public class Main {
    public static void main(String[] args) {
        OuterClass outer = new OuterClass();
        OuterClass.InnerClass inner = outer.new InnerClass();
        inner.display(); // Output: Data: 10
    }
}
```

Java Multithreading

1. **What is multithreading in Java? How do you create and start a thread?**
   Multithreading in Java allows multiple threads to execute concurrently. You can

create and start a thread by extending the `Thread` class or implementing the `Runnable` interface.

Example (Extending Thread):

```
public class MyThread extends Thread {
    public void run() {
        System.out.println("Thread is running.");
    }
}

public class Main {
    public static void main(String[] args) {
        MyThread thread = new MyThread();
        thread.start(); // Output: Thread is running.
    }
}
```

Example (Implementing Runnable):

```
public class MyRunnable implements Runnable {
    public void run() {
        System.out.println("Runnable is running.");
    }
}

public class Main {
    public static void main(String[] args) {
        MyRunnable runnable = new MyRunnable();
        Thread thread = new Thread(runnable);
        thread.start(); // Output: Runnable is running.
    }
}
```

Java I/O

1. **How do you read input from the user using `Scanner` class in Java?**
   You can use the `Scanner` class from `java.util` to read input from the user.

   Example:

```
import java.util.Scanner;

public class ScannerExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter your name: ");
        String name = scanner.nextLine();
        System.out.println("Hello, " + name + "!");
        scanner.close();
    }
}
```

Java Networking

1. **How do you create a simple server-client application in Java using sockets?**
   You can create a server-client application using the `ServerSocket` and `Socket` classes from `java.net`.

   Example (Server):

```
import java.io.*;
import java.net.*;

public class Server {
    public static void main(String[] args) {
        try {
            ServerSocket serverSocket = new ServerSocket(5000);
            System.out.println("Server is waiting for a connection...");
            Socket socket = serverSocket.accept();
            System.out.println("Connection established!");

            BufferedReader reader = new BufferedReader(new InputStreamReader(socket.g
etInputStream()));
            String message = reader.readLine();
            System.out.println("Received message: " + message);

            reader.close();
            socket.close();
            serverSocket.close();
        }
         catch (IOException e) {
e.printStackTrace();
}
}
}Example (Client):
```

Example (Client):

```
```java
import java.io.*;
import java.net.*;

public class Client {
    public static void main(String[] args) {
        try {
            Socket socket = new Socket("localhost", 5000);
            PrintWriter writer = new PrintWriter(socket.getOutputStream(), true);
            writer.println("Hello, server!");

            writer.close();
            socket.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Java AWT

1. **What is AWT in Java? How do you create a simple window using `Frame` class?**
   AWT (Abstract Window Toolkit) is a GUI toolkit in Java. You can create a simple window using the `Frame` class.

   Example:

```
import java.awt.*;

public class SimpleWindow {
    public static void main(String[] args) {
        Frame frame = new Frame("Simple Window");
        frame.setSize(300, 200);
        frame.setVisible(true);
    }
}
```
```

Java Swing

1. **How do you create a simple Swing application with a button and event handling?**
   Swing is a GUI library in Java. You can create a simple Swing application with a button and event handling using `JFrame` and `JButton`.

   Example:

```
import javax.swing.*;
import java.awt.event.*;

public class SimpleSwingApp {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Simple Swing App");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JButton button = new JButton("Click Me!");
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                JOptionPane.showMessageDialog(frame, "Button Clicked!");
            }
        });

        frame.add(button);
        frame.pack();
        frame.setVisible(true);
    }
}
```

JavaFX

1. **What is JavaFX? How do you create a simple JavaFX application with a button and event handling?**
   JavaFX is a GUI library in Java. You can create a simple JavaFX application with a button and event handling using `Stage`, `Scene`, and `Button`.

   Example:

```
import javafx.application.Application;
import javafx.scene.Scene;
```

```
import javafx.scene.control.Button;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;

public class SimpleJavaFXApp extends Application {
    public static void main(String[] args) {
        launch(args);
    }

    public void start(Stage primaryStage) {
        primaryStage.setTitle("Simple JavaFX App");
        Button button = new Button("Click Me!");
        button.setOnAction(e -> {
            System.out.println("Button Clicked!");
        });

        StackPane root = new StackPane();
        root.getChildren().add(button);
        primaryStage.setScene(new Scene(root, 300, 200));
        primaryStage.show();
    }
}
```

Java Applet

1. **What is a Java Applet? How do you create a simple applet in Java?**
   A Java Applet is a small program that runs within a web browser. You can create a simple applet by extending the `Applet` class.

   Example:

```
import java.applet.Applet;
import java.awt.*;

public class SimpleApplet extends Applet {
    public void paint(Graphics g) {
        g.drawString("Hello, Applet!", 20, 30);
    }
}
```

Java Reflection

1. **What is Java Reflection? How do you get the class information and invoke methods using Reflection?**
   Java Reflection allows you to examine or modify the runtime behavior of classes, methods, and fields. You can get the class information and invoke methods dynamically using Reflection.

   Example:

```
import java.lang.reflect.*;

public class ReflectionExample {
    public static void main(String[] args) throws Exception {
        Class<?> clazz = String.class;
        System.out.println("Class Name: " + clazz.getName());

        Method method = clazz.getMethod("toUpperCase");
        String result = (String) method.invoke("hello");
        System.out.println("Result: " + result); // Output: "HELLO"
    }
}
```

Java Date

1. **How do you work with dates in Java? How do you get the current date and format it using `SimpleDateFormat`?**
   In Java, you can work with dates using `java.util.Date` and `SimpleDateFormat`.

   Example:

```
import java.util.*;
import java.text.*;

public class DateExample {
    public static void main(String[] args) {
        Date currentDate = new Date();
        System.out.println("Current Date: " + currentDate);

        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
        String formattedDate = sdf.format(currentDate);
        System.out.println("Formatted Date: " + formattedDate);
    }
}
```

Java Conversion

1. **How do you convert one data type to another in Java? Provide an example of type casting and parsing.**
   You can convert one data type to another using type casting and parsing.

Example (Type Casting):

```
double numDouble = 3.14;
int numInt = (int) numDouble; // Type casting
System.out.println("Converted Integer: " + numInt); // Output: 3
```

Example (Parsing):

```
String numStr = "42";
int numInt = Integer.parseInt(numStr); // Parsing
System.out.println("Parsed Integer: " + numInt); // Output: 42
```