Java Interview Questions

Java Interview Questions

Easy:

1. **Why is the Java platform independent?**
    - *Java is platform-independent* because its code is compiled into bytecode, which can run on any device with a Java Virtual Machine (JVM), abstracting away hardware and operating system specifics.
2. **Explain JVM, JRE, JDK?**
    - *JVM (Java Virtual Machine)*: Interprets Java bytecode into machine code specific to the underlying hardware.
    - *JRE (Java Runtime Environment)*: Includes JVM and libraries for running Java applications.
    - *JDK (Java Development Kit)*: Contains JRE along with development tools for building Java applications.

3. **Why are JRE, JVM, JDK platform dependent?**
    - They are *platform-dependent* because they rely on native implementations specific to the underlying operating system and hardware architecture.
4. **Explain some Java features?**
    - *Object-oriented programming*, *platform independence*, *robustness*, *security*, *multi-threading support*, and *automatic memory management*.
5. **What makes Java better than C++?**
    - *Automatic memory management*, *platform independence*, *strong standard libraries*, and *simpler syntax for certain tasks*.
6. **What is the format of Java files?**
    - Java files have a ".java" extension and typically contain one public class with the same name as the file.
7. **What is OOP in Java?**
    - *Object-Oriented Programming in Java* refers to principles like encapsulation, inheritance, polymorphism, and abstraction.
8. **Why is Java called an OOP language?**
    - Java implements key OOP principles like encapsulation, inheritance, and polymorphism.
9. **Why is Java not a 100% OOP-based language?**
    - Java supports *primitive data types* and *static methods/variables*, which are not strictly objects.
10. **How to make Java a 100% OOP-based language?**
    - All data types would need to be objects, and there would be no provision for static methods or variables.
11. **Explain a class and object in Java with a real-life example?**
    - *Class*: Blueprint defining attributes and methods (e.g., "Car").
    - *Object*: Instance of a class (e.g., Toyota Camry).
12. **Explain the 4 pillars of OOP concept in detail?**

- *Encapsulation*, *Inheritance*, *Polymorphism*, *Abstraction*.

13. **Explain variables and its types in Java?**

- Variables in Java are containers for storing data values. They have a data type and a name. Types include:
  - *Primitive*: Hold simple values like numbers or characters.
  - *Reference*: Store references to objects.
  - *Local*: Declared within a method and accessible only within that method.
  - *Instance (or Object)*: Associated with an instance of a class.
  - *Class (or Static)*: Associated with a class itself rather than instances of the class.

15. **Explain all data types in Java both primitive and non-primitive?**

- *Primitive types*: byte, short, int, long, float, double, char, boolean.
- *Non-primitive types*: Arrays, Strings, Classes, Interfaces.

16. **Difference between objects in Java and JavaScript?**

- *Objects in Java*: Instances of classes, with properties and methods defined by the class blueprint.
- *Objects in JavaScript*: Fundamental data structure consisting of key-value pairs, allowing dynamic addition and removal of properties. JavaScript objects are more flexible and can be used as associative arrays. Additionally, JavaScript is a scripting language used for web development, while Java is a general-purpose programming language.

Medium:

1. *Explain 'this' and 'super' keywords in Java with examples?*
   - *this:* Refers to the current instance of the class and is typically used to access instance variables or methods.

- *super:* Refers to the superclass of the current object or can be used to call the superclass constructor or methods.

2. *What are constructors and how do they differ from methods?*
    - *Constructors* are special methods used to initialize objects of a class. They have the same name as the class and no return type.
    - Unlike methods, constructors are called automatically when an object is created and are used to set initial values for instance variables.
3. *Why are getters and setters used in Java?*
    - Getters and setters are used to access and modify the private fields of a class, ensuring encapsulation and data hiding.
    - They provide control over the access to class fields and allow validation or modification logic to be added.
4. *What are modifiers and explain their use case in real-life software development?*
    - *Modifiers* are keywords used to specify the properties and behaviors of classes, methods, and variables.
    - They control access levels, inheritance, and other characteristics of code elements, enhancing code readability, security, and maintainability.
5. *What is upcasting and downcasting in Java?*
    - *Upcasting:* Casting an object of a subclass to its superclass type. It's implicit and safe.
    - *Downcasting:* Casting an object of a superclass to its subclass type. It requires explicit casting and can lead to ClassCastException if not done properly.
6. *Difference between final, finalize, finally keyword?*
    - *final:* Keyword used to declare constants, prevent method overriding, and make classes immutable.
    - *finalize:* Method called by the garbage collector before reclaiming an object's memory.

- *finally:* Block used in exception handling to execute code whether an exception is thrown or not.

7. *Types of Data Structures in Java?*
    - Arrays, ArrayList, LinkedList, Stack, Queue, HashSet, HashMap, TreeMap, etc.
8. *Why is String a non-primitive data type?*
    - Strings in Java are objects of the String class, which provides various methods to manipulate strings. Unlike primitive data types, they have methods and are stored in the heap memory.
9. *Difference between field and variable in Java?*
    - *Fields* are variables declared within a class, while variables can be declared in various scopes including methods, constructors, and blocks.
10. *Why are Strings immutable?*
    - Strings are immutable to ensure thread safety, security, and performance optimizations. Once created, their values cannot be changed.
11. *What is an array?*
    - An array is a data structure that stores a fixed-size sequential collection of elements of the same type.
12. *Explain about the garbage collector and when it is used?*
    - The garbage collector is a part of the JVM responsible for reclaiming memory by deallocating objects that are no longer in use.
    - It's used to free up memory occupied by objects that are no longer reachable or referenced by any part of the program.
13. *Can we run Java code without the main method?*
    - No, the main method is the entry point for Java programs. Without it, the program cannot be executed.
14. *Why is the main method static?*
    - The main method is static so that it can be called without creating an instance of the class, which is necessary for the JVM to start the

execution of the program.

15. *What is the Collection Framework in Java?*
    - The Collection Framework is a unified architecture for representing and manipulating collections of objects in Java.
16. *Explain various interfaces used in the Collection framework?*
    - Interfaces like List, Set, Queue, Map, etc., provide common methods for manipulating collections and ensure interoperability and flexibility.
17. *What is the problem generics solve?*
    - Generics provide type safety by allowing the use of parameterized types, which enable the compiler to detect and prevent type errors at compile time.
18. *Difference between error and exception?*
    - *Errors* are serious issues that occur at runtime and are usually irrecoverable, such as OutOfMemoryError.
    - *Exceptions* are unexpected events that occur during the execution of a program and can be handled using try-catch blocks.
19. *What is compile time and run time?*
    - *Compile time* refers to the time when the source code is translated into bytecode by the compiler.
    - *Runtime* refers to the period when the program is executed by the JVM.
20. *Explain compile time error and run time error?*
    - *Compile time errors* occur during compilation when the code violates language syntax rules and cannot be translated into bytecode.
    - *Runtime errors* occur during program execution due to logical errors, such as division by zero or accessing null references.

Hard:

1. **What is multitasking and multi-process?**

- *Multitasking*: The ability of an operating system to run *multiple programs or processes* simultaneously, sharing the CPU's time.
- *Multiprocessing*: Involves running *multiple processes* simultaneously on a multi-core processor or multiple processors.

2. **Describe thread life cycle?**
    - Thread life cycle consists of various states: *new, runnable, running, blocked, waiting, timed waiting, and terminated*. Threads transition between these states based on their execution and synchronization requirements.
3. **What is an API given some inbuilt Java APIs?**
    - API (*Application Programming Interface*) provides a set of rules and protocols for building and interacting with software applications. In Java, some inbuilt APIs include *java.util*, *java.io*, *java.net*, etc.
4. **Explain types of memories in Java?**
    - *Heap Memory*: Used for storing objects created during the execution of a program.
    - *Stack Memory*: Stores method call frames, local variables, and partial results.
    - *PermGen (or Metaspace)*: Holds metadata about classes and methods.
5. **Explain String memory management?**
    - Strings are stored in the *string pool* in the heap memory. Literal strings are *interned and reused*, while dynamically created strings are stored separately in the heap.
6. **Difference between array, array list, linked list?**
    - *Array*: Fixed-size data structure containing elements of the same type.
    - *ArrayList*: Dynamic-size list backed by an array, allowing for resizable arrays.
    - *LinkedList*: Collection of nodes, each containing a reference to the next node, providing fast insertion and deletion.

7. **Difference between HashSet, LinkedHashSet, TreeSet?**
    - *HashSet*: Unordered collection that does not allow duplicate elements.
    - *LinkedHashSet*: Ordered collection that maintains insertion order.
    - *TreeSet*: Sorted collection that maintains elements in ascending order.
8. **Why is Java 8 most popular when many versions are available?**
    - Java 8 introduced significant features like *lambda expressions*, the *Stream API*, and *functional interfaces*, which greatly improved developer productivity and code readability.
9. **List out all Java 8 features?**
    - *Lambda expressions*, *Stream API*, *Default methods*, *Method references*, *Functional interfaces*, *Optional class*, *Date and Time API*, etc.
10. **What is a Functional interface, and what's the use of it?**
    - A *Functional Interface* is an interface with only one abstract method. They are used to enable functional programming features like *lambda expressions* and *method references*.
11. **What problem does lambda functions solve?**
    - Lambda functions solve the verbosity issue in anonymous class implementations, making code more *concise and readable*.
12. **What's the use of Comparable and Comparator when we have inbuilt methods?**
    - *Comparable* is used for *natural ordering* of objects, while *Comparator* provides *custom ordering*. They offer flexibility in sorting objects based on different criteria.
13. **What is method reference and types of method reference?**
    - Method reference provides a way to refer to methods without invoking them. Types include *static method reference, instance method reference*, and *constructor reference*.
14. **What are streams in Java 8?**

- Streams represent a sequence of elements and support *functional-style operations* like *map, filter, reduce*, etc. They facilitate efficient processing of collections.

15. **Difference between Stream and Parallel Stream?**
    - Stream performs operations *sequentially*, while *Parallel Stream* utilizes *multiple threads* to perform operations concurrently, improving performance for large datasets.
16. **What's the use of map and flatMap in streams?**
    - *map* transforms each element of a stream using the provided function, while *flatMap* can *flatten nested collections* by applying a function to each element and then concatenating the results.
17. **Is Spring a Java framework or library?**
    - *Spring* is a Java framework used for building enterprise-level applications. It provides comprehensive infrastructure support and promotes *modular, testable, and maintainable code*.
18. **Explain Is-a relationship and has-a relationship?**
    - *Is-a Relationship*: Denotes inheritance, where a class is a subtype of another class. Example: *Car is-a Vehicle*.
    - *Has-a Relationship*: Denotes composition, where a class contains another class as a component or member. Example: *Car has-a Engine*.
19. **What is loose and tight coupling?**

- *Loose coupling*: Components are independent and interact with each other through *well-defined interfaces*, reducing dependencies and promoting *flexibility and maintainability*.
- *Tight coupling*: Components are strongly dependent on each other, making changes in one component require modifications in others, leading to *rigidity and complexity*.

20. **What is dependency injection in Spring Core?**

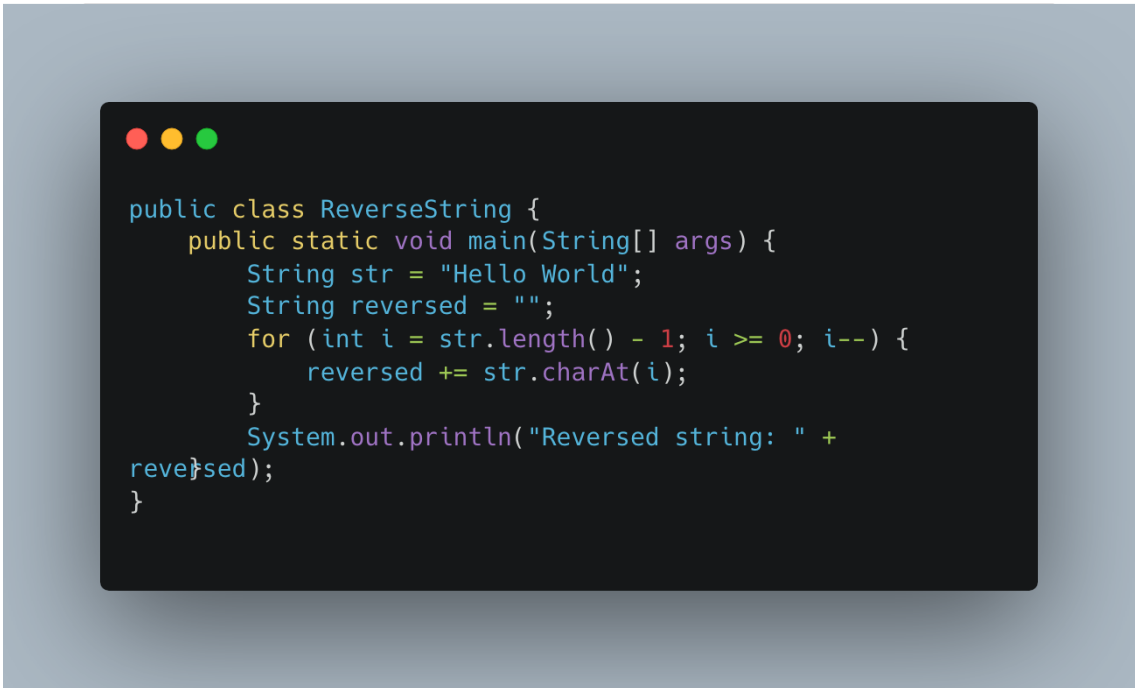
- *Dependency Injection (DI)* is a design pattern used to inject the dependencies of a class from an external source rather than creating them

internally. In Spring Core, DI is achieved using *Inversion of Control (IoC)* containers, where objects are provided with their dependencies.

21. Empty for formatting.
22. **What are different types of dependency injection in Spring Core?**
    - *Constructor Injection*, *Setter Injection*, and *Field Injection*.
23. **What is an annotation example of a few types of annotations and its use case?**
    - *@Autowired*: Used for *dependency injection*.
    - *@Component*: Used to mark a class as a *Spring bean*.
    - *@RequestMapping*: Maps *web requests* to handler methods.
24. **Name five Java Spring Boot features**
    - *Embedded HTTP servers*, *auto-configuration*, *starters*, *production-ready metrics and monitoring*, and *easy deployment*.
25. **What's the use of the pom.xml file?**
    - The *pom.xml* file is used in Maven projects to define project dependencies, build settings, and other configurations. It helps manage project dependencies, automate builds, and ensure consistency across different environments.
26. **Is Spring and Spring Boot the same? If yes, state the reason?**
    - No, Spring and Spring Boot are not the same.
    - *Spring*: It's a comprehensive framework that provides support for various functionalities like *dependency injection*, *aspect-oriented programming*, *transaction management*, etc.
    - *Spring Boot*: It's a sub-project of the Spring Framework that simplifies the development of Spring-based applications. It offers *auto-configuration* and *opinionated defaults* to minimize setup and configuration, allowing developers to focus on writing business logic.

Coding Questions:

1. Write a java program to reverse a String?

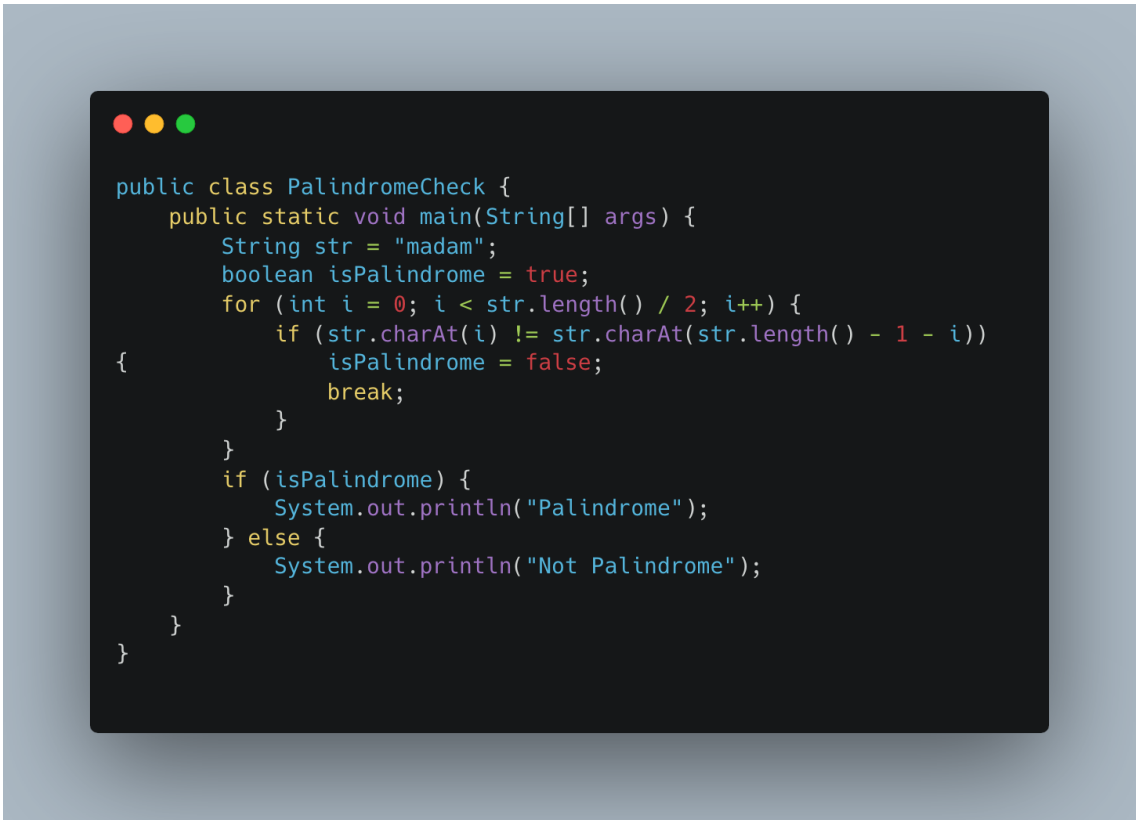


```java
public class ReverseString {
    public static void main(String[] args) {
        String str = "Hello World";
        String reversed = "";
        for (int i = str.length() - 1; i >= 0; i--) {
            reversed += str.charAt(i);
        }
        System.out.println("Reversed string: " +
reversed);
}
```

2. Write a java program to find no.of vowels in an array?

```java
public class VowelCount {
    public static void main(String[] args) {
        String str = "Hello World";
        int count = 0;
        for (char c : str.toCharArray()) {
            if ("aeiouAEIOU".indexOf(c) != -1) {
                count++;
            }
        }
        System.out.println("Number of vowels: " +
count);
}
```

3. Write a java program to find a String palindrome or not?



```java
public class PalindromeCheck {
    public static void main(String[] args) {
        String str = "madam";
        boolean isPalindrome = true;
        for (int i = 0; i < str.length() / 2; i++) {
            if (str.charAt(i) != str.charAt(str.length() - 1 - i))
{
                isPalindrome = false;
                break;
            }
        }
        if (isPalindrome) {
            System.out.println("Palindrome");
        } else {
            System.out.println("Not Palindrome");
        }
    }
}
```

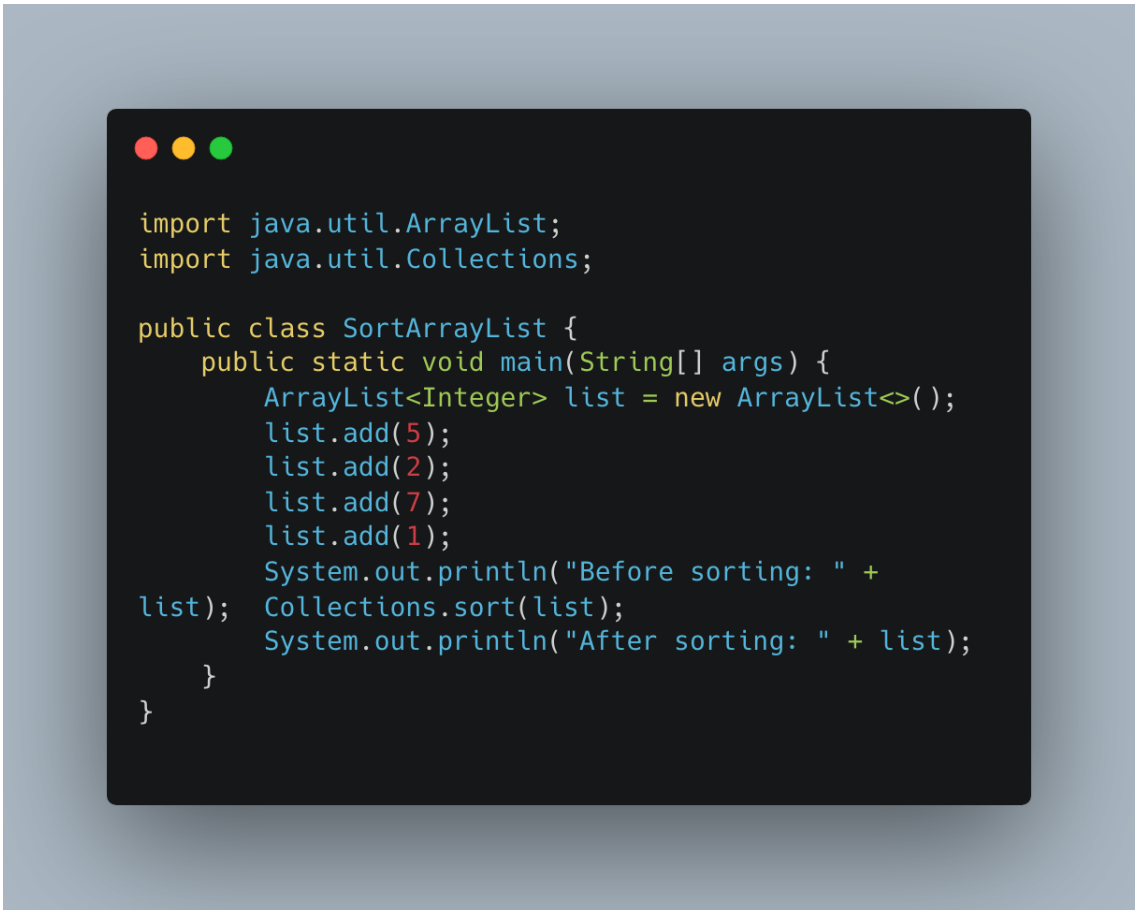
4. Write a java program to print fibonacci sequence without using recursion?

```java
public class PalindromeCheck {
    public static void main(String[] args) {
        String str = "madam";
        boolean isPalindrome = true;
        for (int i = 0; i < str.length() / 2; i++) {
            if (str.charAt(i) != str.charAt(str.length() - 1 - i))
{
                isPalindrome = false;
                break;
            }
        }
        if (isPalindrome) {
            System.out.println("Palindrome");
        } else {
            System.out.println("Not Palindrome");
        }
    }
}
```

5. Write a java program to find the factorial of a number?

```java
public class Factorial {
    public static void main(String[] args) {
        int n = 5;
        int factorial = 1;
        for (int i = 1; i <= n; i++) {
            factorial *= i;
        }
        System.out.println("Factorial of " + n + ": " +
factorial);
}
}
```

6. Write a java program to sort an arraylist?



```java
import java.util.ArrayList;
import java.util.Collections;

public class SortArrayList {
    public static void main(String[] args) {
        ArrayList<Integer> list = new ArrayList<>();
        list.add(5);
        list.add(2);
        list.add(7);
        list.add(1);
        System.out.println("Before sorting: " +
list);  Collections.sort(list);
        System.out.println("After sorting: " + list);
    }
}
```

7. Write a java program to implement binary search?

```java
public class BinarySearch {
    public static void main(String[] args) {
        int[] arr = {1, 2, 3, 4, 5, 6, 7};
        int key = 4;
        int result = binarySearch(arr, key);
        if (result != -1) {
            System.out.println("Element found at index: " +
result);} else {
            System.out.println("Element not found");
        }
    }

    public static int binarySearch(int[] arr, int key) {
        int low = 0, high = arr.length - 1;
        while (low <= high) {
            int mid = low + (high - low) / 2;
            if (arr[mid] == key) {
                return mid;
            } else if (arr[mid] < key) {
                low = mid + 1;
            } else {
                high = mid - 1;
            }
        }
        return -1;
    }
}
```

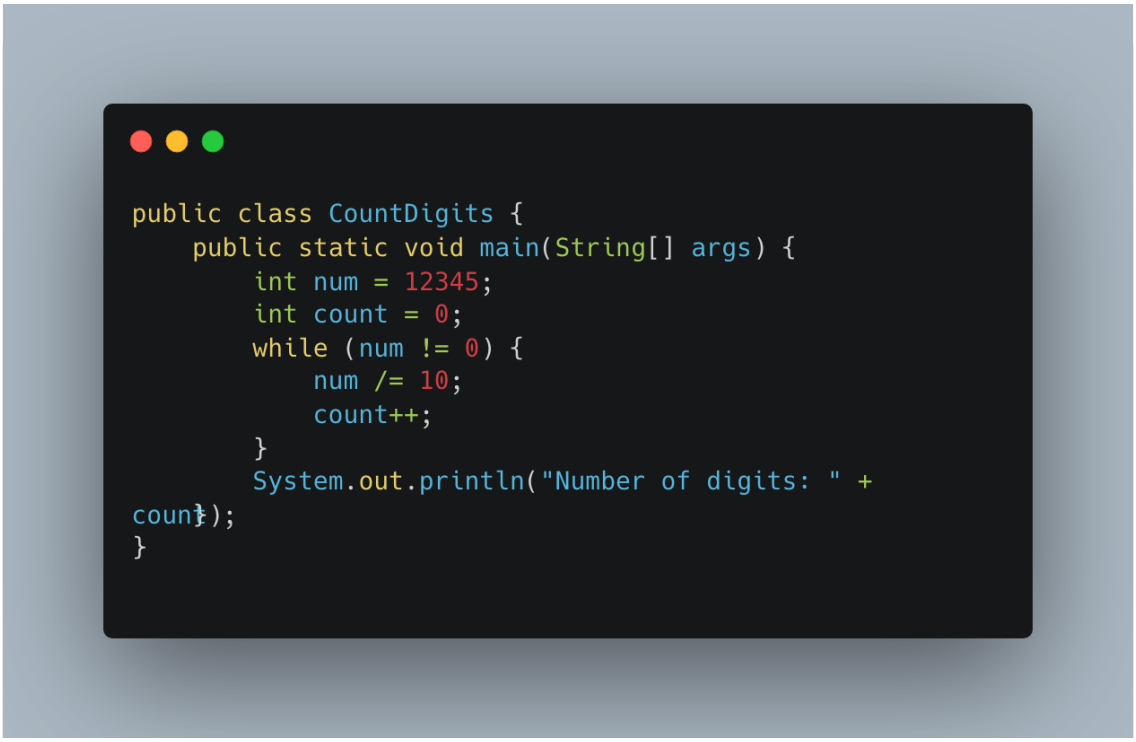
8. Write a java program to print the second largest number in an array?

```java
public class SecondLargest {
    public static void main(String[] args) {
        int[] arr = {5, 8, 2, 1, 9, 7};
        int firstMax = Integer.MIN_VALUE, secondMax =
Integer.MIN_VALUE;
        for (int num : arr) {
            if (num > firstMax) {
                secondMax = firstMax;
                firstMax = num;
            } else if (num > secondMax && num != firstMax) {
                secondMax = num;
            }
        }
        System.out.println("Second largest number: " + secondMax);
    }
}
```

9. Write a java program to swap 2 numbers without using a third variable?

```java
public class SwapNumbers {
    public static void main(String[] args) {
        int a = 5, b = 10;
        System.out.println("Before swapping: a = " + a + ", b = " +
b);
        a = a + b;
        b = a - b;
        a = a - b;
        System.out.println("After swapping: a = " + a + ", b = " + b);
    }
}
```

10. Write a java program to count no of digits in a number?

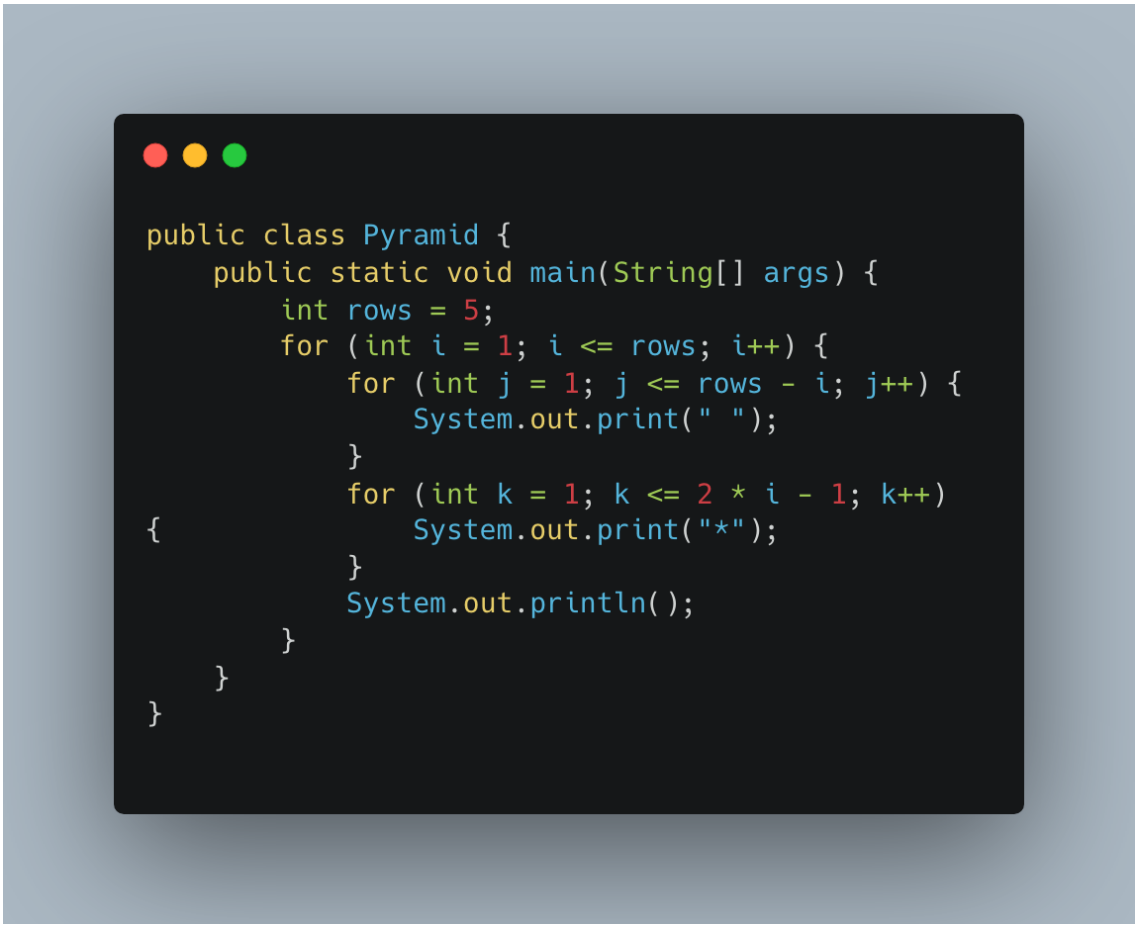


```java
public class CountDigits {
    public static void main(String[] args) {
        int num = 12345;
        int count = 0;
        while (num != 0) {
            num /= 10;
            count++;
        }
        System.out.println("Number of digits: " +
count);
}
```

11. Write a java program to count no of repeating characters in a String ?

```java
import java.util.HashMap;
import java.util.Map;

public class RepeatingCharacters {
    public static void main(String[] args) {
        String str = "hello world";
        Map<Character, Integer> map = new HashMap<>();
        for (char c : str.toCharArray()) {
            if (map.containsKey(c)) {
                map.put(c, map.get(c) + 1);
            } else {
                map.put(c, 1);
            }
        }
        for (Map.Entry<Character, Integer> entry : map.entrySet()) {
            if (entry.getValue() > 1) {
                System.out.println(entry.getKey() + ": " +
entry.getValue());
        }
    }
}
```

12. Write a java program to print the triangle shaped pyramid ?



```
public class Pyramid {
    public static void main(String[] args) {
        int rows = 5;
        for (int i = 1; i <= rows; i++) {
            for (int j = 1; j <= rows - i; j++) {
                System.out.print(" ");
            }
            for (int k = 1; k <= 2 * i - 1; k++)
{               System.out.print("*");
            }
            System.out.println();
        }
    }
}
```

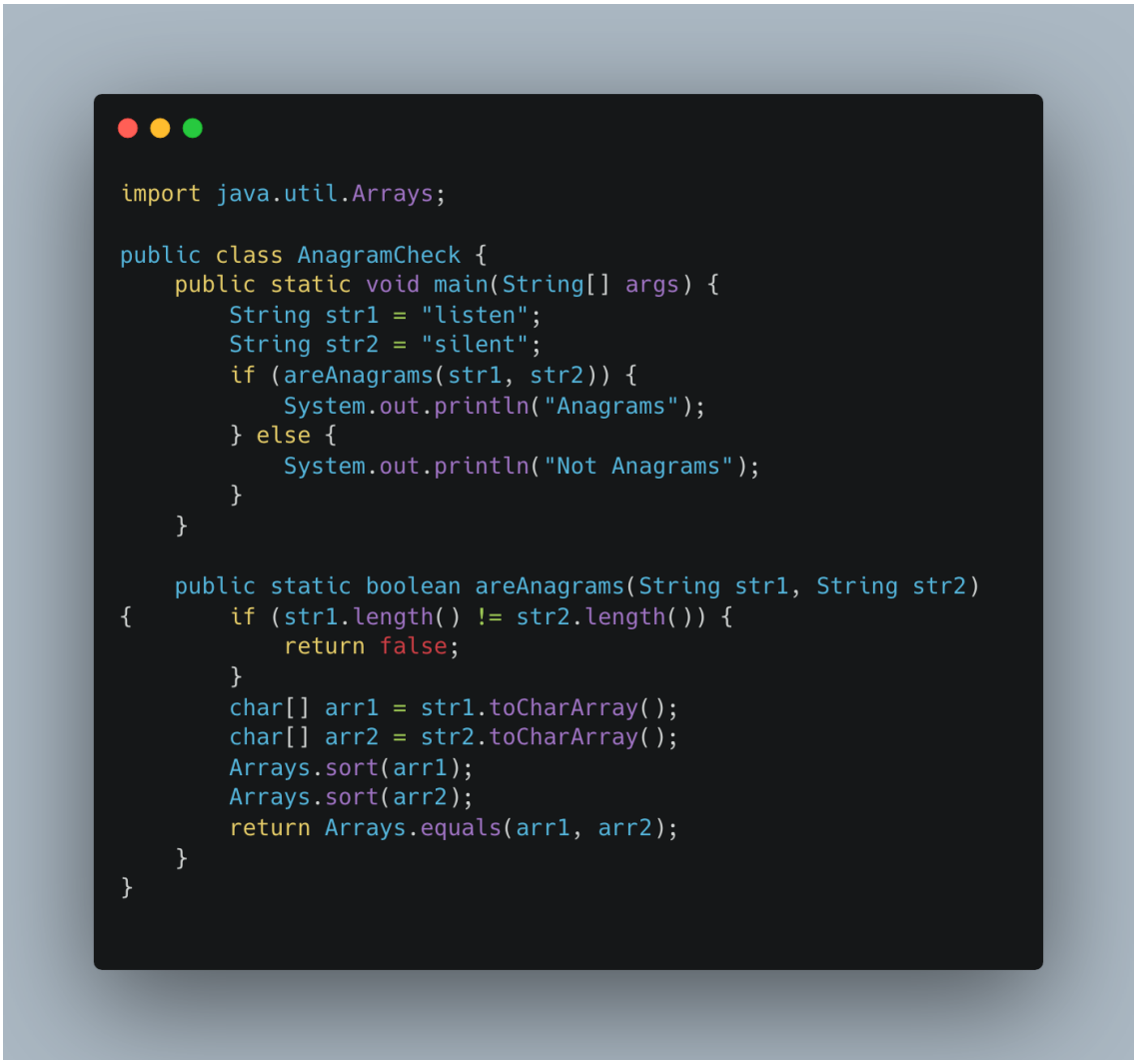
13. Write a java program to filter odd and even numbers from a list using java 8 streams?

```java
import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;

public class OddEvenFilter {
    public static void main(String[] args) {
        List<Integer> numbers = new ArrayList<>();
        numbers.add(1);
        numbers.add(2);
        numbers.add(3);
        numbers.add(4);
        numbers.add(5);
        List<Integer> evenNumbers = numbers.stream().filter(n -> n % 2 ==
0).collect(Collectors.toList());
        List<Integer> oddNumbers = numbers.stream().filter(n -> n % 2 !=
0).collect(Collectors.toList());
        System.out.println("Even numbers: " + evenNumbers);
        System.out.println("Odd numbers: " + oddNumbers);
    }
}
```

14. Write a java program to make all zeros to end in an array? Ex: [1,0,2,3,0,1,8] ans:[1,2,3,1,8,0,0]

```java
public class MoveZerosToEnd {
    public static void main(String[] args) {
        int[] arr = {1, 0, 2, 3, 0, 1, 8};
        int count = 0;
        for (int i = 0; i < arr.length; i++)
{           if (arr[i] != 0) {
                arr[count++] = arr[i];
            }
        }
        while (count < arr.length) {
            arr[count++] = 0;
        }
        for (int num : arr) {
            System.out.print(num + " ");
        }
    }
}
```

15. Write a java program to check if two strings are anagrams or not?



```java
import java.util.Arrays;

public class AnagramCheck {
    public static void main(String[] args) {
        String str1 = "listen";
        String str2 = "silent";
        if (areAnagrams(str1, str2)) {
            System.out.println("Anagrams");
        } else {
            System.out.println("Not Anagrams");
        }
    }

    public static boolean areAnagrams(String str1, String str2)
{       if (str1.length() != str2.length()) {
            return false;
        }
        char[] arr1 = str1.toCharArray();
        char[] arr2 = str2.toCharArray();
        Arrays.sort(arr1);
        Arrays.sort(arr2);
        return Arrays.equals(arr1, arr2);
    }
}
```

16. Write a java program to find if a given number is anagram or not?

```java
public class AnagramNumberCheck {
    public static void main(String[] args) {
        int num1 = 123;
        int num2 = 321;
        if (areAnagrams(num1, num2)) {
            System.out.println("Anagrams");
        } else {
            System.out.println("Not Anagrams");
        }
    }

    public static boolean areAnagrams(int num1, int num2) {
        String str1 = String.valueOf(num1);
        String str2 = String.valueOf(num2);
        return areAnagrams(str1, str2);
    }

    public static boolean areAnagrams(String str1, String str2)
{       if (str1.length() != str2.length()) {
            return false;
        }
        char[] arr1 = str1.toCharArray();
        char[] arr2 = str2.toCharArray();
        Arrays.sort(arr1);
        Arrays.sort(arr2);
        return Arrays.equals(arr1, arr2);
    }
}
```

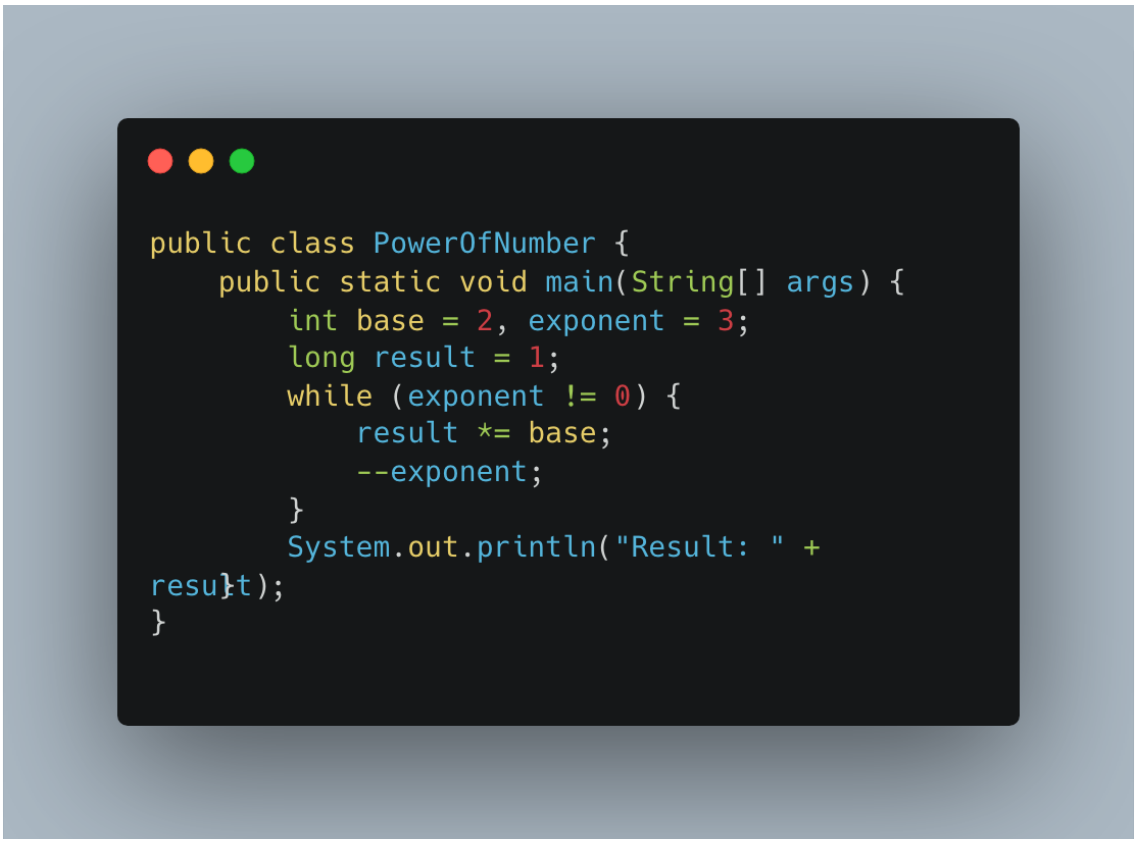
17. Write a java program to remove all white spaces from a string without using the replace method?

```java
public class RemoveSpaces {
    public static void main(String[] args) {
        String str = "Hello World";
        char[] chars = str.toCharArray();
        StringBuilder sb = new StringBuilder();
        for (char c : chars) {
            if (!Character.isWhitespace(c)) {
                sb.append(c);
            }
        }
        System.out.println("String without spaces: " +
sb.toString());
}
```

18. Write a java program to read and write in file using i/o?

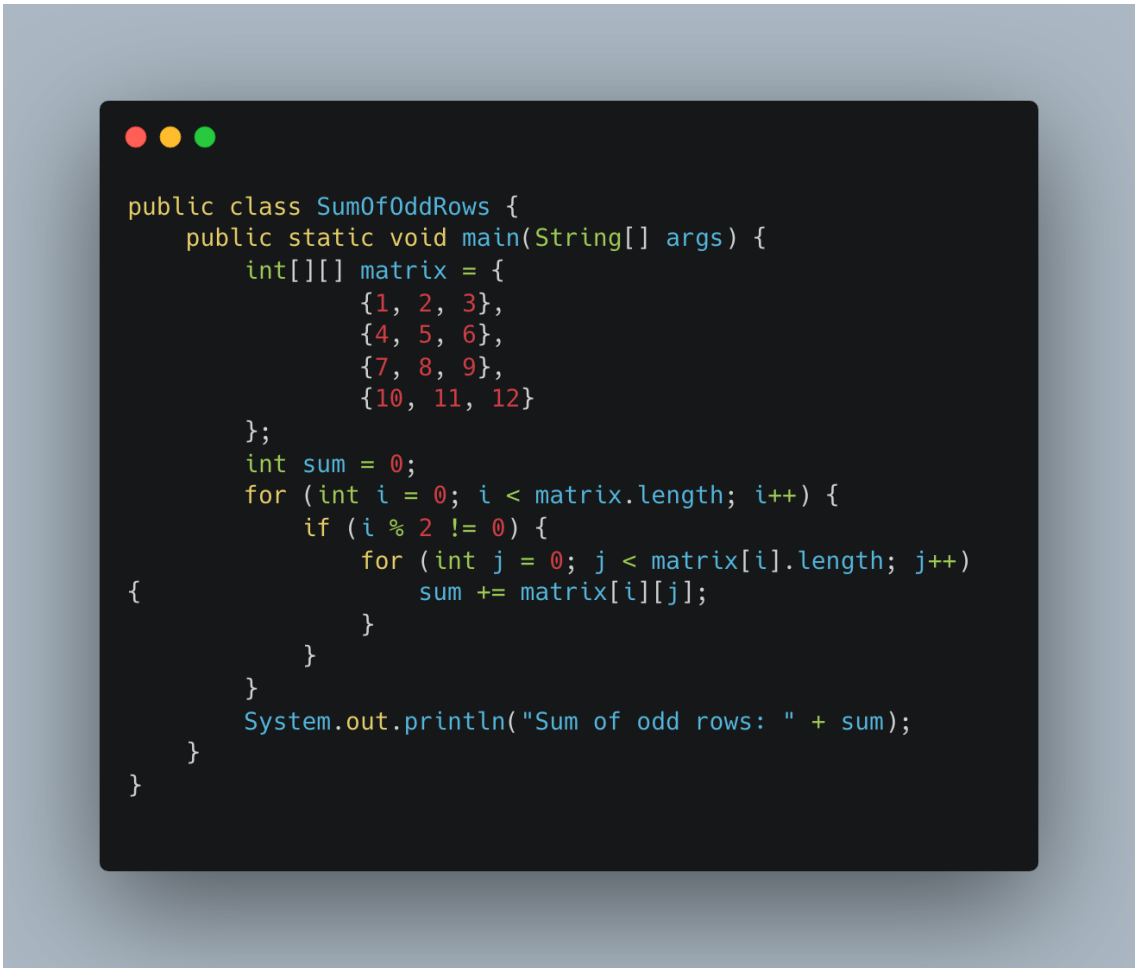
```java
import java.io.*;

public class FileReadWrite {
    public static void main(String[] args) {
        String filename = "test.txt";
        try {
            FileWriter writer = new FileWriter(filename);
            writer.write("Hello World!");
            writer.close();
            FileReader reader = new FileReader(filename);
            BufferedReader bufferedReader = new
BufferedReader(reader);
            String line;
            while ((line = bufferedReader.readLine()) != null) {
                System.out.println(line);
            }
            reader.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

19. Write a java program to find power of a number without using inbuilt functions?



```java
public class PowerOfNumber {
    public static void main(String[] args) {
        int base = 2, exponent = 3;
        long result = 1;
        while (exponent != 0) {
            result *= base;
            --exponent;
        }
        System.out.println("Result: " +
result);
}
```

20. Write a java program print sum of odd rows in a 2d matrix?



```java
public class SumOfOddRows {
    public static void main(String[] args) {
        int[][] matrix = {
                {1, 2, 3},
                {4, 5, 6},
                {7, 8, 9},
                {10, 11, 12}
        };
        int sum = 0;
        for (int i = 0; i < matrix.length; i++) {
            if (i % 2 != 0) {
                for (int j = 0; j < matrix[i].length; j++)
{                   sum += matrix[i][j];
                }
            }
        }
        System.out.println("Sum of odd rows: " + sum);
    }
}
```