OOPS-LAB-QS

1. **Write Java Program to print all the Prime numbers and Armstrong Numbers in a range given by the user. Take input through BufferedReader class object.**

   Code:

```
import java.io.*;


public class Q1 {


   public static void main(String[] args) throws IOException {
      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));



      System.out.print("Enter the start of the range: ");
      int start = Integer.parseInt(br.readLine());

      System.out.print("Enter the end of the range: ");
      int end = Integer.parseInt(br.readLine());

      System.out.println("\nPrime numbers between " + start + " and " + end + " are:");
      for (int i = start; i <= end; i++) {
         if (isPrime(i)) {
            System.out.print(i + " ");
         }
      }

      System.out.println("\n\nArmstrong numbers between " + start + " and " + end + " are:");
      for (int i = start; i <= end; i++) {
         if (isArmstrong(i)) {
            System.out.print(i + " ");
         }
```

```
        }
    }

    public static boolean isPrime(int n) {
        if (n < 2) return false;
        for (int i = 2; i <= Math.sqrt(n); i++) {
            if (n % i == 0)
                return false;
        }
        return true;
    }

    public static boolean isArmstrong(int num) {
        int original = num;
        int sum = 0;
        int digits = String.valueOf(num).length();

        while (num > 0) {
            int digit = num % 10;
            sum += Math.pow(digit, digits);
            num /= 10;
        }
        return sum == original;
    }
}
```

```
PS C:\Users\skari\Java-assign> cd "c:\Users\skari\Java-assign\" ; if ($?)
{ javac Q1.java } ; if ($?) { java Q1 }
Enter the start of the range: 2
Enter the end of the range: 10

Prime numbers between 2 and 10 are:
2 3 5 7

Armstrong numbers between 2 and 10 are:
2 3 4 5 6 7 8 9
PS C:\Users\skari\Java-assign>
```

2. **Write a Java Program to create a class called Complex. Within the class define the relevant data variables as well as methods to sum, subtract, multiply and divide two complex numbers. Create Complex class objects in main method and perform summation and multiplication.**

Code:

```
class Complex {
    double real;
    double imag;



    Complex(double r, double i) {
        this.real = r;
        this.imag = i;
    }


    Complex add(Complex other) {
        return new Complex(this.real + other.real, this.imag + other.imag);
    }


    Complex subtract(Complex other) {
        return new Complex(this.real - other.real, this.imag - other.imag);
```

```
    }

    Complex multiply(Complex other) {
        double r = this.real * other.real - this.imag * other.imag;
        double i = this.real * other.imag + this.imag * other.real;
        return new Complex(r, i);
    }


    Complex divide(Complex other) {
        double denominator = other.real * other.real + other.imag * other.imag;
        double r = (this.real * other.real + this.imag * other.imag) / denominator;
        double i = (this.imag * other.real - this.real * other.imag) / denominator;
        return new Complex(r, i);
    }

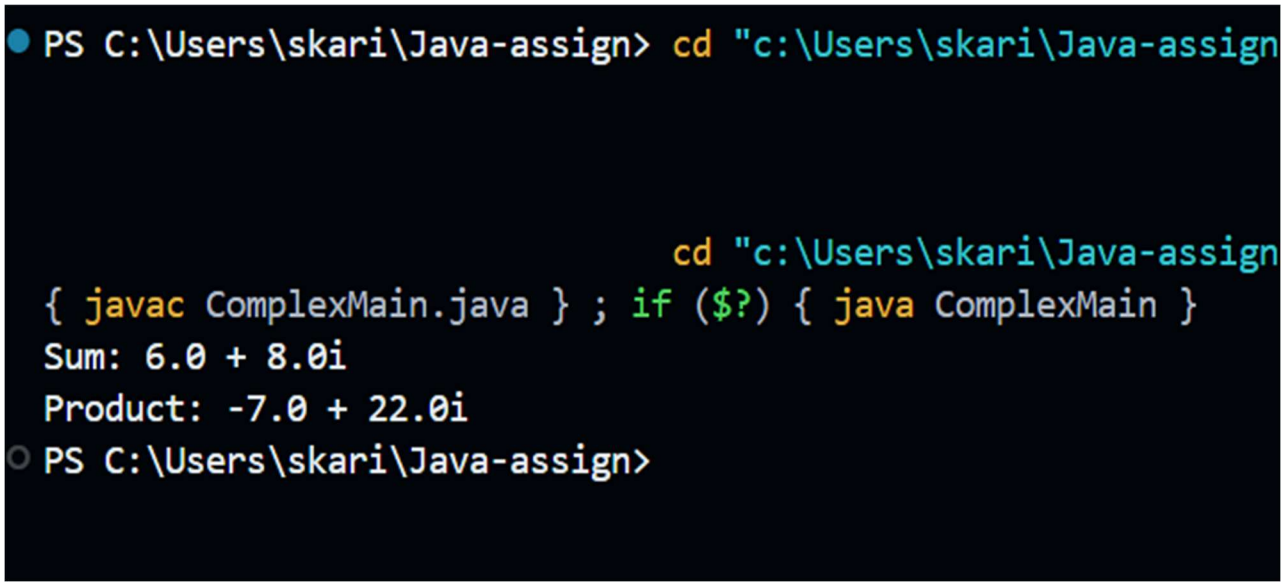

    void display() {
        if (imag >= 0)
            System.out.println(real + " + " + imag + "i");
        else
            System.out.println(real + " - " + (-imag) + "i");
    }
}

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

        Complex c1 = new Complex(4, 5);  // 4 + 5i
        Complex c2 = new Complex(2, 3);  // 2 + 3i
```

```
        Complex sum = c1.add(c2);
        System.out.print("Sum: ");
        sum.display();


        Complex product = c1.multiply(c2);
        System.out.print("Product: ");
        product.display();
    }
}
```



3. **Write a Java Program to take input of seven city names through command line argument. Sort these cities using selection sort in alphabetic order by using compareTo method.**

   Code:

```
public class CitySorter {
    public static void main(String[] args) {
        int n=args.length;
        if (n != 7) {
            System.out.println("Please provide exactly 7 city names as command-line arguments.");
```

```
            return;
        }



        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - 1 - i; j++) {
                if (args[j].compareToIgnoreCase(args[j + 1]) > 0) {

                    String temp = args[j];
                    args[j] = args[j + 1];
                    args[j + 1] = temp;
                }
            }
        }



        System.out.println("Cities in alphabetical order:");
        for (String city : args) {
            System.out.println(city);
        }
    }
}
```

```
PS C:\Users\skari\Java-assign> javac CitySorter.java
PS C:\Users\skari\Java-assign>  java CitySorter kolkata mumbai chennai jam
shedpur patna bengaluru guwahati
Cities in alphabetical order:
bengaluru
chennai
guwahati
jamshedpur
kolkata
mumbai
patna
PS C:\Users\skari\Java-assign>
```

4. **Write a Java Program to take a string input by using BufferedReader class object. Reverse this string and print it.**

Code:

```
import java.io.*;


public class ReverseString {
   public static void main(String[] args) throws Exception {


      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));


      System.out.print("Enter a string: ");
      String input = br.readLine();


      String reversed = "";


      for (int i = input.length() - 1; i >= 0; i--) {
         reversed += input.charAt(i);
      }


      System.out.println("Reversed string: " + reversed);
```

```
    }
}
```

```
                                cd "c:\Users\skari\Java-assign\" ; if ($?)
{ javac ReverseString.java } ; if ($?) { java ReverseString }
Enter a string: Kolkata
Reversed string: ataKloK
PS C:\Users\skari\Java-assign>
```

5. **Define an Abstract class Shape and its two Subclasses Square and Circle. Use method overriding to calculate the area of the square and circle.**

Code:

```
abstract class Shape {
    abstract void area();
}


class Square extends Shape {
    double side;

    Square(double side) {
        this.side = side;
    }

    void area() {
        double result = side * side;
        System.out.println("Area of Square: " + result);
    }
}
```

```
class Circle extends Shape {
    double radius;

    Circle(double radius) {
        this.radius = radius;
    }

    void area() {
        double result = 3.1416 * radius * radius;
        System.out.println("Area of Circle: " + result);
    }
}

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

        Shape s1 = new Square(5);
        s1.area();

        Shape s2 = new Circle(3);
        s2.area();
    }
}
```

```
PS C:\Users\skari\Java-assign> cd "c:\Users\skari\
{ javac Main.java } ; if ($?) { java Main }
Area of Square: 25.0
Area of Circle: 28.2744
PS C:\Users\skari\Java-assign>
```

6. **Define a class Shape and within the class define overloaded methods calculateArea to calculate the area of Circle and Square. Use BufferedReader for input.**

Code:

```
import java.io.*;

class Shape {


   void calculateArea(double radius) {

      double area = 3.1416 * radius * radius;

      System.out.println("Area of Circle: " + area);

   }


   void calculateArea(int side) {

      int area = side * side;

      System.out.println("Area of Square: " + area);

   }

}


public class Main {

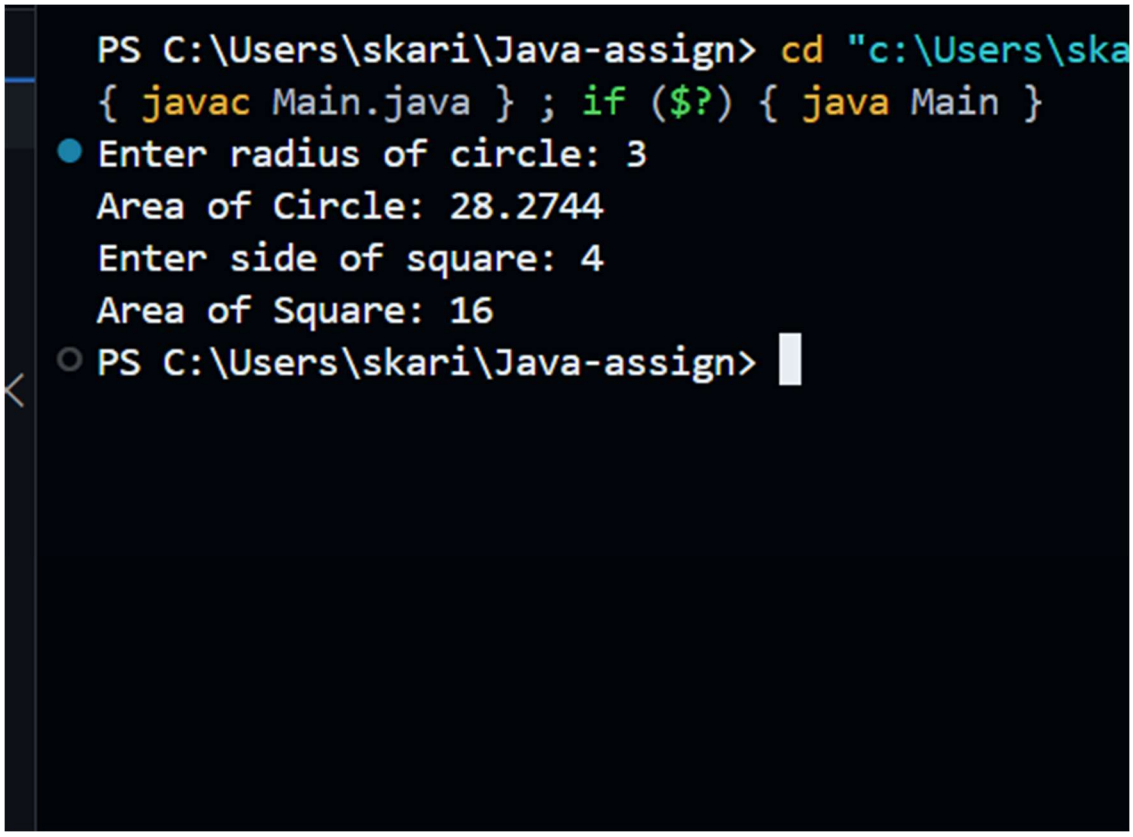
   public static void main(String[] args) throws Exception {


      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
```

```
        Shape shape = new Shape();

        System.out.print("Enter radius of circle: ");
        double radius = Double.parseDouble(br.readLine());
        shape.calculateArea(radius);

        System.out.print("Enter side of square: ");
        int side = Integer.parseInt(br.readLine());
        shape.calculateArea(side);
    }
}
```



7. **Define an interface Shape with abstract method calculatePerimeter. Implement the interface in two classes called Square and Circle. Create its objects to calculate perimeter and area.**

   Code:

```
interface Shape {
    void calculatePerimeter();
```

```
}

class Square implements Shape {
    int side;

    Square(int side) {
        this.side = side;
    }

    public void calculatePerimeter() {
        System.out.println("Square Perimeter: " + (4 * side));
    }

    void calculateArea() {
        System.out.println("Square Area: " + (side * side));
    }
}

class Circle implements Shape {
    double radius;

    Circle(double radius) {
        this.radius = radius;
    }

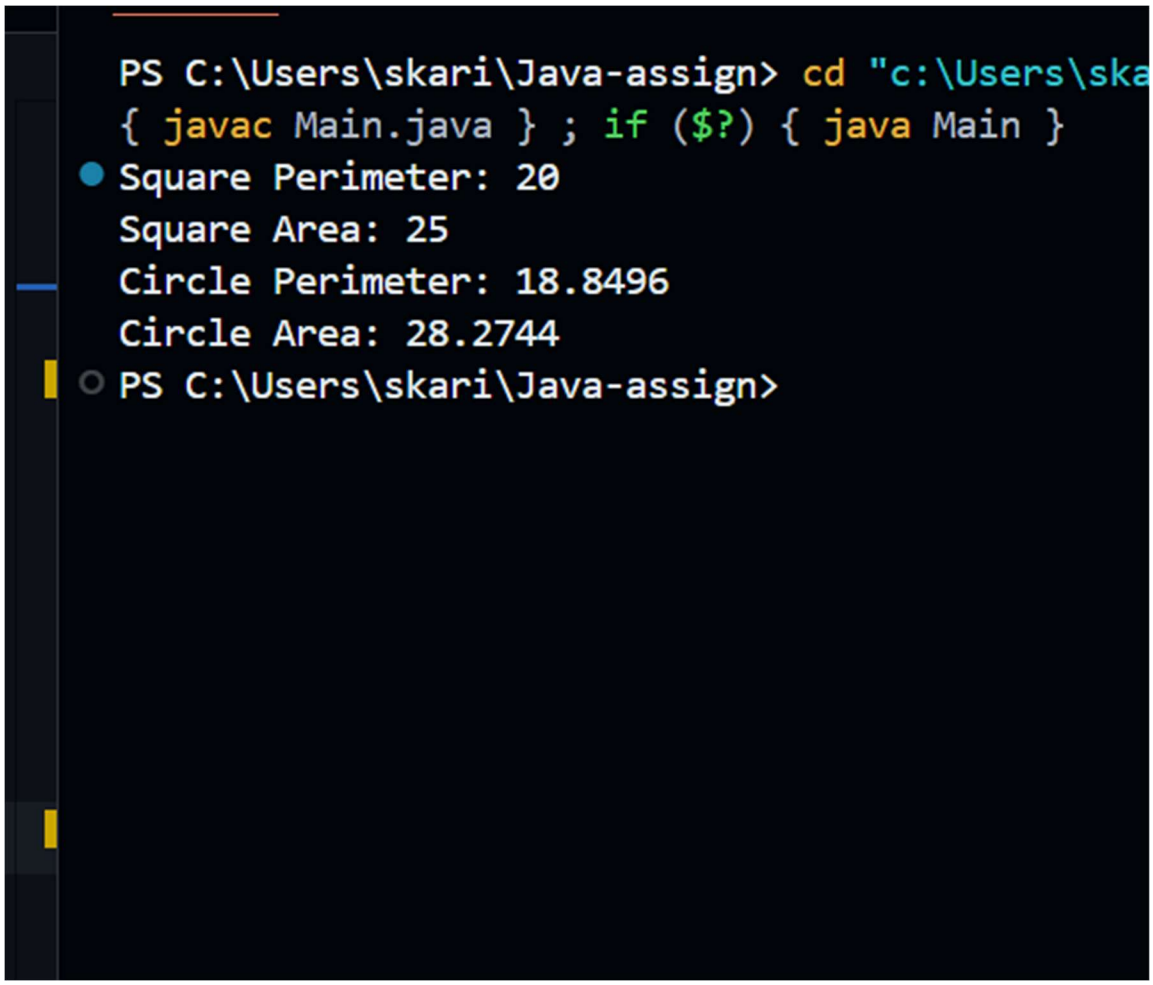
    public void calculatePerimeter() {
        System.out.println("Circle Perimeter: " + (2 * 3.1416 * radius));
    }

    void calculateArea() {
```

```
        System.out.println("Circle Area: " + (3.1416 * radius * radius));
    }
}

public class Main {
    public static void main(String[] args) {
        Square s = new Square(5);
        s.calculatePerimeter();
        s.calculateArea();

        Circle c = new Circle(3);
        c.calculatePerimeter();
        c.calculateArea();
    }
}
```



8. **Use multithreading to create two threads Fibonacci and Prime. Fibonacci thread will display the series for the n terms with 2000ms delay and Prime thread will display prime numbers of a given range with 5000ms delay.**

Code:

```
class FibonacciThread extends Thread {
    int n;

    FibonacciThread(int n) {
        this.n = n;
    }

    public void run() {
        int a = 0, b = 1;
        for (int i = 0; i < n; i++) {
            System.out.print(a + " ");
            int c = a + b;
```

```
            a = b;
            b = c;
            try {
                Thread.sleep(2000);
            } catch (Exception e) {}
        }
        System.out.println();
    }
}

class PrimeThread extends Thread {
    int start, end;

    PrimeThread(int start, int end) {
        this.start = start;
        this.end = end;
    }

    public void run() {
        for (int i = start; i <= end; i++) {
            if (isPrime(i)) {
                System.out.print(i + " ");
                try {
                    Thread.sleep(5000);
                } catch (Exception e) {}
            }
        }
        System.out.println();
    }
```

```
    boolean isPrime(int n) {
        if (n < 2) return false;
        for (int i = 2; i <= Math.sqrt(n); i++) {
            if (n % i == 0) return false;
        }
        return true;
    }
}

public class ThreadsMain {
    public static void main(String[] args) {
        FibonacciThread fib = new FibonacciThread(5);
        PrimeThread prime = new PrimeThread(10, 30);

        fib.start();
        prime.start();
    }
}
```



9. **Write a Java program to handle "divide by zero exception" with the following message – "Exception Occurred" if the quotient is -1. Otherwise print the result in finally block.**

Code:

```
public class DivideByZero {
    public static void main(String[] args) {
        int a = 10;
        int b = 0;

        try {
            int result = a / b;
            if (result == -1) {
                throw new ArithmeticException("Exception Occurred");
            }
        } catch (ArithmeticException e) {
            System.out.println("Exception Occurred");
        } finally {
            if (b != 0) {
                int result = a / b;
```

```
            if (result != -1) {
                System.out.println("Quotient: " + result);
            }
        }
    }
}
```



10. **Define a class Students with marks as one of the data members. In the main method create an array of Student Object and calculate the obtained average marks by all the students. Take input using Scanner class object.**

    Code:

```
import java.util.Scanner;

class Student {
    int marks;

    Student(int marks) {
        this.marks = marks;
    }
}

public class StudentsAverage {
```

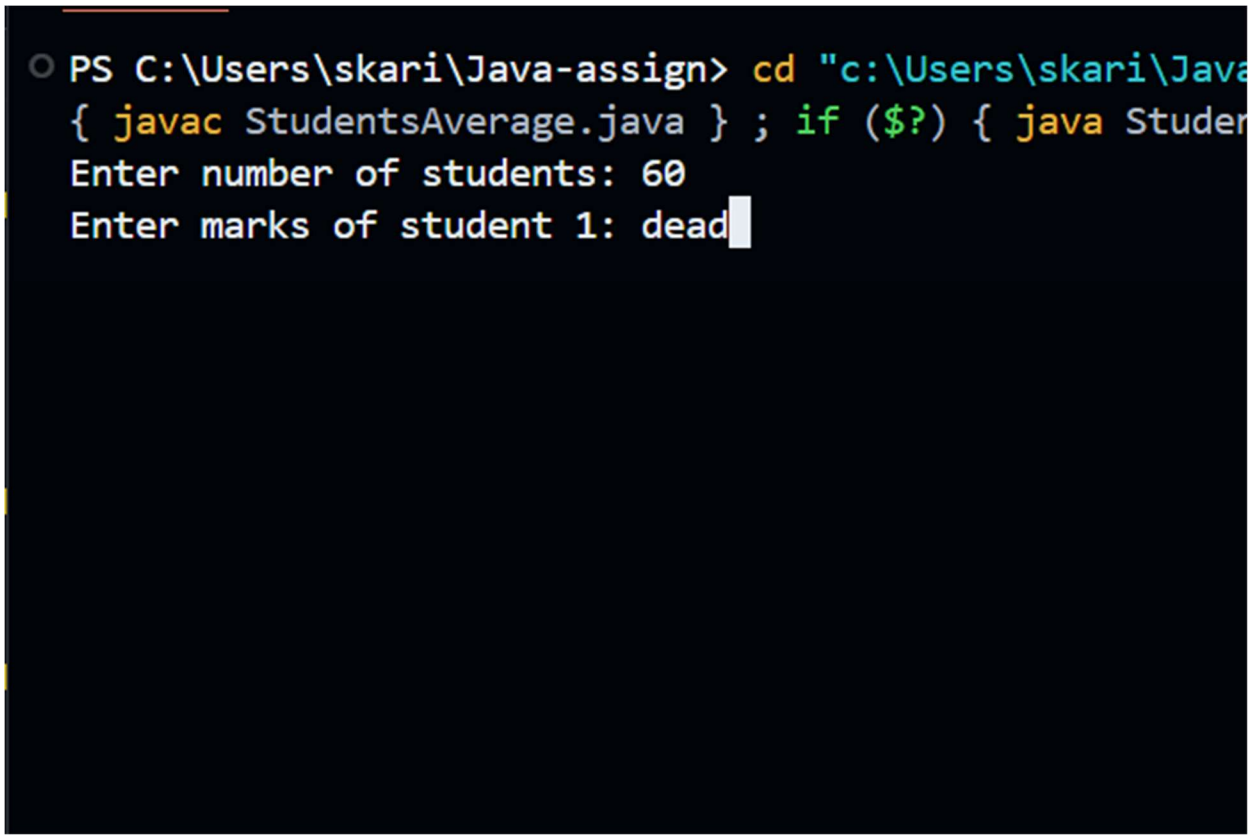
```
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter number of students: ");
        int n = sc.nextInt();

        Student[] students = new Student[n];
        int total = 0;

        for (int i = 0; i < n; i++) {
            System.out.print("Enter marks of student " + (i + 1) + ": ");
            int marks = sc.nextInt();
            students[i] = new Student(marks);
            total += marks;
        }

        double avg = (double) total / n;
        System.out.println("Average Marks: " + avg);
    }
}
```



🟢 BASIC JAVA TOPICS

✅ 1. Access Specifiers

- **public**: accessible everywhere
- **private**: accessible within the same class
- **protected**: accessible within the same package and by subclasses
- **default** (no modifier): accessible within the same package

Q1. What is the difference between private and protected?
A: private is only accessible within the same class, while protected can be accessed in subclasses and same package.

✅ 2. Class and Object

- **Class** is a blueprint
- **Object** is an instance of the class

Q2. How is an object created in Java?
A: ClassName obj = new ClassName();

✅ **3. Constructors**

- Used to initialize objects
- Same name as class, no return type

Q3. Can constructors be overloaded?
A: Yes, by changing the number or type of parameters.

✅ **4. Static Keyword**

- Belongs to class, not object
- Can be used with variables, methods, and blocks

Q4. What is a static method?
A: A method that can be called without creating an object.

✅ **5. this Keyword**

- Refers to current class object

Q5. Why do we use this keyword?
A: To avoid confusion between class variables and parameters with the same name.

🟡 **OOP CONCEPTS**

✅ **6. Encapsulation**

- Wrapping data and methods together
- Achieved using private variables and public getters/setters

Q6. How is encapsulation achieved in Java?
A: By making variables private and accessing them via public getters and setters.

✅ **7. Inheritance**

- Reuse code from parent class
- Use extends keyword

Q7. Can Java support multiple inheritance?
A: Not with classes, but yes through interfaces.

✅ 8. Polymorphism

- Method overloading (compile-time)
- Method overriding (runtime)

Q8. What is method overloading and overriding?
A: Overloading is defining multiple methods with the same name and different parameters. Overriding means redefining a method in a subclass.

✅ 9. Abstraction

- Hiding internal details and showing only functionality
- Achieved using **abstract classes** or **interfaces**

Q9. What is the difference between abstract class and interface?
A: Abstract class can have both abstract and non-abstract methods. Interface can only have abstract methods (Java 7), or also default/static (Java 8+).

✅ 10. Interface

- Defines contract
- All methods are public and abstract by default

Q10. Can a class implement multiple interfaces?
A: Yes, this is how Java achieves multiple inheritance.

🔵 ADVANCED JAVA TOPICS

✅ 11. Exception Handling

- Handled using try-catch-finally
- Keywords: try, catch, finally, throw, throws

Q11. Difference between checked and unchecked exceptions?
A: Checked exceptions are checked at compile-time (e.g., IOException), unchecked are checked at runtime (e.g., NullPointerException).

✅ 12. Multithreading

- Concurrency using Thread class or Runnable interface

Q12. How do you create a thread in Java?
A: Either by extending Thread class or implementing Runnable interface.

✅ 13. String Handling

- String is immutable
- Use String, StringBuilder, or StringBuffer

Q13. Difference between String and StringBuilder?
A: String is immutable, StringBuilder is mutable and faster for string operations.

✅ 14. BufferedReader & Scanner

- Used for input
- BufferedReader for fast reading (used with InputStreamReader)
- Scanner is more user-friendly

Q14. Which is better: BufferedReader or Scanner?
A: BufferedReader is faster, but Scanner is easier to use.

✅ 15. Arrays & ArrayList

- Array: fixed size
- ArrayList: dynamic size, from java.util package

Q15. Can we store different types in an ArrayList?
A: Yes, by using ArrayList<Object>, but not recommended.

🧠 BONUS QUESTIONS FOR VIVA

Question	Answer
Q: What is JVM?	JVM (Java Virtual Machine) runs Java bytecode and provides platform independence.
Q: What is JDK?	Java Development Kit – includes compiler, JVM, and tools.
Q: What is the difference between == and .equals()?	== compares references; .equals() compares content.
Q: What are wrapper classes?	Classes that convert primitive types to objects (e.g., int → Integer).
Q: What is final keyword?	Used to declare constants, prevent method overriding, and inheritance.
Q: What is the default value of an uninitialized variable?	Local variables have no default value. Instance variables get default values (e.g., 0, null).

🔁 **OOPs + Java Viva Questions & Answers (Extended Set)**

🔹 **General Java Basics**

Q1. What is Java?
A: Java is a high-level, platform-independent, object-oriented programming language.

Q2. What is the JRE, JDK, and JVM?
A:

- **JDK (Java Development Kit)**: Includes tools to develop Java programs.
- **JRE (Java Runtime Environment)**: Environment to run Java programs.
- **JVM (Java Virtual Machine)**: Executes the compiled Java bytecode.

Q3. What are the primitive data types in Java?
A: byte, short, int, long, float, double, char, boolean.

Q4. What is type casting in Java?
A: Converting one data type into another. Two types: implicit and explicit casting.

Q5. What is the default value of boolean and int?
A: boolean → false, int → 0.

- **Classes, Objects, and Methods**

Q6. Can a Java class have multiple constructors?
A: Yes, this is called constructor overloading.

Q7. Can we overload the main method?
A: Yes, but the JVM always looks for the standard main(String[] args) method to start the program.

Q8. What is a method signature?
A: Method name + parameter list (not return type).

Q9. What is the difference between instance variable and class variable?
A: Instance variable belongs to object; class variable is static and shared by all objects.

- **Static, Final, and This**

Q10. Can a constructor be static?
A: No, constructors cannot be static.

Q11. What is a static block?
A: A block that runs once when the class is loaded.

Q12. What is the use of the final keyword in Java?
A: Final can be used to prevent method overriding, inheritance, and reassignment of variables.

Q13. What is the difference between this and super keyword?
A: this refers to the current class, super refers to the parent class.

- **Inheritance and Polymorphism**

Q14. What is hierarchical inheritance?
A: When multiple classes inherit from a single superclass.

Q15. What is runtime polymorphism?
A: Method overriding, where the method call is resolved at runtime.

Q16. Can private methods be overridden?
A: No, private methods are not inherited.

Q17. What is upcasting and downcasting?
A: Upcasting: Subclass → Superclass
Downcasting: Superclass → Subclass (needs explicit cast)

◆ Abstraction and Interface

Q18. Can an abstract class have a constructor?
A: Yes, it can be used to initialize data.

Q19. Can we create an object of an abstract class?
A: No, but we can use it as a reference.

Q20. Can interfaces have constructors?
A: No, interfaces cannot have constructors.

Q21. Can a class implement multiple interfaces?
A: Yes, Java allows multiple interface implementation.

Q22. Can we define variables in interface?
A: Yes, but they are implicitly public static final.

◆ Exception Handling

Q23. What is the difference between throw and throws?
A: throw is used to explicitly throw an exception, throws declares it in the method signature.

Q24. What is the use of finally block?
A: It executes whether or not an exception occurs.

Q25. Can we catch multiple exceptions in one catch block?
A: Yes, using multi-catch: catch (IOException | ArithmeticException e)

◆ Multithreading

Q26. Difference between Thread class and Runnable interface?
A: Thread class is inherited; Runnable is implemented. Runnable is better for multiple inheritance.

Q27. What is thread lifecycle in Java?
A: New → Runnable → Running → (Waiting/Sleeping) → Dead

Q28. What is the difference between sleep() and wait()?
A: sleep() is used for delay, wait() is used for thread communication (with notify()/notifyAll()).

Q29. What is synchronization in Java?
A: Ensures that only one thread can access a block of code at a time.

🔹 String Handling

Q30. Why are strings immutable in Java?
A: For security, caching, and synchronization.

Q31. What are differences between String, StringBuffer, and StringBuilder?
A:

- String: immutable
- StringBuffer: mutable and thread-safe
- StringBuilder: mutable but not thread-safe (faster)

🔹 Scanner vs BufferedReader

Q32. Difference between Scanner and BufferedReader?
A:

- Scanner: Easier to use, slower
- BufferedReader: Faster, requires parsing (readLine + Integer.parseInt)

🔹 Miscellaneous

Q33. What is the use of package in Java?
A: It organizes classes and avoids name conflicts.

Q34. What is garbage collection in Java?
A: Automatic memory management where unused objects are deleted.

Q35. What is the purpose of finalize() method?
A: Called before an object is garbage collected.

Q36. Can we override static methods?
A: No, static methods are bound at compile-time.

Q37. Can an interface extend another interface?
A: Yes, interfaces can extend multiple interfaces.

Q38. Can an abstract class implement an interface?
A: Yes, but it must provide implementations or remain abstract.