Solutions to 2018 ICSE Paper.

SECTION A

Question 1

(a) **Abstraction**: Abstraction is a principle in object-oriented programming that focuses on hiding the complexity and details of an object's functionality and only showing the relevant, high-level interface. For example, when using a car, you don't need to know how the engine works to drive it. Similarly, in programming, abstraction allows the user to work with an object without knowing its underlying code.

(b) **Searching vs Sorting**:

- **Searching**: It is the process of finding a specific element in a collection of data (e.g., an array or list). Examples include linear search and binary search.
- **Sorting**: It involves rearranging elements in a specific order, either ascending or descending. Examples include bubble sort, quicksort, and merge sort.

The main difference: Searching helps you **find** an element, whereas sorting helps you **arrange** elements in a particular order.

(c) **isUpperCase() vs toUpperCase()**:

- **isUpperCase()**: This method checks if a given character is in uppercase. It returns a boolean (`true` or `false`). Example: `isUpperCase('A')` returns `true` .
- **toUpperCase()**: This method converts a given character to its uppercase equivalent. Example: `toUpperCase('a')` returns `'A'` .

(d) **Private vs Public Members**:

- **Private members**: These are accessible only within the class where they are defined. Other classes cannot directly access private members.
- **Public members**: These can be accessed from any class, not just the one where they are defined.

For example, in Java:

```
class Example {
    private int x;  // only accessible inside this class
    public int y;   // accessible from any class
}
```

(e) **Classifying Datatypes**:

- **Primitive Datatypes**: These are basic data types in Java that store simple values. Examples include `int` , `char` , `boolean` , and `double` .
- **Non-Primitive Datatypes**: These are more complex types that refer to objects. They include arrays, classes, and interfaces.

(i) `char` → **Primitive**
(ii) `arrays` → **Non-Primitive**
(iii) `int` → **Primitive**
(iv) `classes` → **Non-Primitive**

Question 2

(a)
(i) `int res = 'A';` :

- The value of `res` will be the ASCII value of the character `'A'`. In the ASCII table, `'A'` corresponds to **65**.

(ii) Wrapper classes (such as `Integer`, `Character`, `Double`, etc.) are found in the **`java.lang` package**.

(b) **while vs do-while loop**:

- **while loop**: The condition is checked at the beginning of the loop. If the condition is false at the start, the loop's body will not execute at all.

```
while (condition) {
  // loop body
}
```

- **do-while loop**: The loop's body is executed at least once, regardless of the condition, as the condition is checked after executing the loop's body.

```
do {
  // loop body
} while (condition);
```

(c) Output of the code:

```
System.out.print("BEST ");
System.out.println("OF LUCK");
```

- The first line prints "BEST" without moving to the next line, while the second line prints "OF LUCK" and moves to the next line. The output will be:

```
BEST OF LUCK
```

(d) The prototype of a function `check` that takes an integer and returns a character is:

```
char check(int number);
```

(e)
(i) `endsWith()` : This function checks if a string ends with a specified suffix and returns a **boolean** value (`true` or `false`).

(ii) `log()` : This function returns the natural logarithm of a number and returns a **double** value.

Question 3

(a) The Java expression for the given formula (\frac{\sqrt{3x + x^2}}{a + b}) is:

```
Math.sqrt(3 * x + Math.pow(x, 2)) / (a + b);
```

- `Math.sqrt()` calculates the square root.
- `Math.pow()` calculates the power of a number.

(b) For the given expression `y += ++y + y-- + --y;` with `int y = 8;`, let's evaluate step by step:

1. `++y`: Pre-increment makes `y = 9`.
2. `y--`: Returns `y` as `9`, then decrements `y` to `8`.
3. `--y`: Pre-decrements `y` to `7`.

So, `y += 9 + 9 + 7;` means `y = 8 + 25 = 33`. The final value of `y` is **33**.

(c)
(i) `Math.floor(-4.7)` → This rounds the value down to the nearest integer, which is **-5.0**. (ii) `Math.ceil(3.4)` → This rounds the value up to the nearest integer, which is **4**. `Math.pow(2, 3)` → This calculates (2^3), which is **8**.

The result is `4 + 8 = 12.0`.

(d) **Characteristics of a constructor**:

1. A constructor has the **same name** as the class and does not have a return type.
2. A constructor is called **automatically** when an object is created and is used to initialize object properties.

(e) Output of the code:

```
System.out.println("Incredible" + "\n" + "world");
```

- The `\n` adds a new line between the words. The output will be:

  ```
  Incredible
  world
  ```

(f) **Switch case for the given if-else if**:

```
switch (var) {
    case 1:
        System.out.println("good");
        break;
    case 2:
        System.out.println("better");
        break;
    case 3:
        System.out.println("best");
        break;
    default:
        System.out.println("invalid");
}
```

(g)
(i) `"ACHIEVEMENT".replace('E', 'A')`: This replaces all occurrences of `'E'` with `'A'`. The result is `"ACHAIEVAMANT"`. (ii) `"DEDICATE".compareTo("DEVOTE")`: The function compares strings lexicographically, returning the difference between the first unmatched characters. Here, `'I' - 'O' = -2`. So, the result is **-2**.

(h) Output for the string array:

```
String arr[] = {"DELHI", "CHENNAI", "MUMBAI", "LUCKNOW", "JAIPUR"};
```

- `arr[0].length() > arr[3].length()` : `"DELHI".length() = 5`, `"LUCKNOW".length() = 7`. So, the result is **false**.
- `arr[4].substring(0, 3)` : This extracts the first three characters of `"JAIPUR"`, which results in **"JAI"**.

(i) **Ternary operator** equivalent of the given if-else statement:

```
discount = (bill > 10000) ? bill * 10.0 / 100 : bill * 5.0 / 100;
```

(j) **Output and loop execution**:

```
for (i = 5; i > 10; i++) {
    System.out.println(i);
}
System.out.println(i * 4);
```

- The condition `i > 10` is false when `i = 5`, so the loop does not execute.
- The statement `System.out.println(i * 4)` will output `5 * 4 = 20`. The loop executes **0 times**.

SECTION B

Question 4: RailwayTicket Class

```
class RailwayTicket {
    String name, coach;
    long mobno;
    int amt, totalamt;

    // Method to accept input from user
    void accept(String name, String coach, long mobno, int amt) {
        this.name = name;
        this.coach = coach;
        this.mobno = mobno;
        this.amt = amt;
    }

    // Method to update total amount based on coach type
    void update() {
        switch (coach) {
            case "First_AC":
                totalamt = amt + 700;
                break;
            case "Second_AC":
                totalamt = amt + 500;
                break;
            case "Third_AC":
                totalamt = amt + 250;
                break;
```

```
            default:
                totalamt = amt;
                break;
        }
    }

    // Method to display ticket details
    void display() {
        System.out.println("Name: " + name);
        System.out.println("Coach: " + coach);
        System.out.println("Total Amount: " + totalamt);
        System.out.println("Mobile: " + mobno);
    }

    public static void main(String[] args) {
        RailwayTicket ticket = new RailwayTicket

();
        ticket.accept("John", "First_AC", 9876543210L, 2000);
        ticket.update();
        ticket.display();
    }
}
```

- **accept()** takes input for customer details.
- **update()** adjusts the ticket price based on the coach type.
- **display()** shows the final ticket details, including total amount and mobile number.

Question 5: Pronic Number Check

```
import java.util.Scanner;

public class PronicNumber {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter a number:");
        int num = sc.nextInt();
        boolean isPronic = false;

        // Checking for a Pronic number
        for (int i = 0; i < num; i++) {
            if (i * (i + 1) == num) {
                isPronic = true;
                break;
            }
        }

        // Display result
        if (isPronic) {
            System.out.println(num + " is a Pronic number.");
        } else {
```

```
            System.out.println(num + " is not a Pronic number.");
        }
    }
}
```

- A **Pronic number** is the product of two consecutive integers. For example, 12 is a Pronic number because (12 = 3 \times 4).

Question 6: Capitalize First Letter of Each Word

```
import java.util.Scanner;

public class CapitalizeWords {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter a string:");
        String input = sc.nextLine();

        // Split the string into words
        String[] words = input.split(" ");
        String result = "";

        // Capitalizing the first letter of each word
        for (String word : words) {
            result += word.substring(0, 1).toUpperCase() + word.substring(1) + " ";
        }

        // Display the final result
        System.out.println(result.trim());
    }
}
```

- This program splits the input string into words, capitalizes the first letter of each word, and rejoins the words into a final string.

Question 7: Volume Overloading

```
class Volume {
    // Calculate the volume of a sphere
    double volume(double R) {
        return (4.0 / 3) * (22.0 / 7) * Math.pow(R, 3);
    }

    // Calculate the volume of a cylinder
    double volume(double H, double R) {
        return (22.0 / 7) * Math.pow(R, 2) * H;
    }

    // Calculate the volume of a cuboid
    double volume(double L, double B, double H) {
        return L * B * H;
    }
```

```
    public static void main(String[] args) {
        Volume vol = new Volume();
        System.out.println("Volume of Sphere: " + vol.volume(5));
        System.out.println("Volume of Cylinder: " + vol.volume(7, 5));
        System.out.println("Volume of Cuboid: " + vol.volume(4, 5, 6));
    }
}
```

- **Method overloading** allows multiple methods with the same name but different parameters. The program calculates the volume of a sphere, cylinder, and cuboid based on provided dimensions.

Question 8: Menu-Driven Pattern

```
import java.util.Scanner;

public class PatternMenu {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter 1 for Pattern 1 or 2 for Pattern 2:");
        int choice = sc.nextInt();

        // Pattern 1: Alphabet triangle
        switch (choice) {
            case 1:
                for (int i = 5; i >= 1; i--) {
                    for (int j = 1; j <= i; j++) {
                        System.out.print((char) (64 + j));
                    }
                    System.out.println();
                }
                break;

            // Pattern 2: Letters increasing by count
            case 2:
                char ch = 'B';
                for (int i = 1; i <= 5; i++) {
                    for (int j = 1; j <= i; j++) {
                        System.out.print(ch);
                    }
                    System.out.println();
                    ch++;
                }
                break;

            default:
                System.out.println("Invalid option.");
        }
    }
}
```

- This program lets the user choose between two patterns. If an invalid option is entered, an error message is displayed.

Question 9: Student Marks Deviation

```
import java.util.Scanner;

public class StudentMarks {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number of students:");
        int N = sc.nextInt();
        String[] names = new String[N];
        int[] totalMarks = new int[N];
        int sum = 0;

        // Input student details
        for (int i = 0; i < N; i++) {
            System.out.println("Enter name and total marks for student " + (i + 1) +
":");
            names[i] = sc.next();
            totalMarks[i] = sc.nextInt();
            sum += totalMarks[i];
        }

        // Calculate the average marks
        double average = sum / (double) N;
        System.out.println("Average Marks: " + average);

        // Calculate and display deviation for each student
        for (int i = 0; i < N; i++) {
            double deviation = totalMarks[i] - average;
            System.out.println(names[i] + "'s deviation: " + deviation);
        }
    }
}
```

- This program calculates the average marks and the deviation of each student’s marks from the average.
