Java Assignment 1 - Report

Print Report

Part 1: Leap Year or Not?

Code implementation:

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        try {
            System.out.println("Write an year to check: ");
            int year = scanner.nextInt(); // throws so wrap i
            boolean isLeapYear = (year % 4 == 0);
            if (isLeapYear)
                System.out.println(year + " is a leap year.")
            else
                System.out.println(year + " is not a leap yea
        } finally {
            scanner.close();
        }
    }
}
```

Screenshot of output:

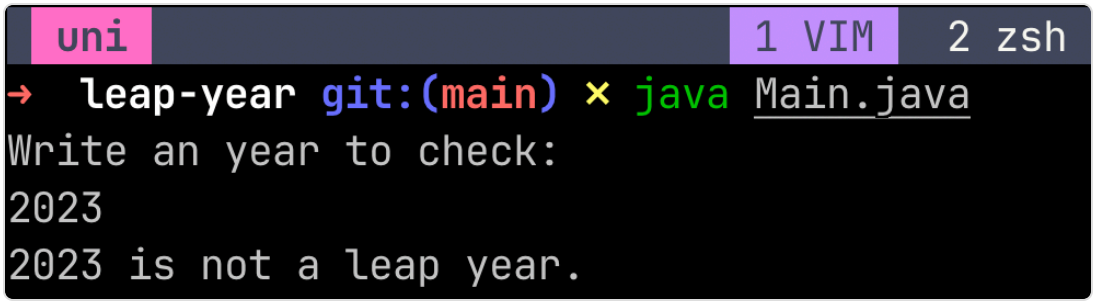

Figure 1: Output of the Leap Year Program

Flow chart:

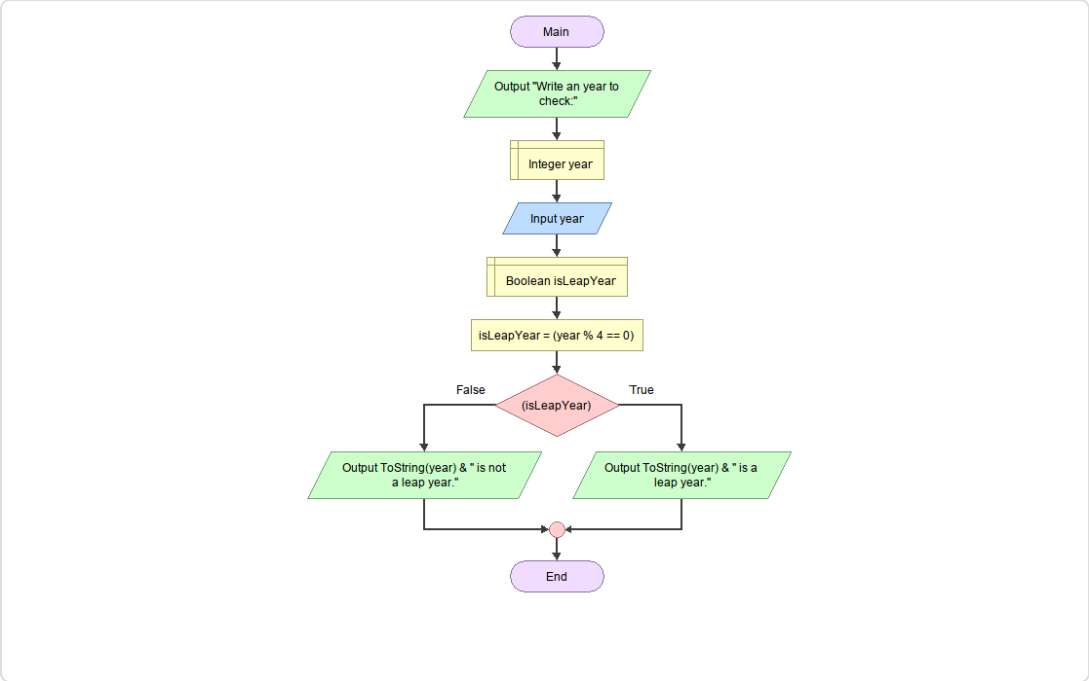

Figure 2: Flowchart for Leap Year Program

Part 2: Binary Sum

Code implementation:

```
import java.util.Scanner;
class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        try {
            System.out.println("Write the first binary number
            String binNum1Str = scanner.nextLine();
            System.out.println("Write the second binary numbe
            String binNum2Str = scanner.nextLine();
            int binaryNum1 = Integer.parseInt(binNum1Str, 2);
            int binaryNum2 = Integer.parseInt(binNum2Str, 2);
            int result = binaryNum1 + binaryNum2;
            System.out.println("The result is: " + Integer.to
        } finally {
            scanner.close();
        }
    }
}
```

Screenshot of output:

```
uni                                    1 VIM   2 zsh
➜  binary-addition git:(main) ✗ java Main.java
Write the first binary number:
10011101101
Write the second binary number:
1010111011
The result is: 11110101000
➜  binary-addition git:(main) ✗
```

Figure 3: Output of the Binary Sum Program

Flow chart:

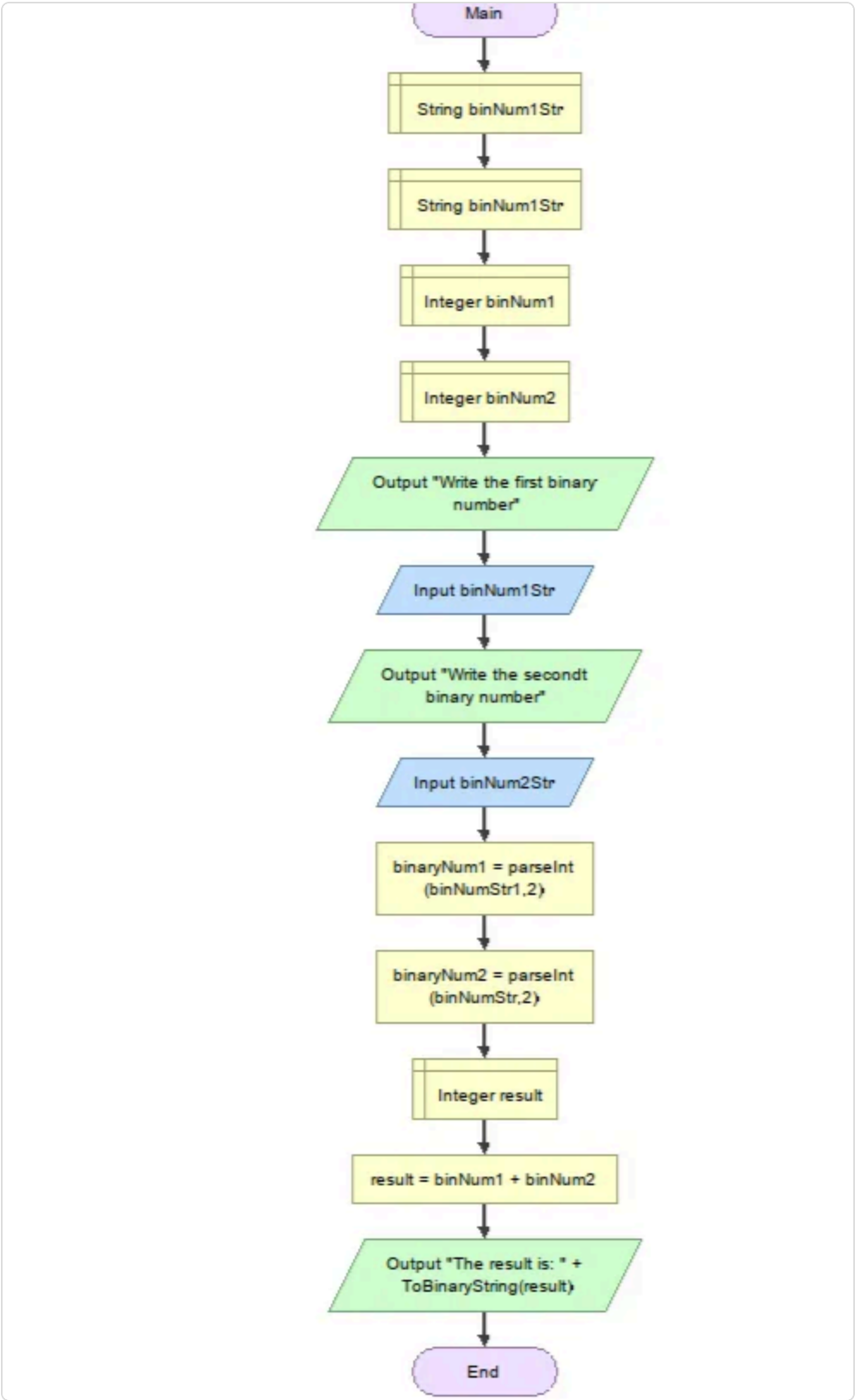

Figure 4: Flowchart for Binary Sum Program

Part 3: Print Pattern - Star/Diamond

Code implementation:

```java
import java.util.Scanner;
class Main {
    public static void main(String[] args) {
        final char printChar = '*';
        Scanner scanner = new Scanner(System.in);
        int rows;
        try (scanner) { // try-with-resources closes the scan
            rows = scanner.nextInt();
        }
        int actualRows = rows;
        if (rows % 2 == 0) {
            actualRows++;
        }
        System.out.printf("Printing diamond star pattern with
        int mid = actualRows / 2;
        for (int i = 0; i < actualRows; i++) {
            StringBuilder sb = new StringBuilder();// more me
            int spaces;
            int stars;
            if (i <= mid) {
                spaces = mid - i;
                stars = 2 * i + 1;
            } else {
                spaces = i - mid;
                stars = 2 * (actualRows - i) - 1;
            }
            for (int j = 0; j < spaces; j++) {
                sb.append(" ");
            }
            for (int j = 0; j < stars; j++) {
                sb.append(printChar);
            }
            System.out.println(sb.toString());
        }
    }
```

```
}
```

Screenshot of output:

Figure 5: Output of the Diamond Pattern Program

Flow chart:

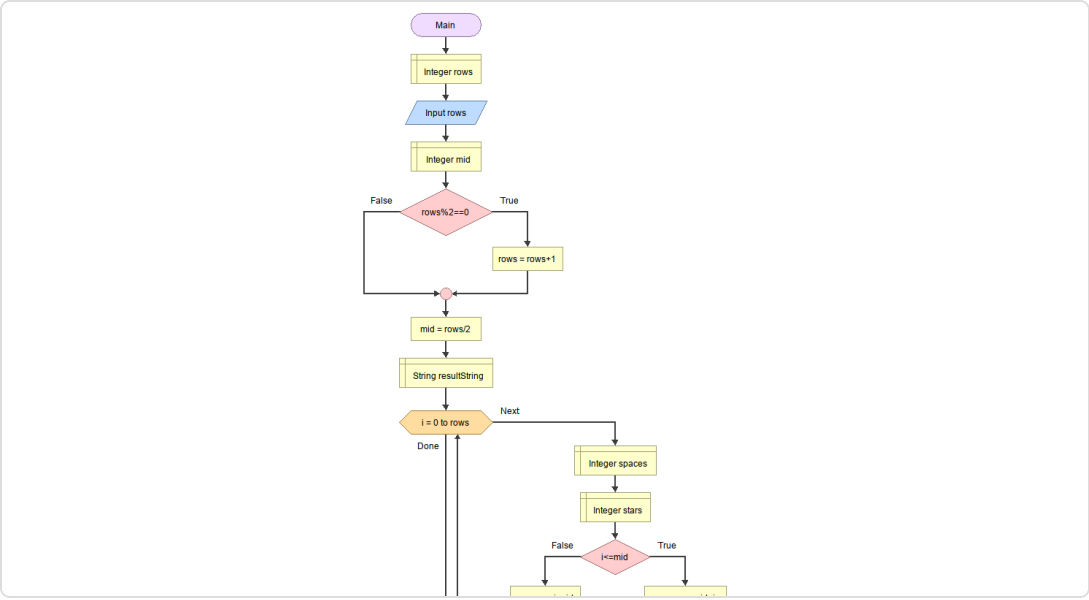

Figure 6a: Flowchart for Diamond Pattern Program (Part 1)

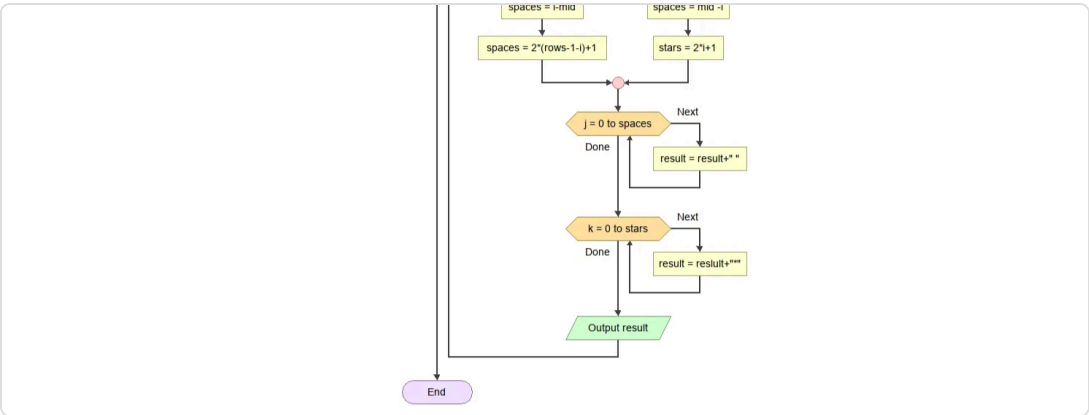

Figure 6b: Flowchart for Diamond Pattern Program (Part 2)

Part 4: Print Pattern - Steps

Code implementation:

```
class Main {
    public static void main(String[] args) {
        final char printChar = '*';
        final int rows = 5;
        System.out.printf("Printing pattern steps patter with
        for (int i = 0; i < rows; i++) {
            int spaces = rows - i;
            String stringToBePrinted = "";
            for (int k = 0; k < spaces; k++) {
                stringToBePrinted += " ";
            }
            for (int j = 0; j <= i; j++) {
                stringToBePrinted += printChar;
            }
            System.out.println(stringToBePrinted);
        }
    }
}
```

Screenshot of output:

```
uni                                  1 VIM   2 zsh
➜  print-pattern git:(main) ✗ java Main.java
Printing pattern steps patter with: 5 rows
     *
    **
   ***
  ****
 *****
➜  print-pattern git:(main) ✗
```

Figure 7: Output of the Steps Pattern Program

Flow chart:

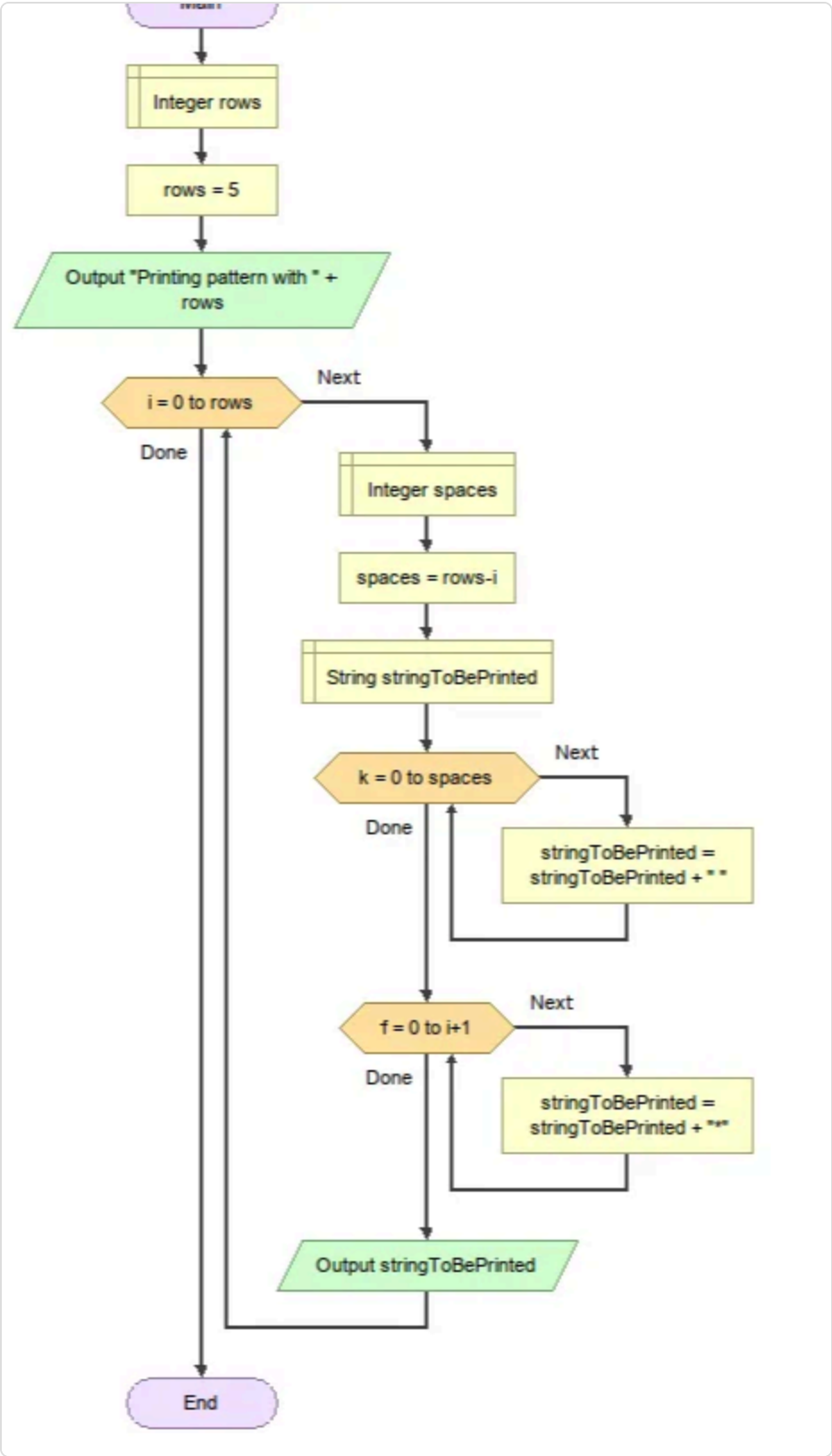

Figure 8: Flowchart for Steps Pattern Program

Java Assignment 1 Report | ☕ Java Programming