Discrete Structures

Lab (1): Sets and Bits Manipulation

Name	**ID**
Ranime Ahmed Elsayed Shehata.	21010531

Part 1: Basic Bit Operations

1. Problem statement:

You have to implement 4 bits operations, so your program might allow user to choose one of the following operations.

1. getBit(int number, int position): This function returns the bit value (an integer, 0 or 1) in the number at position position, according to its binary representation. The least significant bit in a number is position 0.

2. setBit(int number, int position):This function set the bit value (to be 1) in the number at position position, according to its binary representation. The least significant bit in a number is position 0 and return number after setting the bit.

3. clearBit(int number, int position): This function cleat the bit value (to be 0) in the number at position position, according to its binary representation. The least significant bit in a number is position 0 and return number after clearing the bit.

4. updateBit(int number, int position, boolean value): This function set the bit value according to value parameter which is false (0) or true (1) in the number at position position, according to its binary representation. The least significant bit in a number is position 0 and return number after update.

2. Used data structures:

No data structures used.

3. Algorithms documented using flow charts or pseudo code:

- **Documenting algorithms through pseudocode:**

First: 4 functions are made.

(1) **getBit() :** It takes the number and its position as an input, performs a “RIGHT SHIFT” operation on the number by value of its position. Then, performs a “BITWISE AND” to the number and “1”. Finally, it returns that a bit. The purpose is to get the value of a certain bit in a number given the number and required position. The return value is either 1 or 0.

(2) **setBit():** It takes the number and its position as an input, performs a “LEFT SHIFT” operation on the number by value of its position. Then, performs a “BITWISE OR” to the number and itself. Finally, it returns that operated on number. The purpose is to set the value of a certain bit in a number given the number and required position to “1”. The return value is the new number after setting the bit to 1.

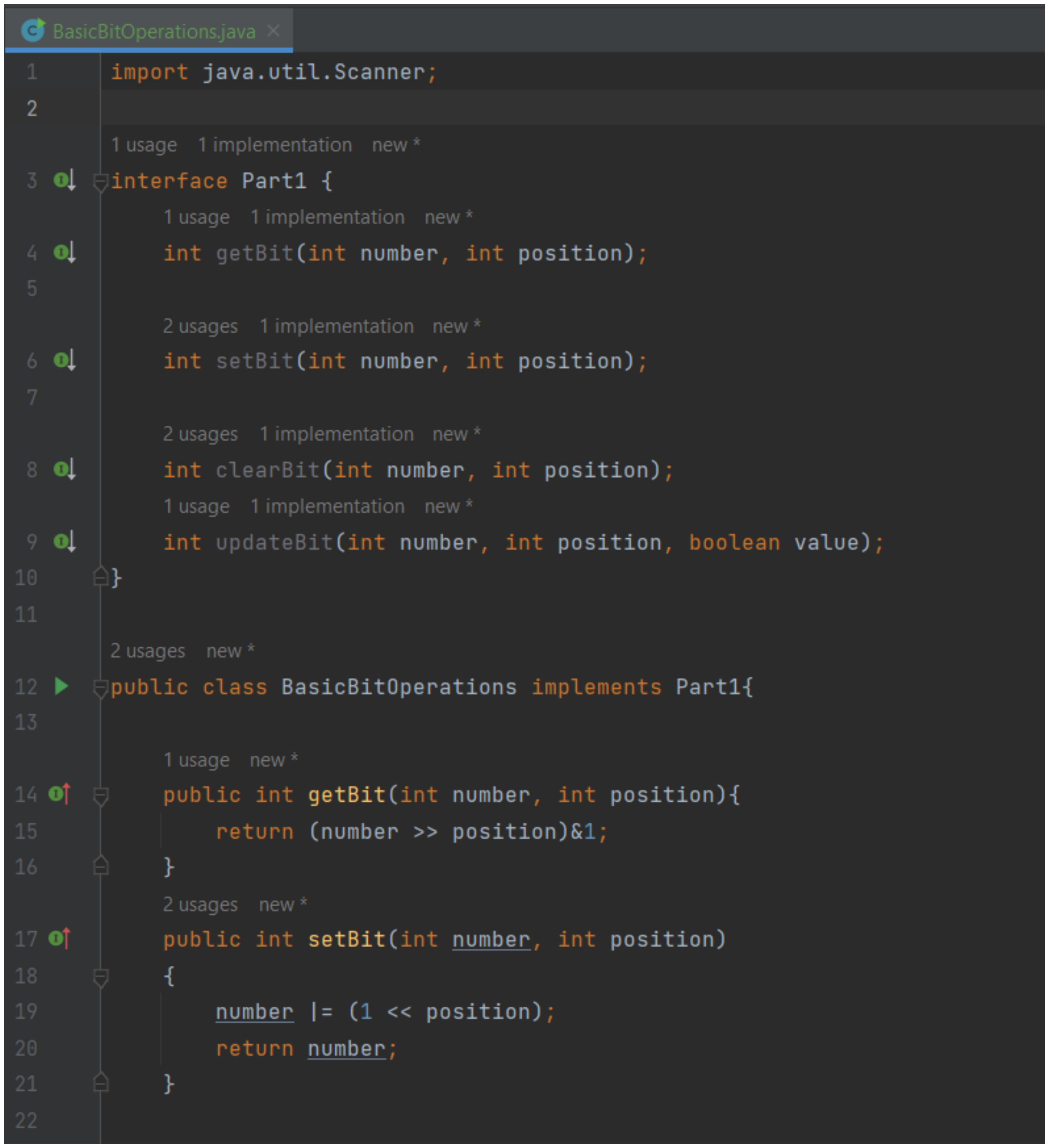
(3) **clearBit():** It takes the number and its position as an input, performs a “LEFT SHIFT” operation on the number by value of its position. Then, performs a “NEGATION” to the number and “BITWISE AND” to the number with itself. Finally, it returns that operated on number. The purpose is to clear the value of a certain bit in a number given the number and required position. The return value is the new number after setting the bit.

(4) **updateBit():** It takes the number, its position and a value, either 1 or 0. If the value was “1”, it calls the setBit() function. Whereas, if the value was “0”, it calls the clearBit() function.

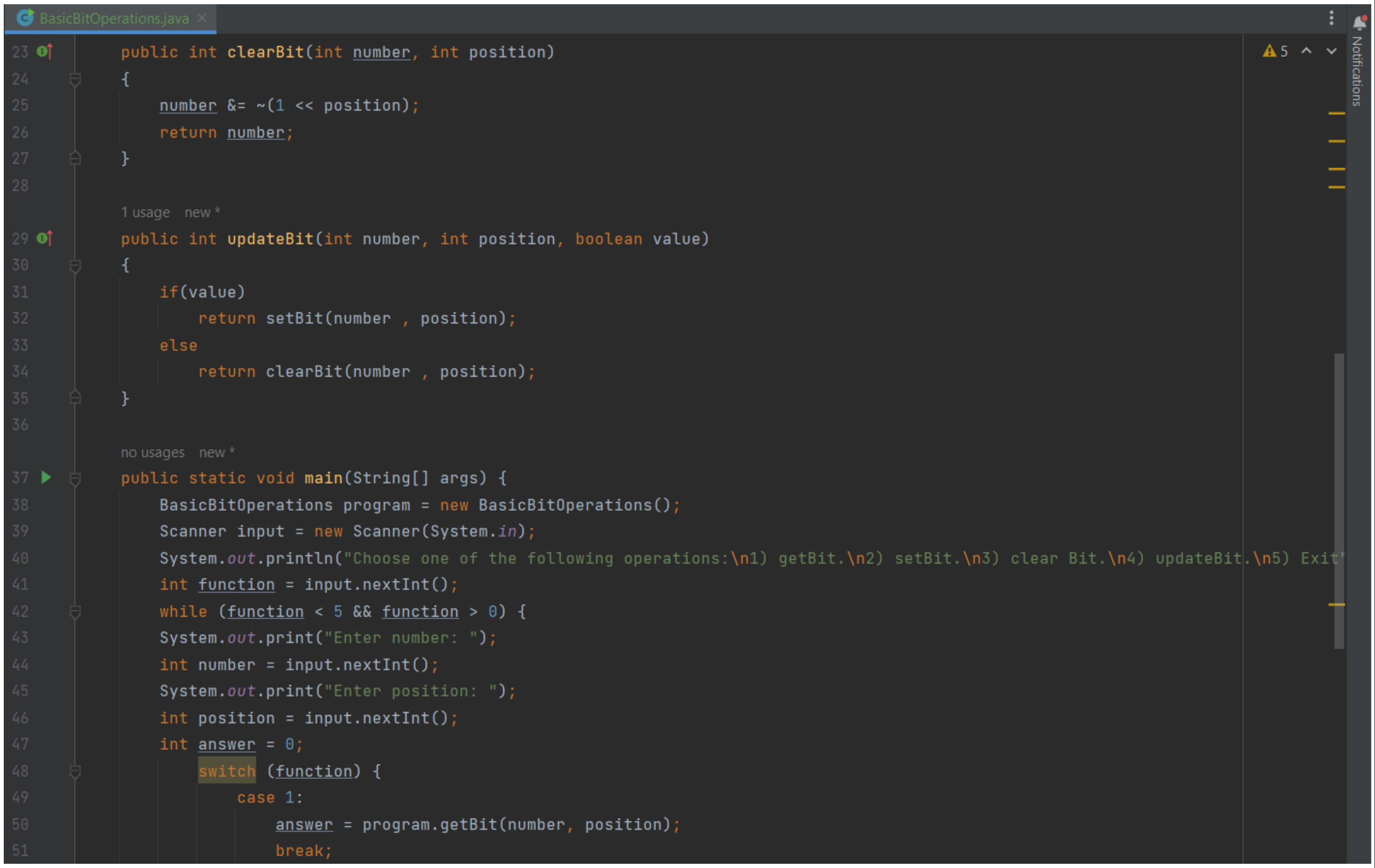
In the main function:

1. The menu is printed which asks the user to choose one of the following operations: 1) getBit. 2) setBit. 3) clear Bit. 4) updateBit. 5) Exit.
2. The user enters a number corresponding to the function he wants to perform and that number is stored in a variable named “function”.
3. As long as the entered function is bigger than zero and smaller than 5, the user is asked to enter the number and the position.
4. A variable “answer” is set to 0, in which all the performed functions will be stored into.
5. If the user enters “1”, the getBit() function is called and stored in the variable answer. Then, the switch case breaks.
6. If the user enters “2”, the setBit() function is called and stored in the variable answer. Then, the switch case breaks.
7. If the user enters “3”, the clearBit() function is called and stored in the variable answer. Then, the switch case breaks.
8. If the user enters “4”, the updateBit() function is called and stored in the variable answer. Then, the switch case breaks.
9. Finally, the content of the variable “answer” is printed and the menu is reprinted, the user is asked to enter the function number he wishes to perform and the program continues on till the user terminates it.

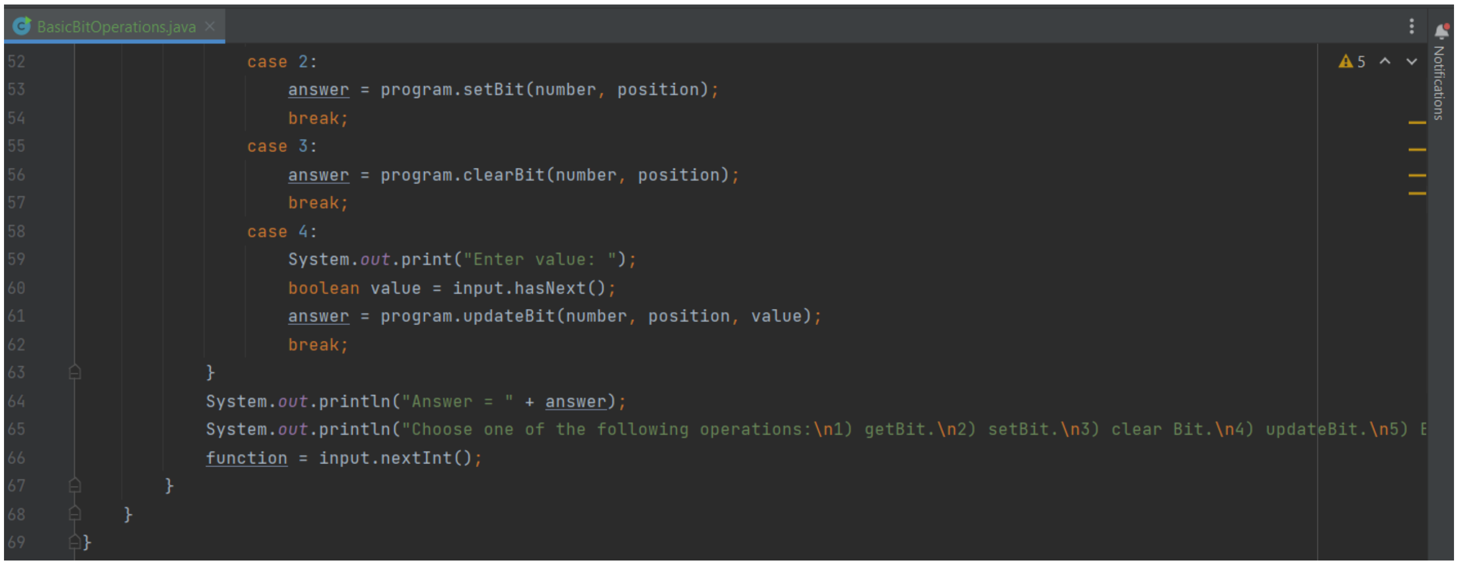
4. Code Snippets:


BasicBitOperations.java

```
import java.util.Scanner;

1 usage   1 implementation   new *
interface Part1 {
    1 usage   1 implementation   new *
    int getBit(int number, int position);

    2 usages   1 implementation   new *
    int setBit(int number, int position);

    2 usages   1 implementation   new *
    int clearBit(int number, int position);
    1 usage   1 implementation   new *
    int updateBit(int number, int position, boolean value);
}

2 usages   new *
public class BasicBitOperations implements Part1{

    1 usage   new *
    public int getBit(int number, int position){
        return (number >> position)&1;
    }
    2 usages   new *
    public int setBit(int number, int position)
    {
        number |= (1 << position);
        return number;
    }

```



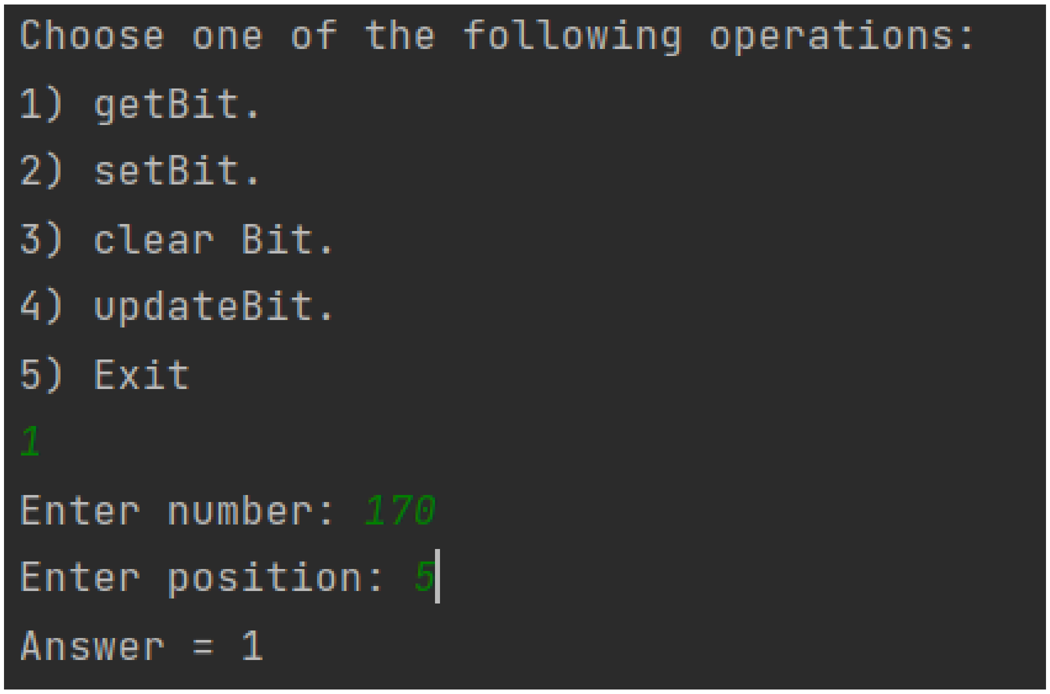
```
    public int clearBit(int number, int position)
    {
        number &= ~(1 << position);
        return number;
    }

    1 usage   new *
    public int updateBit(int number, int position, boolean value)
    {
        if(value)
            return setBit(number , position);
        else
            return clearBit(number , position);
    }

    no usages   new *
    public static void main(String[] args) {
        BasicBitOperations program = new BasicBitOperations();
        Scanner input = new Scanner(System.in);
        System.out.println("Choose one of the following operations:\n1) getBit.\n2) setBit.\n3) clear Bit.\n4) updateBit.\n5) Exit"
        int function = input.nextInt();
        while (function < 5 && function > 0) {
        System.out.print("Enter number: ");
        int number = input.nextInt();
        System.out.print("Enter position: ");
        int position = input.nextInt();
        int answer = 0;
            switch (function) {
                case 1:
                    answer = program.getBit(number, position);
                    break;
```



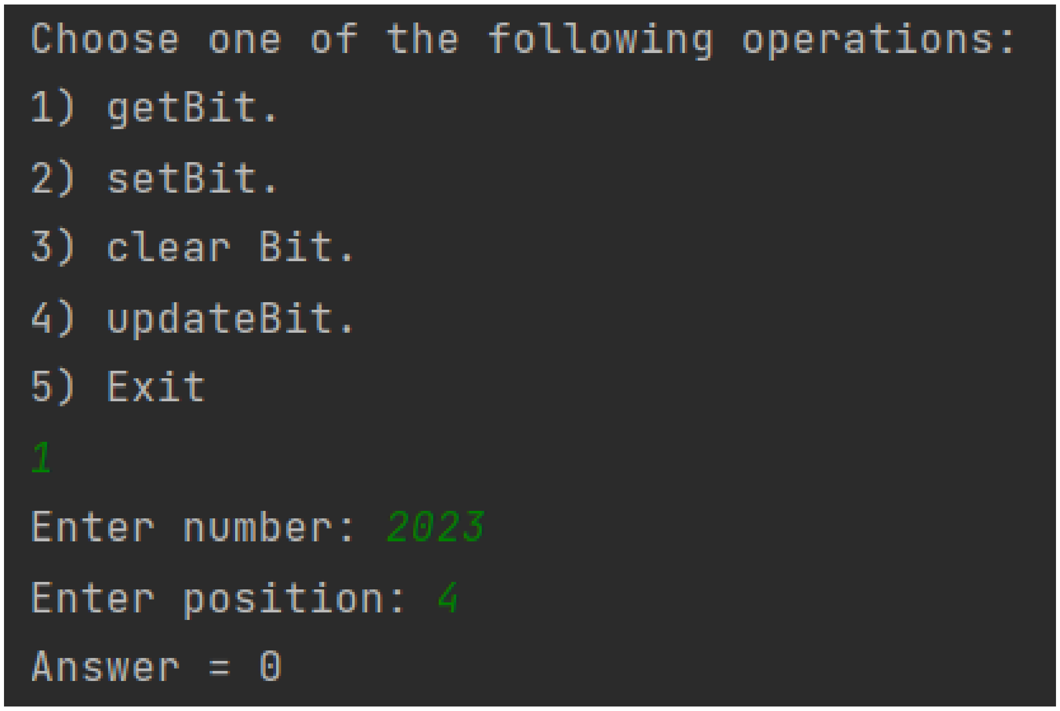
```
                case 2:
                    answer = program.setBit(number, position);
                    break;
                case 3:
                    answer = program.clearBit(number, position);
                    break;
                case 4:
                    System.out.print("Enter value: ");
                    boolean value = input.hasNext();
                    answer = program.updateBit(number, position, value);
                    break;
            }
            System.out.println("Answer = " + answer);
            System.out.println("Choose one of the following operations:\n1) getBit.\n2) setBit.\n3) clear Bit.\n4) updateBit.\n5) E
            function = input.nextInt();
        }
    }
}
```

5. Sample runs and different test cases:

⇒ **The first operation:**

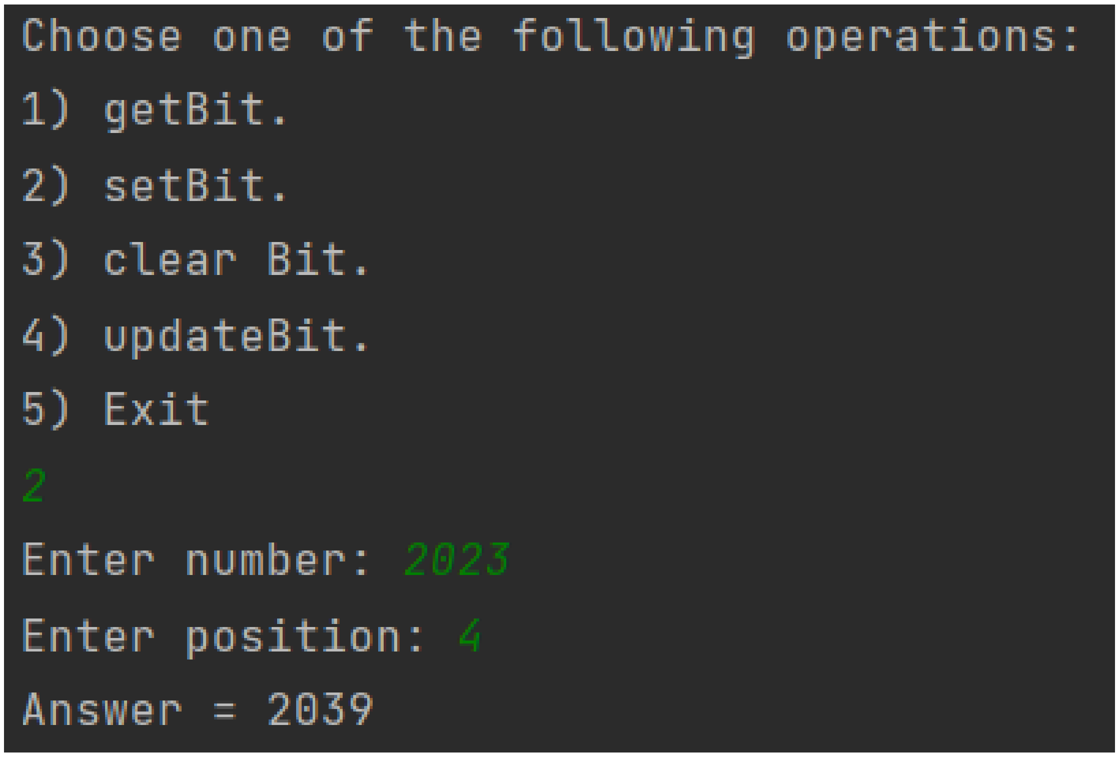


```
Choose one of the following operations:
1) getBit.
2) setBit.
3) clear Bit.
4) updateBit.
5) Exit
1
Enter number: 170
Enter position: 5
Answer = 1
```

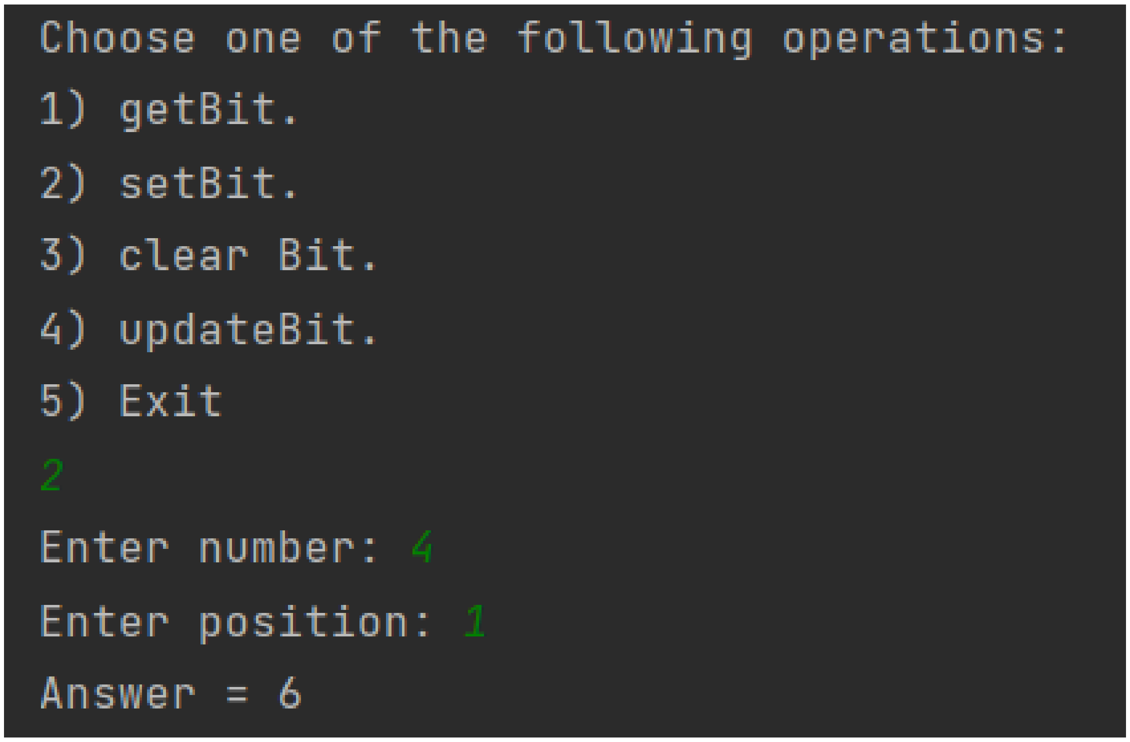


```
Choose one of the following operations:
1) getBit.
2) setBit.
3) clear Bit.
4) updateBit.
5) Exit
1
Enter number: 2023
Enter position: 4
Answer = 0
```

⇒ **The second operation:**

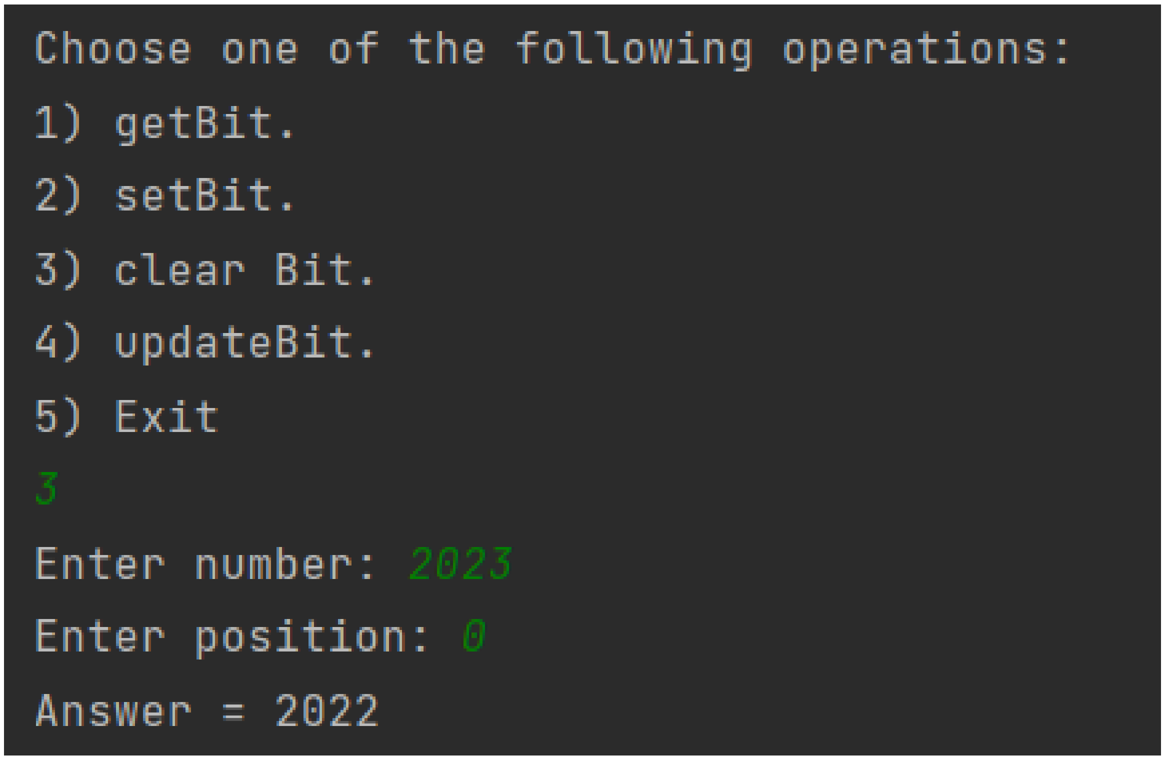


```
Choose one of the following operations:
1) getBit.
2) setBit.
3) clear Bit.
4) updateBit.
5) Exit
2
Enter number: 2023
Enter position: 4
Answer = 2039
```

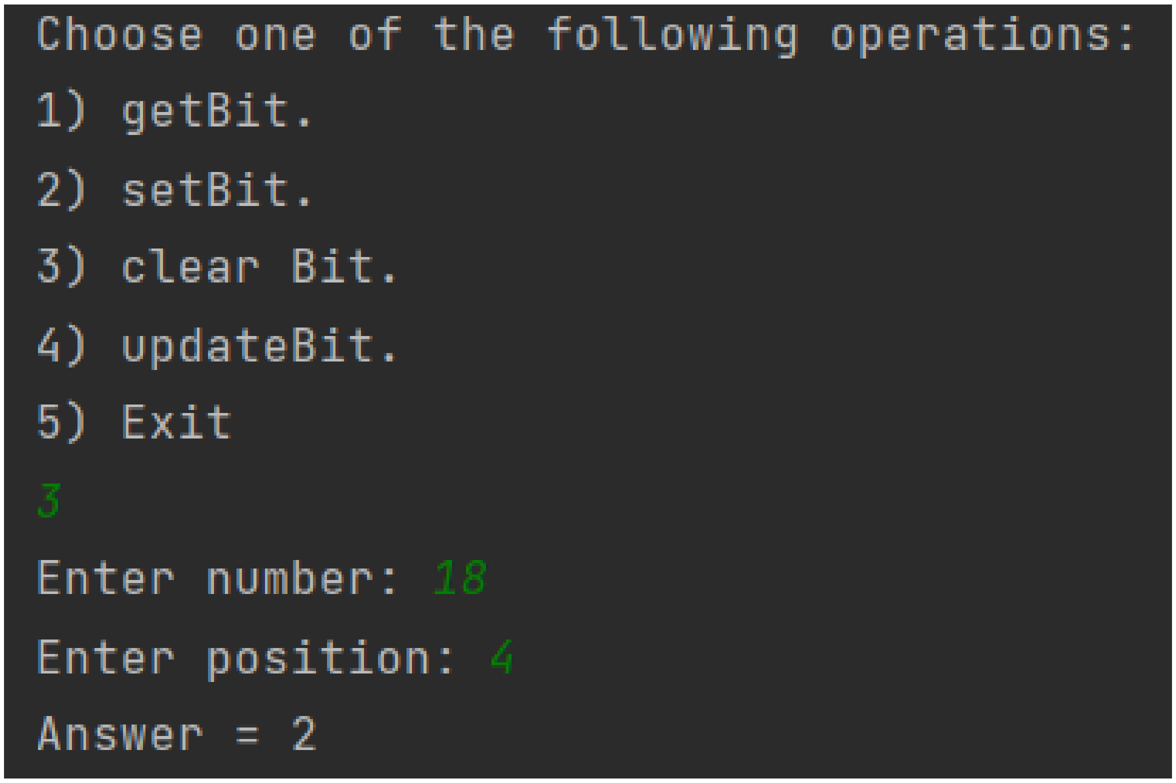


```
Choose one of the following operations:
1) getBit.
2) setBit.
3) clear Bit.
4) updateBit.
5) Exit
2
Enter number: 4
Enter position: 1
Answer = 6
```

⇒ **The third operation:**

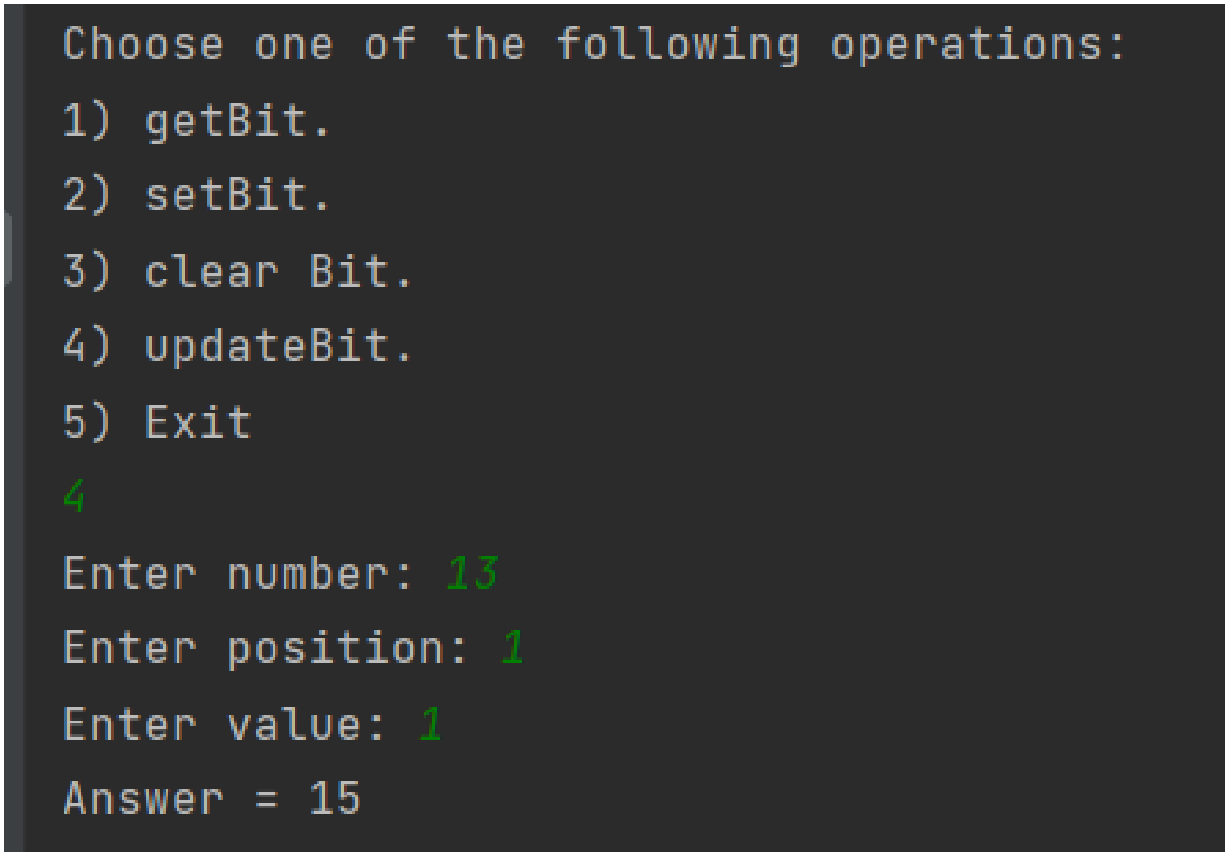


```
Choose one of the following operations:
1) getBit.
2) setBit.
3) clear Bit.
4) updateBit.
5) Exit
3
Enter number: 2023
Enter position: 0
Answer = 2022
```

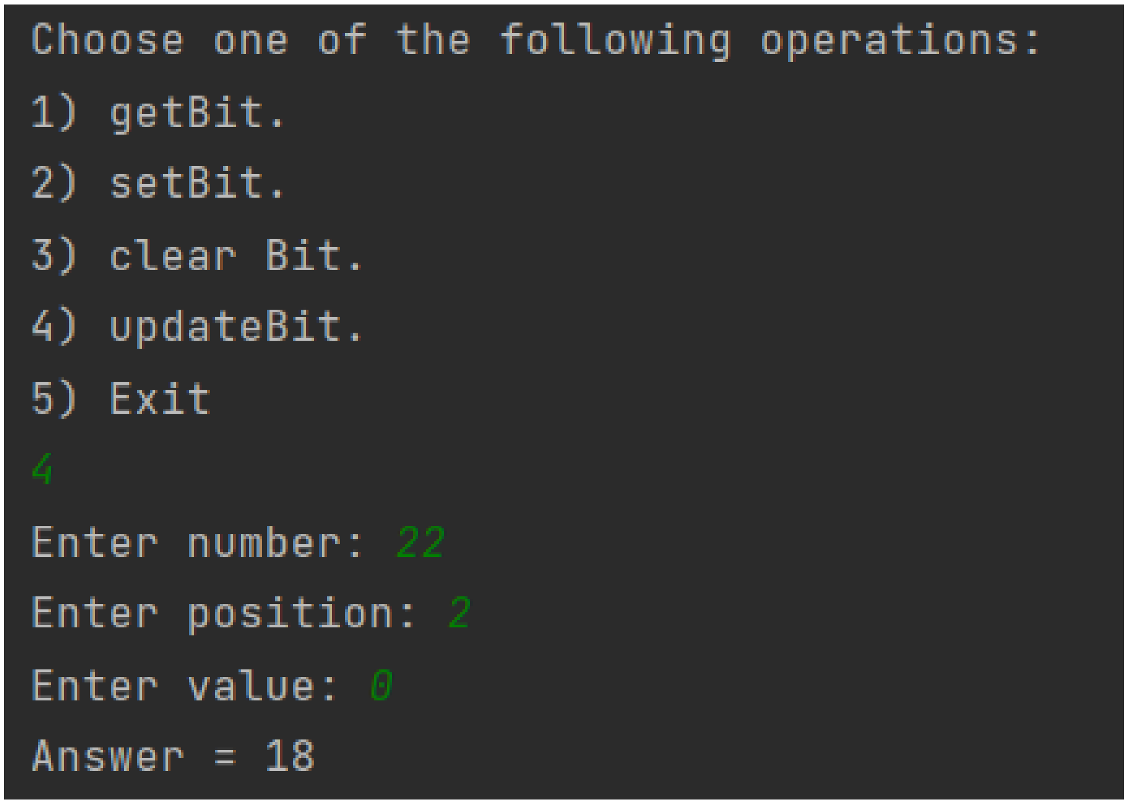


```
Choose one of the following operations:
1) getBit.
2) setBit.
3) clear Bit.
4) updateBit.
5) Exit
3
Enter number: 18
Enter position: 4
Answer = 2
```

⇒ **The fourth operation:**



```
Choose one of the following operations:
1) getBit.
2) setBit.
3) clear Bit.
4) updateBit.
5) Exit
4
Enter number: 13
Enter position: 1
Enter value: 1
Answer = 15
```



```
Choose one of the following operations:
1) getBit.
2) setBit.
3) clear Bit.
4) updateBit.
5) Exit
4
Enter number: 22
Enter position: 2
Enter value: 0
Answer = 18
```

6. Assumptions and details you find necessary to be clarified:

In the function "updateBit", assuming that the entered value is either 1 or 0, nothing else.

Part 2: Sets Operations using Bits Manipulation:

1. Problem statement:

1. Implement a Set data structure that takes in the constructor a list of strings as a Universe (U). The elements in the Set are subset of U. You must use bits to represent the set. The Set data structure should include the main operations:

1) Add string to the set

2) Union with another set

3) Intersection with another set

4) Complement of the set

5) Difference from another set

6) Cardinality of the set

7) Get elements of the set

2. Write a program that:

(a) Asks the user to enter a list of strings as a Universe (U)

(b) Then asks for a number of sets (that are subsets of U). The user will enter the elements in each set

(c) Then asks the user about the operations they want to perform:

1) Union of two sets

2) Intersection of two sets

3) Complement of a set

4) Difference between two sets

5) Cardinality of a set

6) Print a set

2. Used data structures:

1) Lists:

Name: universe
Data type: String
Functionality: All the elements of the universe are stored into.

2) Arrays:

a) Name: arr
Data type: int
Functionality: To store elements of the universe after removing the braces and the commas.

b) Name: subsetsIndex
Data type: int
Functionality: where a number is stored in each cell and this number is an indicator to the elements present in the subsets.

c) Name: newArray
Data type: int
Functionality: in case of adding new elements, they are stored here then copied to the "universe" list.

3. Algorithms documented using flow charts or pseudo code:

- **Documenting algorithms through pseudocode:**

First: 2 functions are made.

(1) **getBit() :** It takes the number and its position as an input, performs a "RIGHT SHIFT" operation on the number by value of its position. Then, performs a "BITWISE AND" to the number and "1". Finally, it returns that a bit. The purpose is to get the value of a certain bit in a number given the number and required position. The return value is either 1 or 0.

(2) **setBit():** It takes the number and its position as an input, performs a "LEFT SHIFT" operation on the number by value of its position. Then, performs a "BITWISE OR" to the number and itself. Finally, it returns that operated on number. The purpose is to set the value of a certain bit in a number given the number and required position to "1". The return value is the new number after setting the bit to 1.

In the main function:

1. The user is asked to enter the elements of the universe set as a string in the form of "{ , , ,}".
2. The curly braces are then removed and the string is split at each comma. The elements are stored in an array of integer type.
3. A list named "universe" of type string is created where all elements of the universe stored in the array are copied to the list "universe"
4. The user is asked to enter the number of subsets and an array named "subsetIndex" whose size is the number of the subsets, is created. Its cells' values is all initialized to 0.
5. The user is asked to enter the size of the subset and its elements. An extra step is performed to check that all elements entered in the subsets already exist in the universal set through an if

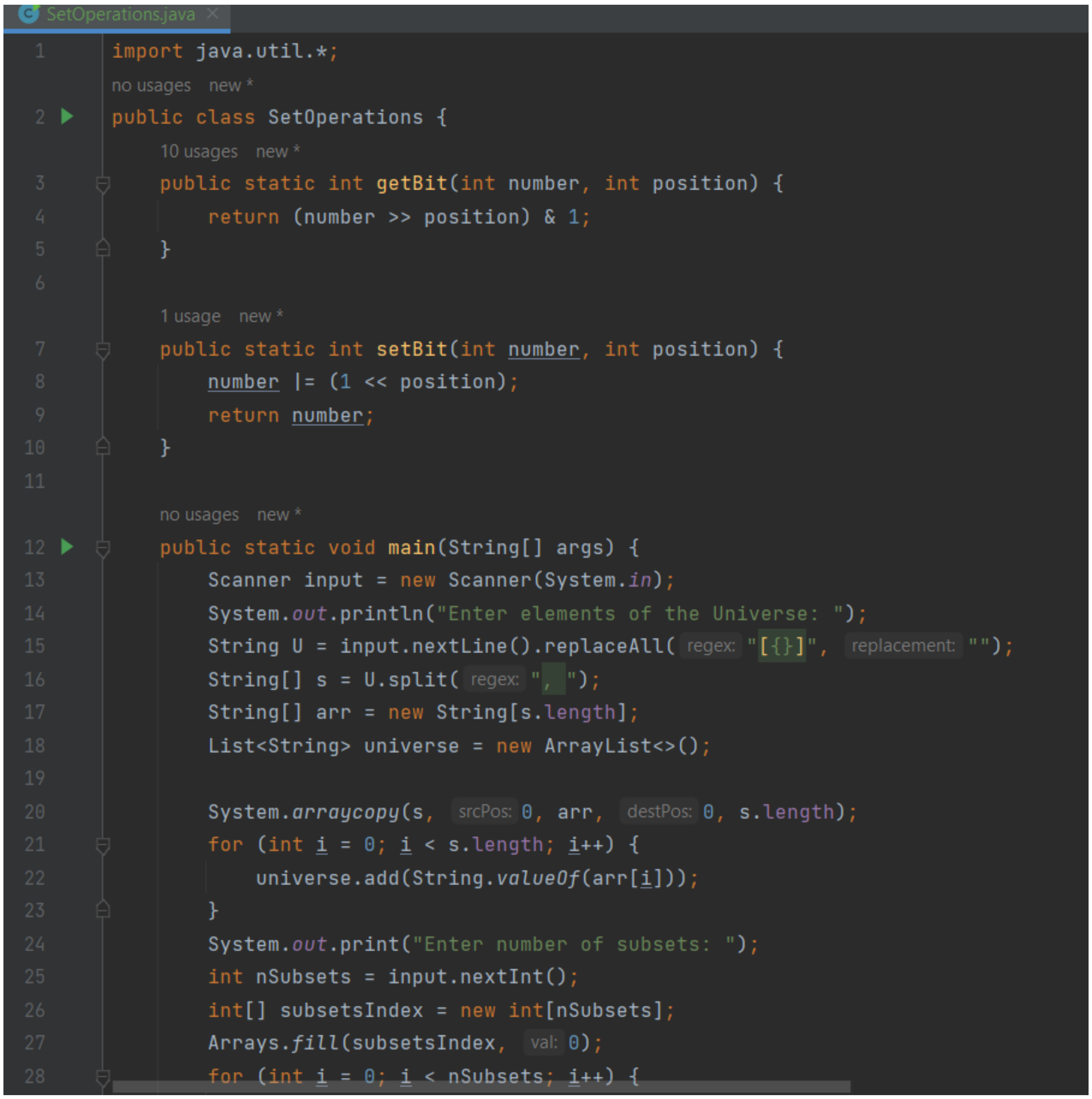
conditional and comparing the entered element to the elements of the universal. If element is not found, a message is printed asking the user to enter a valid one.

6. A variable named “number” of type int, is declared and initialized to 0.
7. If element is valid, the function setBit() is called with given parameters: the variable “number”, and the index of the element in the universe. This step is performed to each element in each subset. The resulting number is stored in the array “subsetIndex” in an index which is equal to the corresponding set order.
8. The menu is printed which asks the user to choose one of the following operations: 1) Union of two sets. 2) Intersection of two sets. 3) Complement of a set. 4) Difference between two sets. 5) Cardinality of a set. 6) Print a set. 7) Add string to the set. 8) Exit.
9. The user enters a number corresponding to the function he wants to perform and that number is stored in a variable named “operation”.
10. Two variables are initialized to represent the order of subsets. “s1” & “s2”.
11. As long as the entered operation is bigger than zero and smaller than 8, a switch case works.
12. In case the user enters “1”, he is asked to enter the order of two sets. If any of them was greater than the number of subsets an “operation invalid” message is printed. If any of “s1” or “s2” was equal to 0, the U is printed. Otherwise, an OR-ing to the bits of the variable “number” which is stored in the array of “subsetIndex” is perforemed to get any elements present in any set of both. Finally, these elements are printed.
13. In case the user enters “2”, he is asked to enter the order of two sets. If any of them was greater than the number of subsets an “operation invalid” message is printed. If any of “s1” or “s2” was equal to 0, the elements of the other set (whose bits are 1 in the variable “number”) are printed. Otherwise, an AND-ing to the bits of the variable “number” which is stored in the array of “subsetIndex” is perforemed to get the elements present in both sets together. Finally, these elements are printed. The NO INTERSECTION case is handled.
14. In case the user enters “3”, he is asked to enter the order of the set. If it was greater than the number of subsets an “operation invalid” message is printed. If “s1” was equal to 0, “NULL” message is printed. Otherwise, a NEGATION to the bits of the variable “number” which is stored in the array of “subsetIndex” is perforemed to get the elements not present in the subset. Finally, these elements are printed.
15. In case the user enters “4”, he is asked to enter the order of two sets. If any of them was greater than the number of subsets an “operation invalid” message is printed. If “s2” was equal to 0, “NULL” message is printed. If “s1” was equal to 0, the Negation of the bits present in the pother subset “s2” is printed. Otherwise, a NEGATION to the bits of the variable “number” of “s2” which are stored in the array of “subsetIndex” is performed with AND-ing with the bits of “s1” to get the elements present in s1 and not in s2. Finally, these elements are printed.
16. In case the user enters “5”, he is asked to enter the order of the set. If it was greater than the number of subsets an “operation invalid” message is printed. If “s1” was equal to 0, the length of the “universe” set is printed. Otherwise, it’s checked if the bits of the entered set are equal to one, a count is incremented. Finally, the result of that count is printed indicating the “Cardinality” of the set.
17. In case the user enters “5”, he is asked to enter the order of the set. If it was greater than the number of subsets an “operation invalid” message is printed. If “s1” was equal to 0, the elements of the universe are printed. Otherwise, it’s checked if the bits of the entered set are

equal to one, that corresponding element is printed. Finally, the elements of the required set are printed.

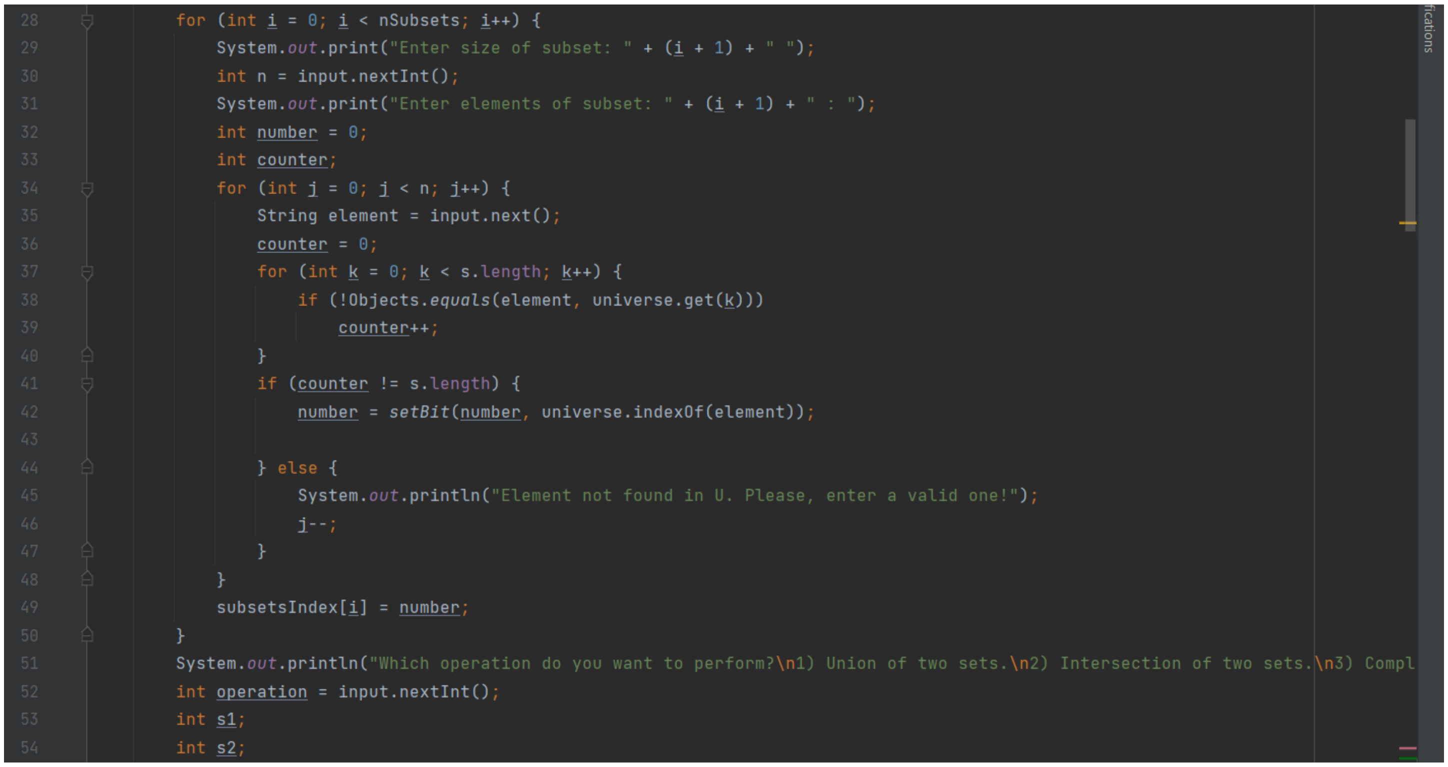
18. In case the user enters “7”, he is asked to enter the number of elements he wants to add and those elements are scanned. They are added to the original U and it’s printed with the newly added elements.
19. Every time an operation is performed, the menu is reprinted, the user is asked to enter the operation number he wishes to perform and the program continues on till the user terminates it.

(3) Code Snippets:

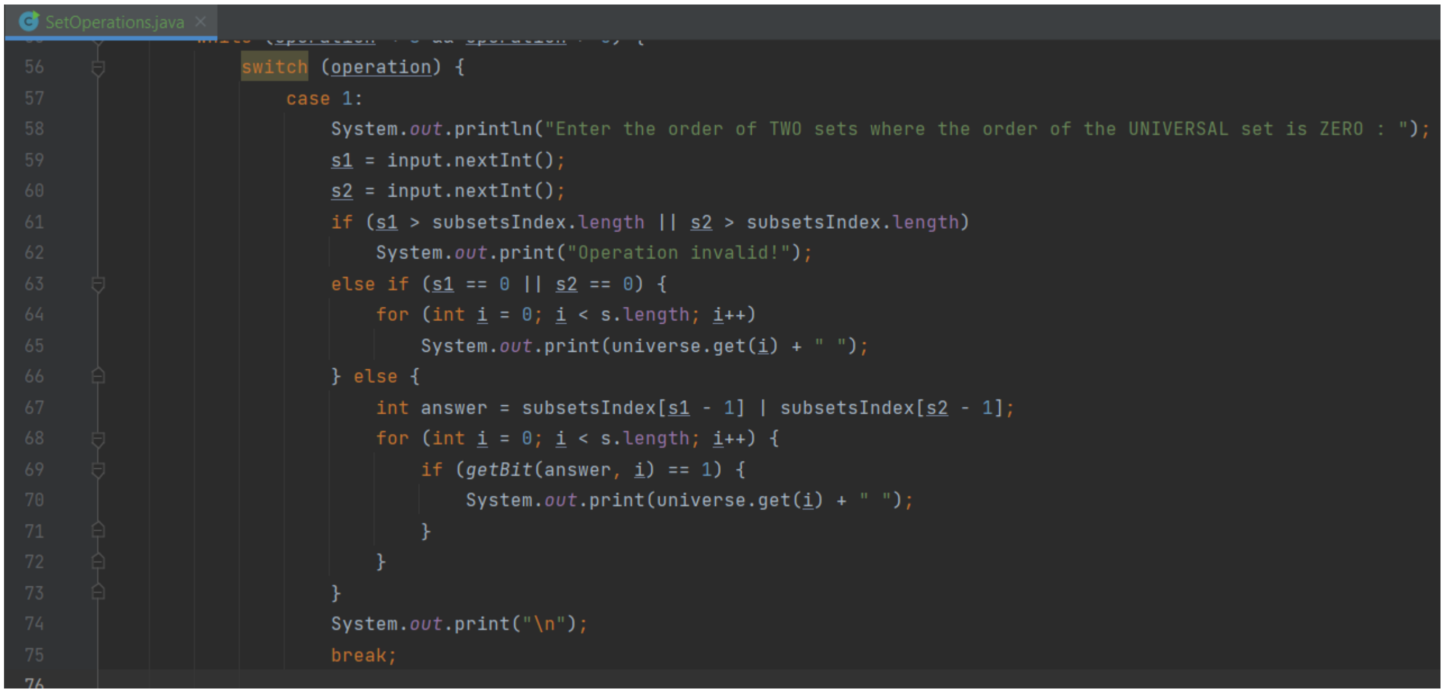


SetOperations.java

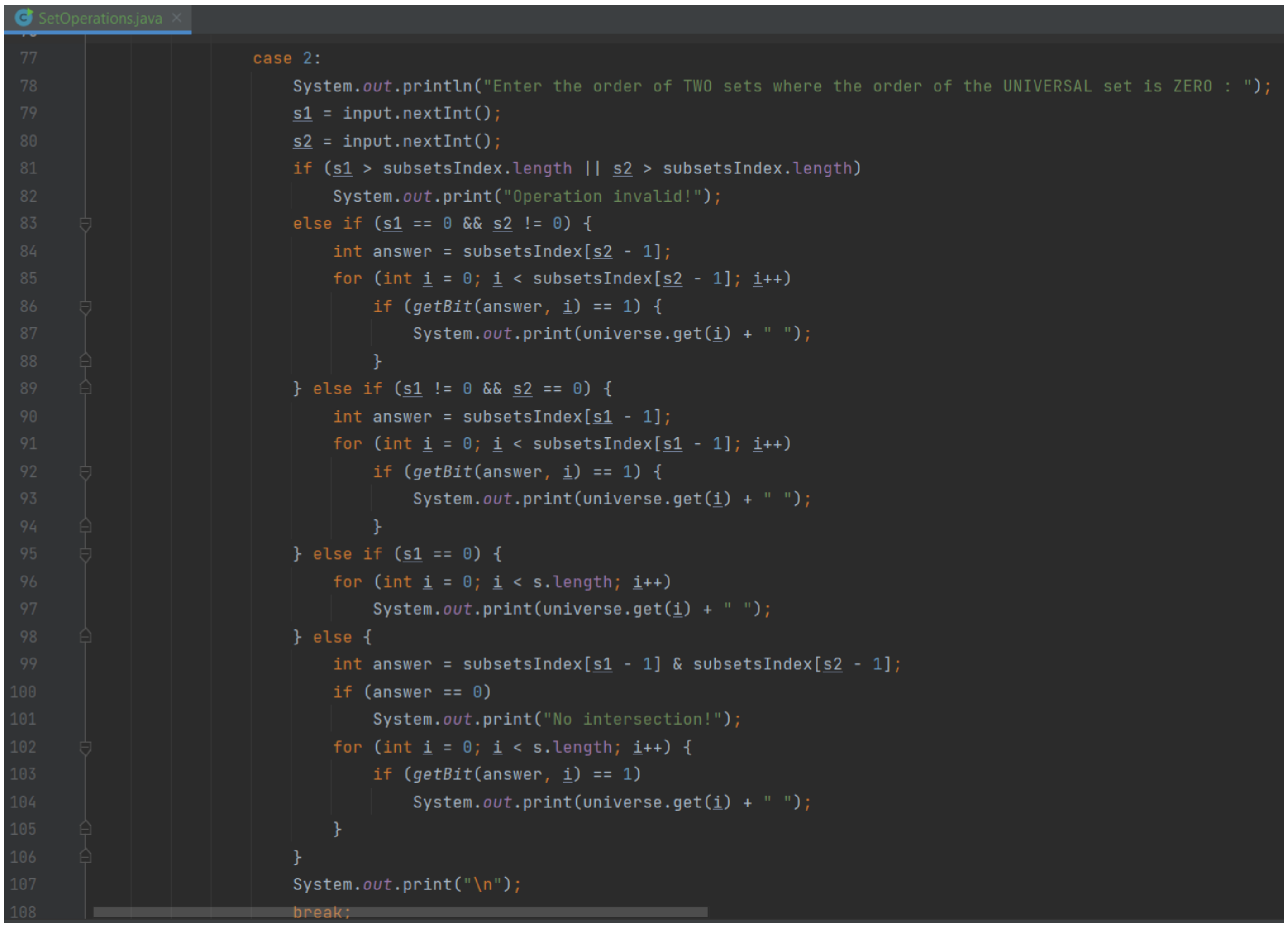
```java
import java.util.*;
no usages   new *
public class SetOperations {
    10 usages   new *
    public static int getBit(int number, int position) {
        return (number >> position) & 1;
    }

    1 usage   new *
    public static int setBit(int number, int position) {
        number |= (1 << position);
        return number;
    }

    no usages   new *
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("Enter elements of the Universe: ");
        String U = input.nextLine().replaceAll( regex: "[{}]",  replacement: "");
        String[] s = U.split( regex: ", ");
        String[] arr = new String[s.length];
        List<String> universe = new ArrayList<>();

        System.arraycopy(s,  srcPos: 0, arr,  destPos: 0, s.length);
        for (int i = 0; i < s.length; i++) {
            universe.add(String.valueOf(arr[i]));
        }
        System.out.print("Enter number of subsets: ");
        int nSubsets = input.nextInt();
        int[] subsetsIndex = new int[nSubsets];
        Arrays.fill(subsetsIndex,  val: 0);
        for (int i = 0; i < nSubsets; i++) {
```



```java
        for (int i = 0; i < nSubsets; i++) {
            System.out.print("Enter size of subset: " + (i + 1) + " ");
            int n = input.nextInt();
            System.out.print("Enter elements of subset: " + (i + 1) + " : ");
            int number = 0;
            int counter;
            for (int j = 0; j < n; j++) {
                String element = input.next();
                counter = 0;
                for (int k = 0; k < s.length; k++) {
                    if (!Objects.equals(element, universe.get(k)))
                        counter++;
                }
                if (counter != s.length) {
                    number = setBit(number, universe.indexOf(element));

                } else {
                    System.out.println("Element not found in U. Please, enter a valid one!");
                    j--;
                }
            }
            subsetsIndex[i] = number;
        }
        System.out.println("Which operation do you want to perform?\n1) Union of two sets.\n2) Intersection of two sets.\n3) Compl
        int operation = input.nextInt();
        int s1;
        int s2;
```

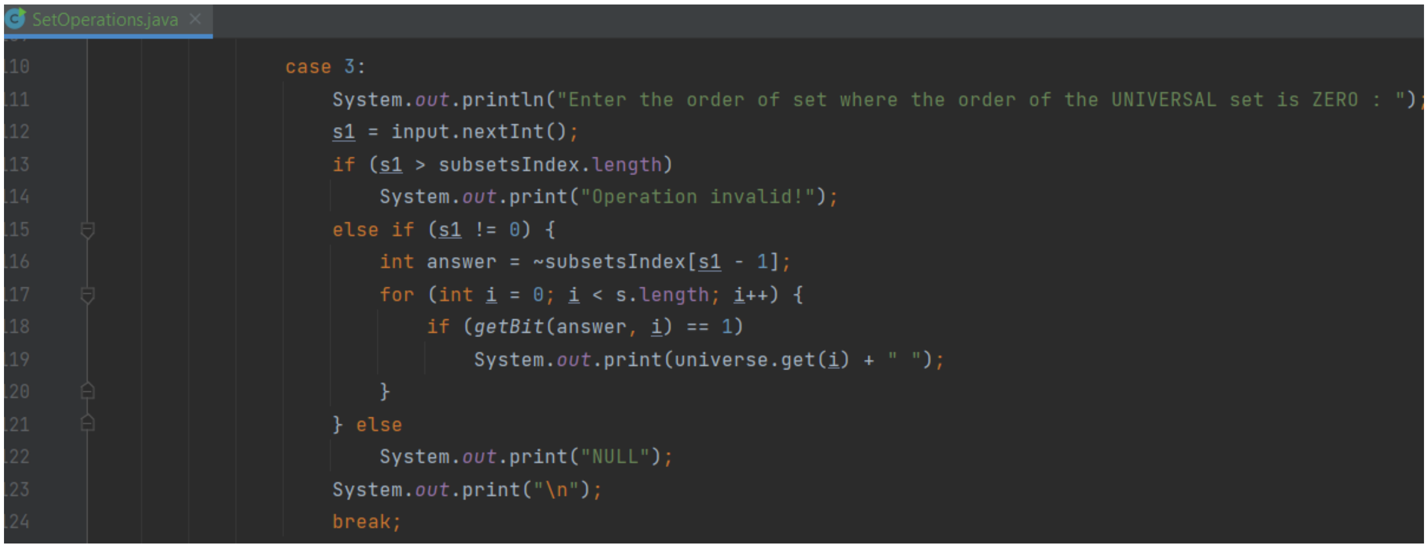


```java
            switch (operation) {
                case 1:
                    System.out.println("Enter the order of TWO sets where the order of the UNIVERSAL set is ZERO : ");
                    s1 = input.nextInt();
                    s2 = input.nextInt();
                    if (s1 > subsetsIndex.length || s2 > subsetsIndex.length)
                        System.out.print("Operation invalid!");
                    else if (s1 == 0 || s2 == 0) {
                        for (int i = 0; i < s.length; i++)
                            System.out.print(universe.get(i) + " ");
                    } else {
                        int answer = subsetsIndex[s1 - 1] | subsetsIndex[s2 - 1];
                        for (int i = 0; i < s.length; i++) {
                            if (getBit(answer, i) == 1) {
                                System.out.print(universe.get(i) + " ");
                            }
                        }
                    }
                    System.out.print("\n");
                    break;
```



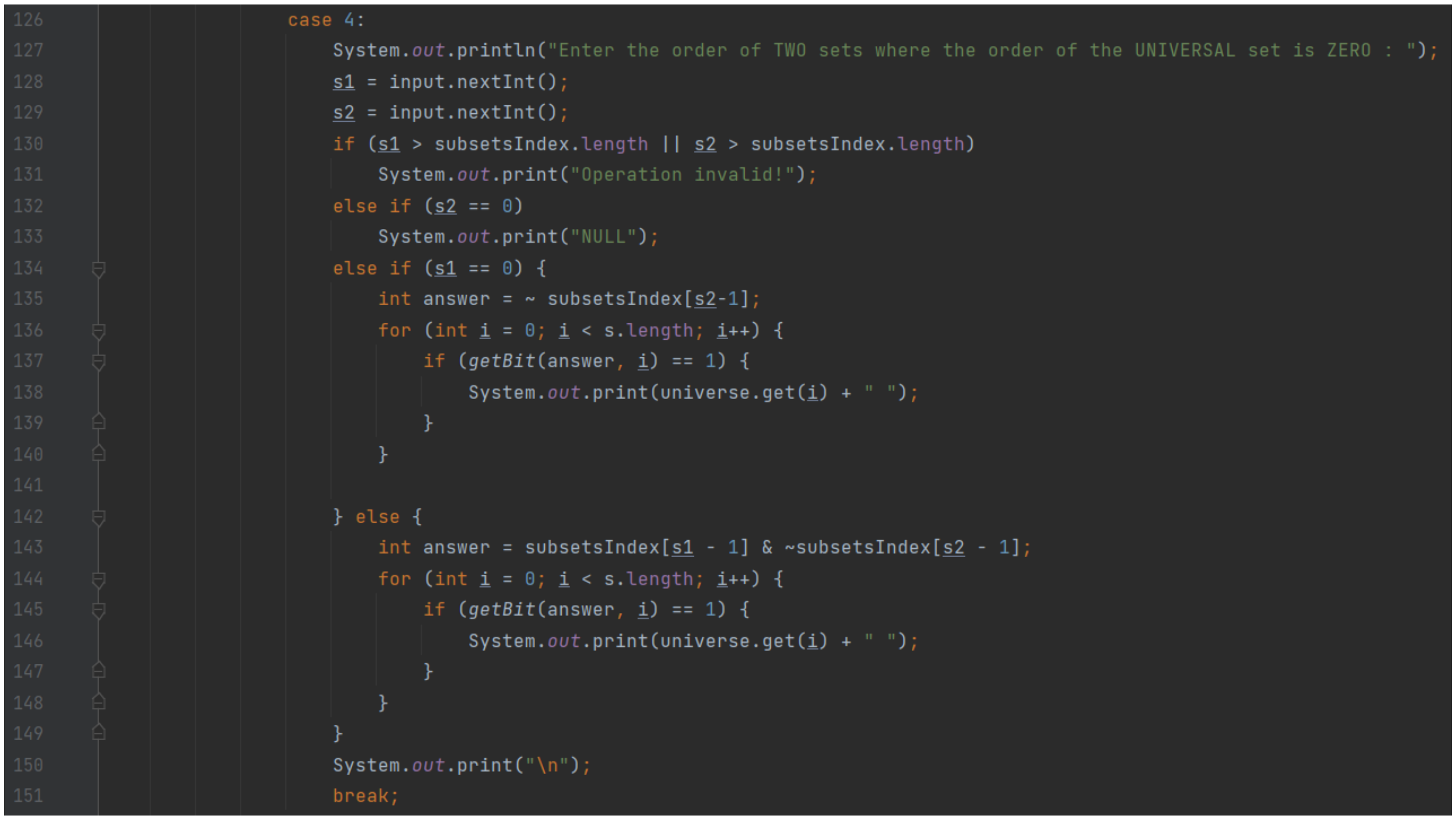
SetOperations.java

```java
                case 2:
                    System.out.println("Enter the order of TWO sets where the order of the UNIVERSAL set is ZERO : ");
                    s1 = input.nextInt();
                    s2 = input.nextInt();
                    if (s1 > subsetsIndex.length || s2 > subsetsIndex.length)
                        System.out.print("Operation invalid!");
                    else if (s1 == 0 && s2 != 0) {
                        int answer = subsetsIndex[s2 - 1];
                        for (int i = 0; i < subsetsIndex[s2 - 1]; i++)
                            if (getBit(answer, i) == 1) {
                                System.out.print(universe.get(i) + " ");
                            }
                    } else if (s1 != 0 && s2 == 0) {
                        int answer = subsetsIndex[s1 - 1];
                        for (int i = 0; i < subsetsIndex[s1 - 1]; i++)
                            if (getBit(answer, i) == 1) {
                                System.out.print(universe.get(i) + " ");
                            }
                    } else if (s1 == 0) {
                        for (int i = 0; i < s.length; i++)
                            System.out.print(universe.get(i) + " ");
                    } else {
                        int answer = subsetsIndex[s1 - 1] & subsetsIndex[s2 - 1];
                        if (answer == 0)
                            System.out.print("No intersection!");
                        for (int i = 0; i < s.length; i++) {
                            if (getBit(answer, i) == 1)
                                System.out.print(universe.get(i) + " ");
                        }
                    }
                    System.out.print("\n");
                    break;
```

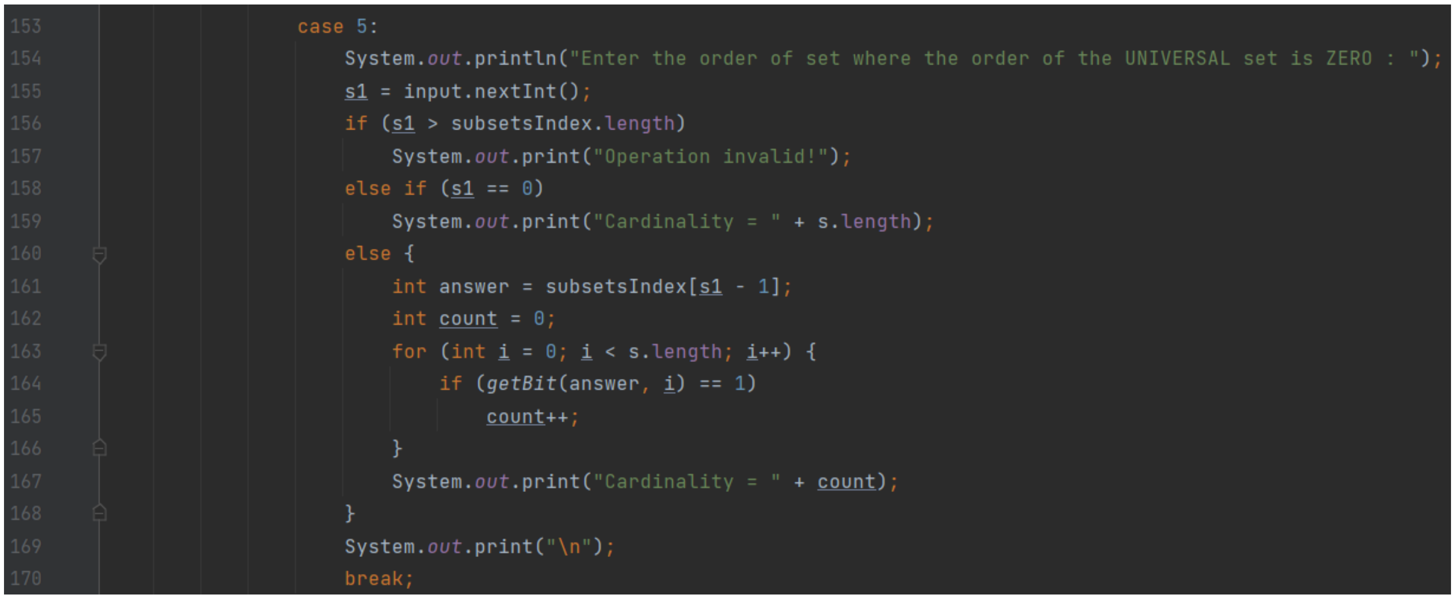


SetOperations.java

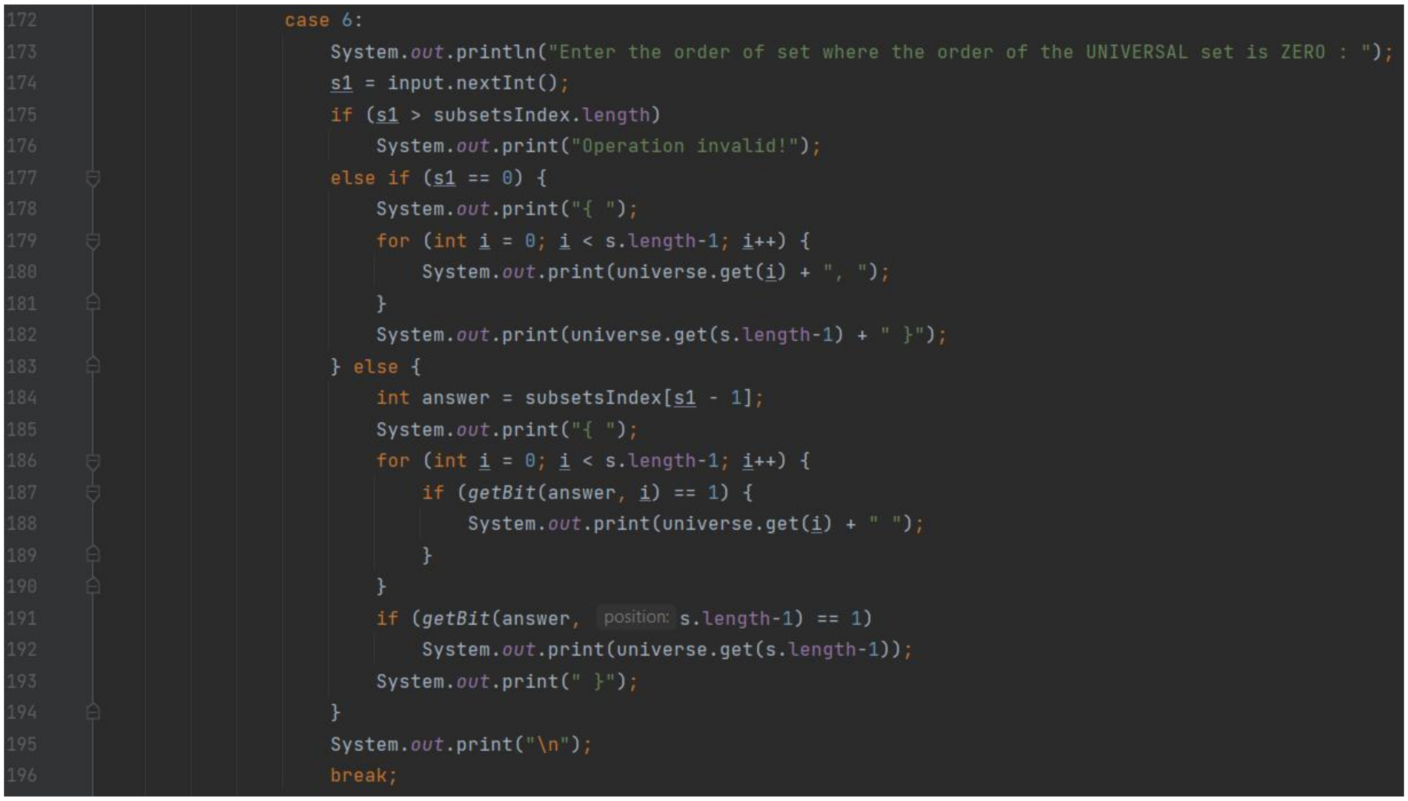
```java
                case 3:
                    System.out.println("Enter the order of set where the order of the UNIVERSAL set is ZERO : ")
                    s1 = input.nextInt();
                    if (s1 > subsetsIndex.length)
                        System.out.print("Operation invalid!");
                    else if (s1 != 0) {
                        int answer = ~subsetsIndex[s1 - 1];
                        for (int i = 0; i < s.length; i++) {
                            if (getBit(answer, i) == 1)
                                System.out.print(universe.get(i) + " ");
                        }
                    } else
                        System.out.print("NULL");
                    System.out.print("\n");
                    break;
```



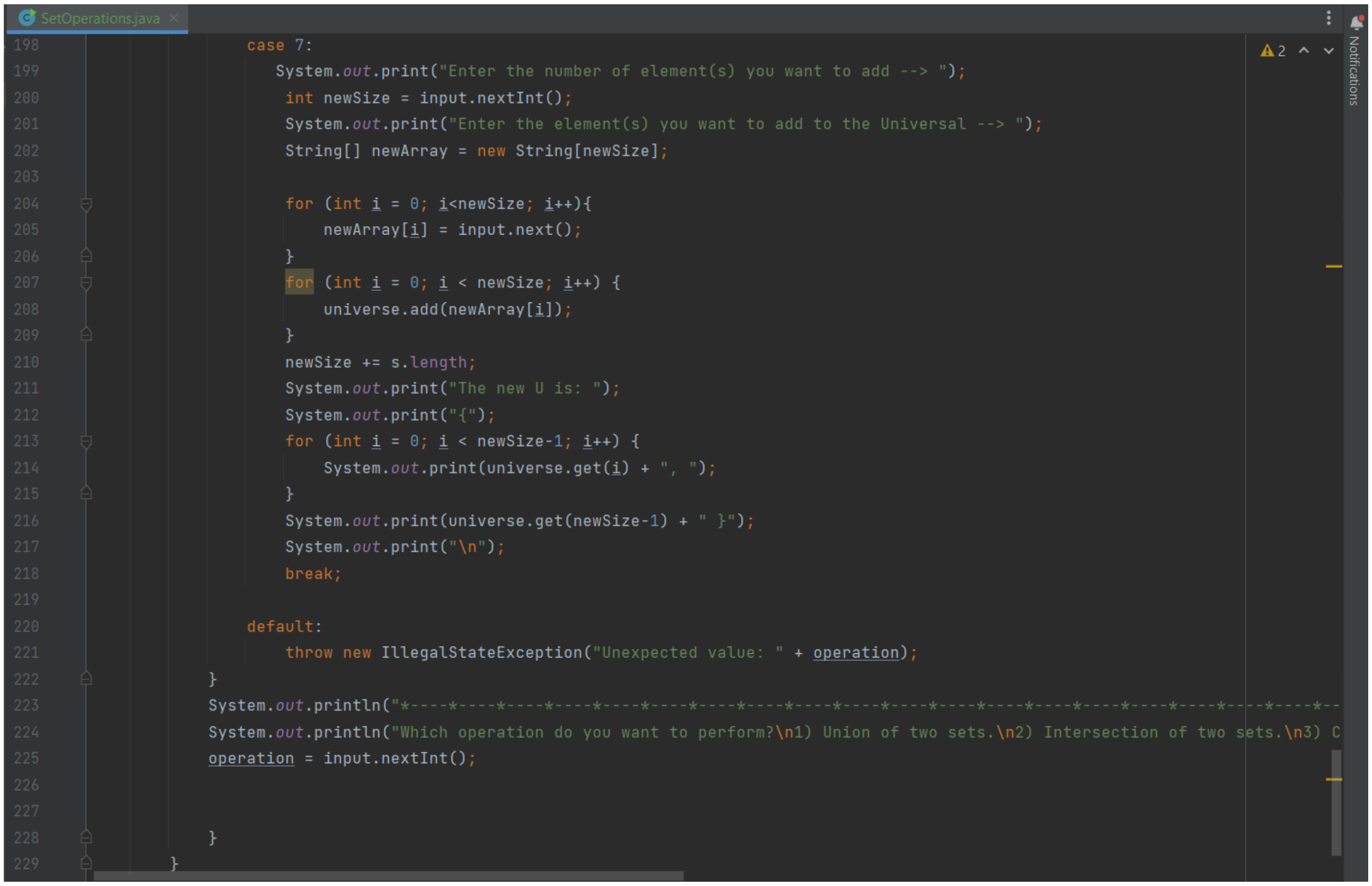
```java
                case 4:
                    System.out.println("Enter the order of TWO sets where the order of the UNIVERSAL set is ZERO : ");
                    s1 = input.nextInt();
                    s2 = input.nextInt();
                    if (s1 > subsetsIndex.length || s2 > subsetsIndex.length)
                        System.out.print("Operation invalid!");
                    else if (s2 == 0)
                        System.out.print("NULL");
                    else if (s1 == 0) {
                        int answer = ~ subsetsIndex[s2-1];
                        for (int i = 0; i < s.length; i++) {
                            if (getBit(answer, i) == 1) {
                                System.out.print(universe.get(i) + " ");
                            }
                        }

                    } else {
                        int answer = subsetsIndex[s1 - 1] & ~subsetsIndex[s2 - 1];
                        for (int i = 0; i < s.length; i++) {
                            if (getBit(answer, i) == 1) {
                                System.out.print(universe.get(i) + " ");
                            }
                        }
                    }
                    System.out.print("\n");
                    break;
```



```java
                case 5:
                    System.out.println("Enter the order of set where the order of the UNIVERSAL set is ZERO : ");
                    s1 = input.nextInt();
                    if (s1 > subsetsIndex.length)
                        System.out.print("Operation invalid!");
                    else if (s1 == 0)
                        System.out.print("Cardinality = " + s.length);
                    else {
                        int answer = subsetsIndex[s1 - 1];
                        int count = 0;
                        for (int i = 0; i < s.length; i++) {
                            if (getBit(answer, i) == 1)
                                count++;
                        }
                        System.out.print("Cardinality = " + count);
                    }
                    System.out.print("\n");
                    break;
```



```java
            case 6:
                System.out.println("Enter the order of set where the order of the UNIVERSAL set is ZERO : ");
                s1 = input.nextInt();
                if (s1 > subsetsIndex.length)
                    System.out.print("Operation invalid!");
                else if (s1 == 0) {
                    System.out.print("{ ");
                    for (int i = 0; i < s.length-1; i++) {
                        System.out.print(universe.get(i) + ", ");
                    }
                    System.out.print(universe.get(s.length-1) + " }");
                } else {
                    int answer = subsetsIndex[s1 - 1];
                    System.out.print("{ ");
                    for (int i = 0; i < s.length-1; i++) {
                        if (getBit(answer, i) == 1) {
                            System.out.print(universe.get(i) + " ");
                        }
                    }
                    if (getBit(answer,  position: s.length-1) == 1)
                        System.out.print(universe.get(s.length-1));
                    System.out.print(" }");
                }
                System.out.print("\n");
                break;
```

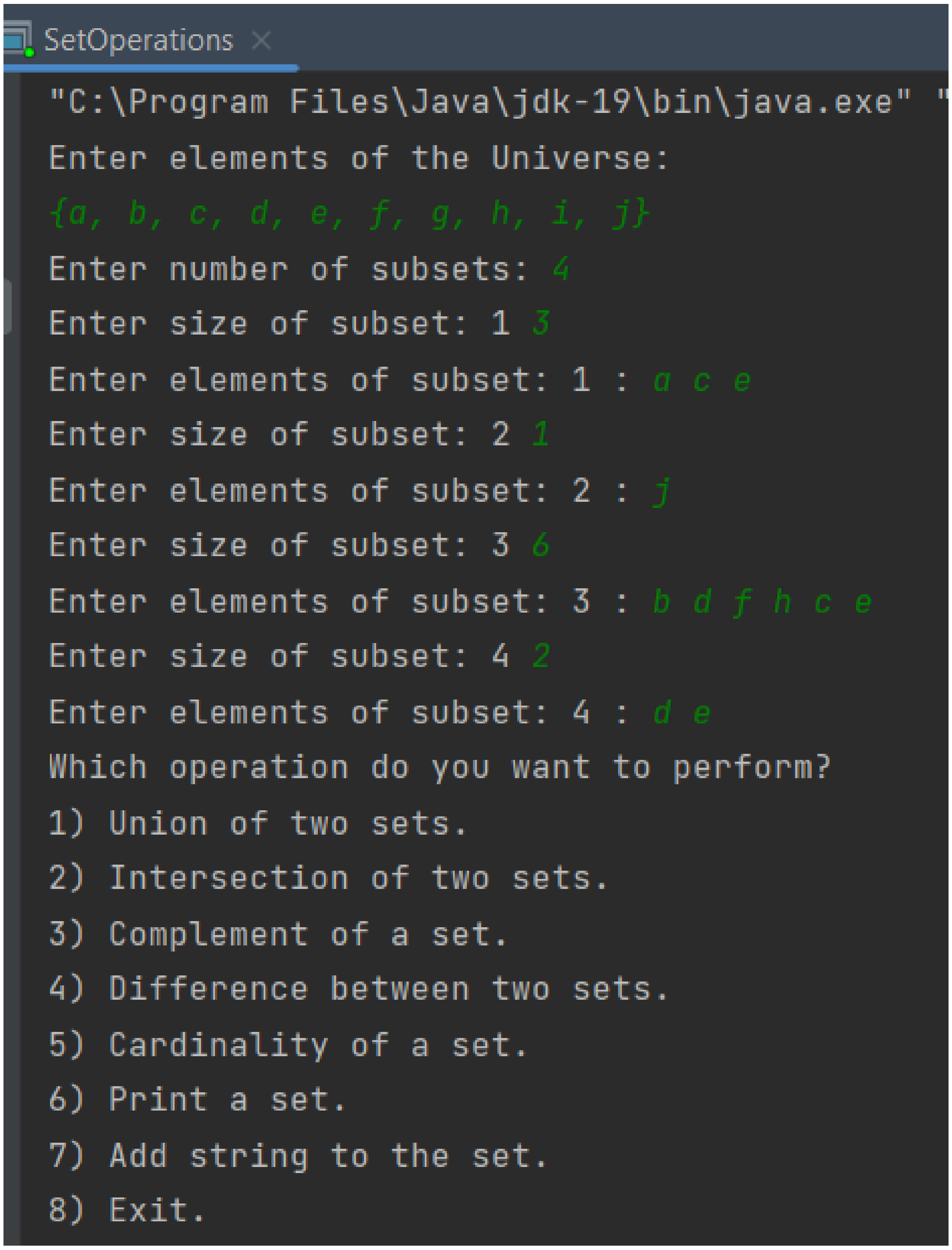


SetOperations.java

```java
            case 7:
               System.out.print("Enter the number of element(s) you want to add --> ");
                int newSize = input.nextInt();
                System.out.print("Enter the element(s) you want to add to the Universal --> ");
                String[] newArray = new String[newSize];

                for (int i = 0; i<newSize; i++){
                    newArray[i] = input.next();
                }
                for (int i = 0; i < newSize; i++) {
                    universe.add(newArray[i]);
                }
                newSize += s.length;
                System.out.print("The new U is: ");
                System.out.print("{");
                for (int i = 0; i < newSize-1; i++) {
                    System.out.print(universe.get(i) + ", ");
                }
                System.out.print(universe.get(newSize-1) + " }");
                System.out.print("\n");
                break;

            default:
                throw new IllegalStateException("Unexpected value: " + operation);
        }
        System.out.println("*----*----*----*----*----*----*----*----*----*----*----*----*----*----*----*----*----*----*----*--
        System.out.println("Which operation do you want to perform?\n1) Union of two sets.\n2) Intersection of two sets.\n3) C
        operation = input.nextInt();


        }
    }
```

(4) Sample runs and different test cases:

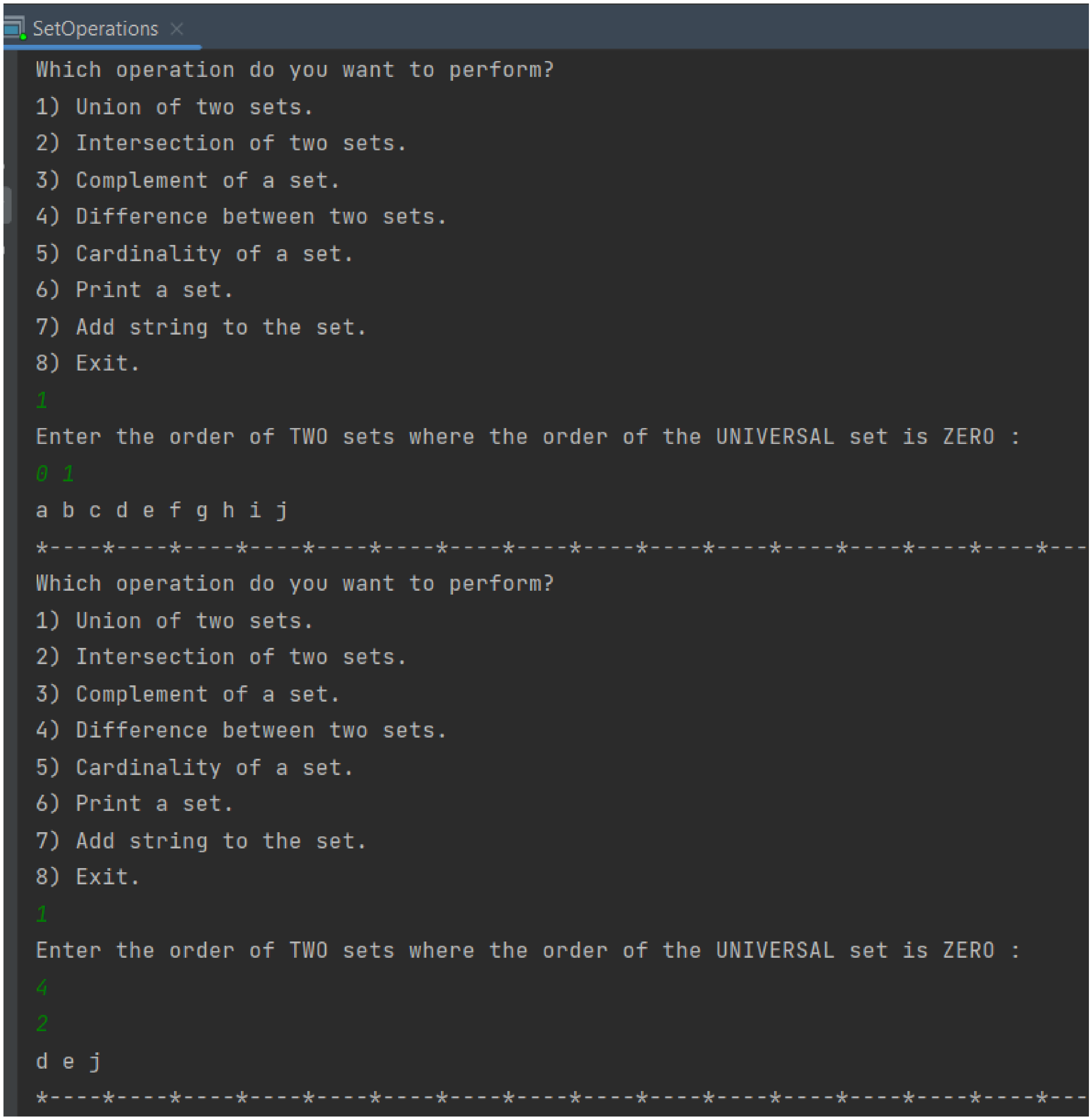
⇒ **Data entry:**



SetOperations

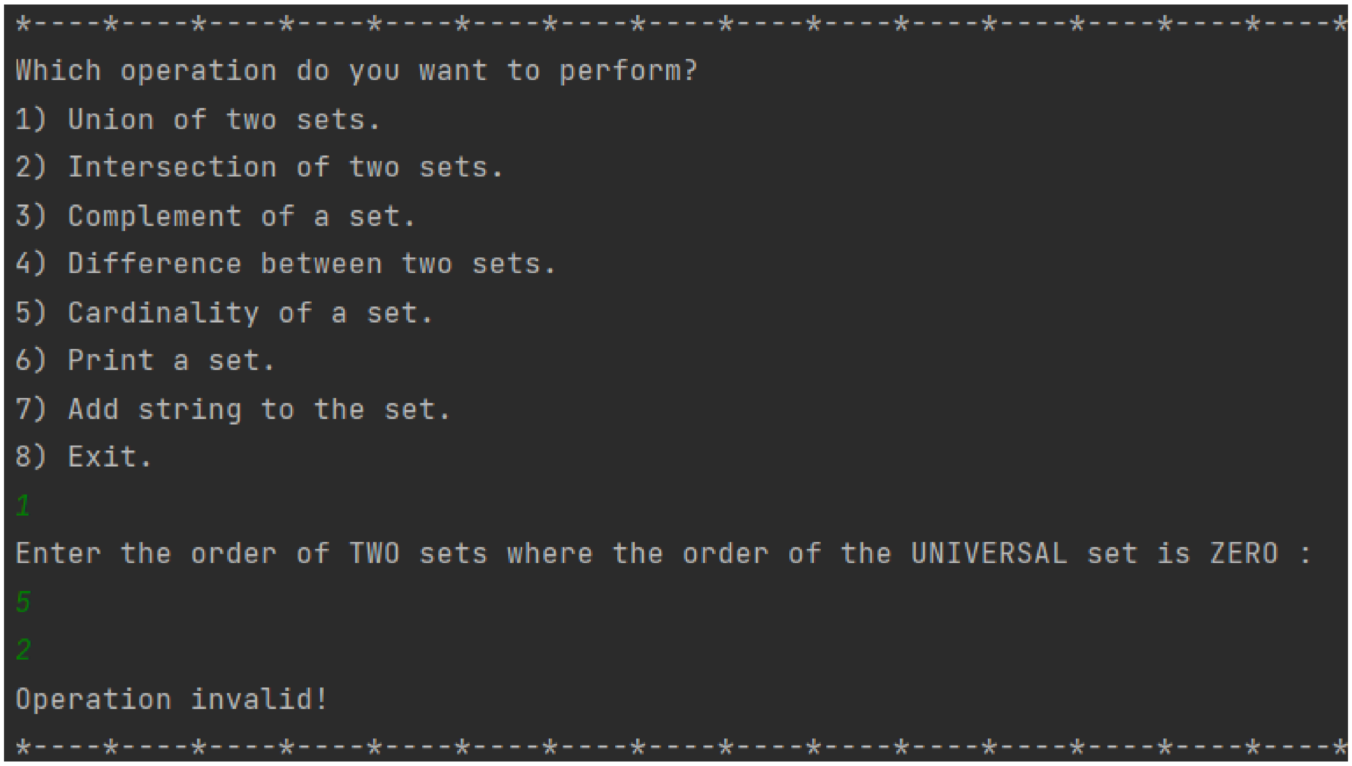
```
"C:\Program Files\Java\jdk-19\bin\java.exe" "
Enter elements of the Universe:
{a, b, c, d, e, f, g, h, i, j}
Enter number of subsets: 4
Enter size of subset: 1 3
Enter elements of subset: 1 : a c e
Enter size of subset: 2 1
Enter elements of subset: 2 : j
Enter size of subset: 3 6
Enter elements of subset: 3 : b d f h c e
Enter size of subset: 4 2
Enter elements of subset: 4 : d e
Which operation do you want to perform?
1) Union of two sets.
2) Intersection of two sets.
3) Complement of a set.
4) Difference between two sets.
5) Cardinality of a set.
6) Print a set.
7) Add string to the set.
8) Exit.
```

⇒ Union operation:



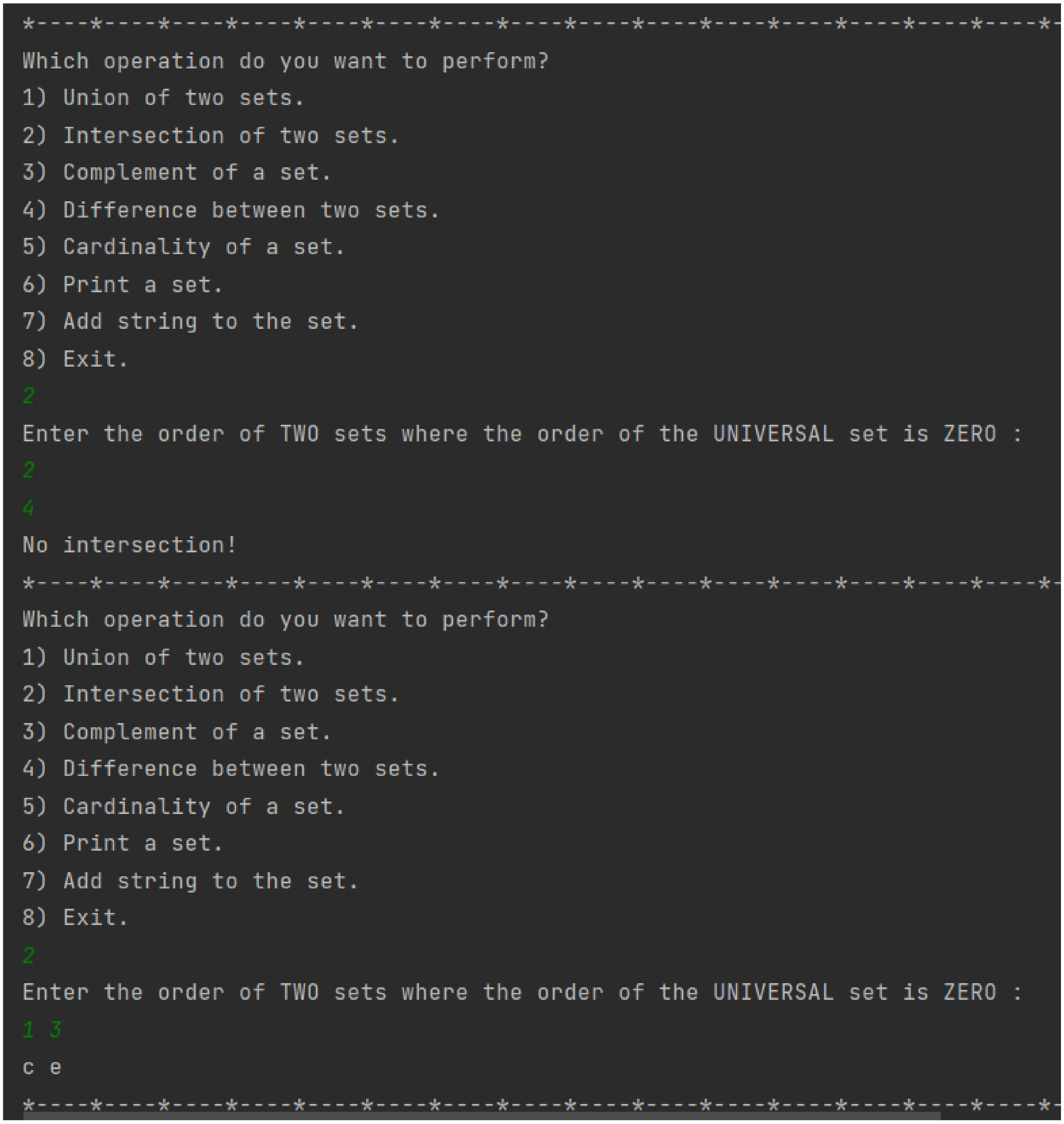
```
Which operation do you want to perform?
1) Union of two sets.
2) Intersection of two sets.
3) Complement of a set.
4) Difference between two sets.
5) Cardinality of a set.
6) Print a set.
7) Add string to the set.
8) Exit.
1
Enter the order of TWO sets where the order of the UNIVERSAL set is ZERO :
0 1
a b c d e f g h i j
*----*----*----*----*----*----*----*----*----*----*----*----*----*----*----*---
Which operation do you want to perform?
1) Union of two sets.
2) Intersection of two sets.
3) Complement of a set.
4) Difference between two sets.
5) Cardinality of a set.
6) Print a set.
7) Add string to the set.
8) Exit.
1
Enter the order of TWO sets where the order of the UNIVERSAL set is ZERO :
4
2
d e j
*----*----*----*----*----*----*----*----*----*----*----*----*----*----*----*---
```

⇒ Entering invalid set order:



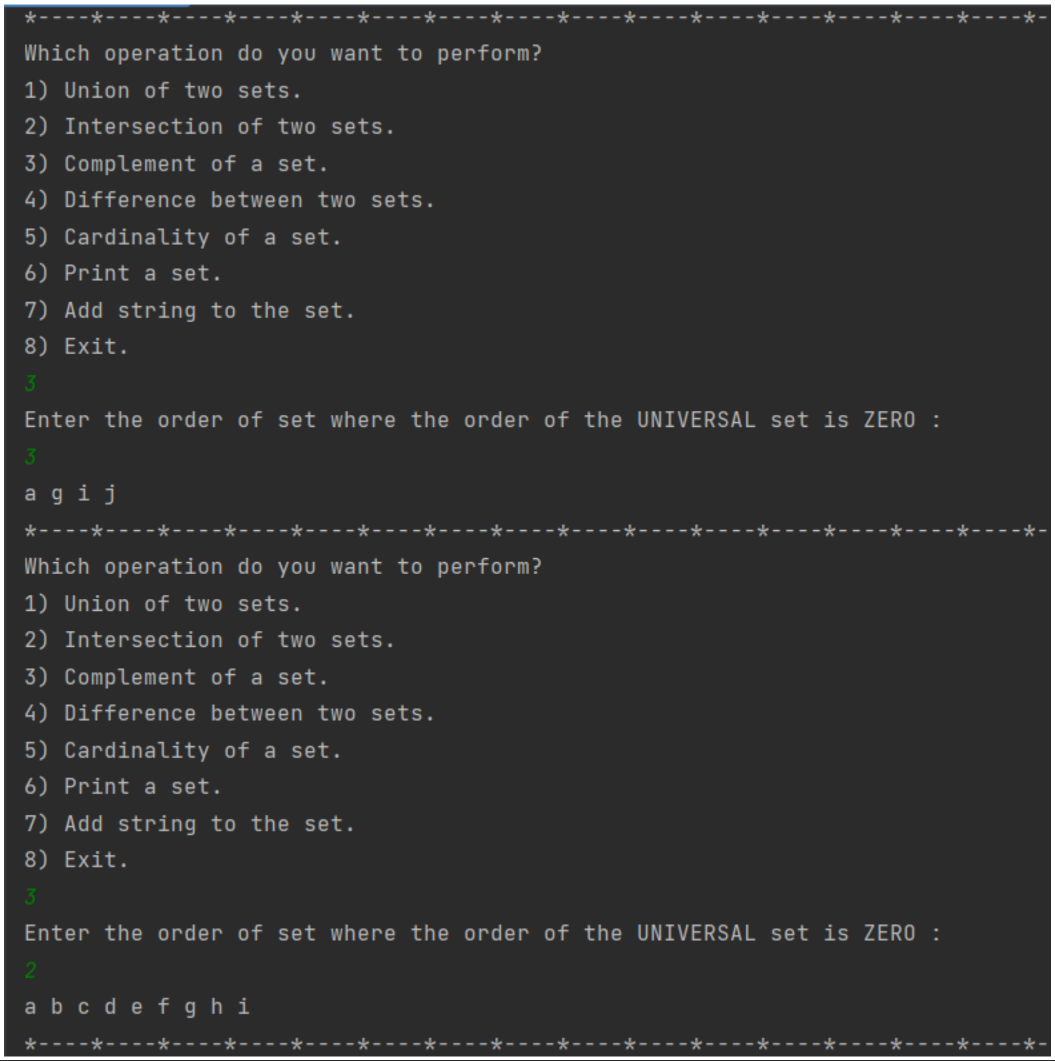
```
*----*----*----*----*----*----*----*----*----*----*----*----*----*----*----*
Which operation do you want to perform?
1) Union of two sets.
2) Intersection of two sets.
3) Complement of a set.
4) Difference between two sets.
5) Cardinality of a set.
6) Print a set.
7) Add string to the set.
8) Exit.
1
Enter the order of TWO sets where the order of the UNIVERSAL set is ZERO :
5
2
Operation invalid!
*----*----*----*----*----*----*----*----*----*----*----*----*----*----*----*
```

⇒**Intersection operation:**

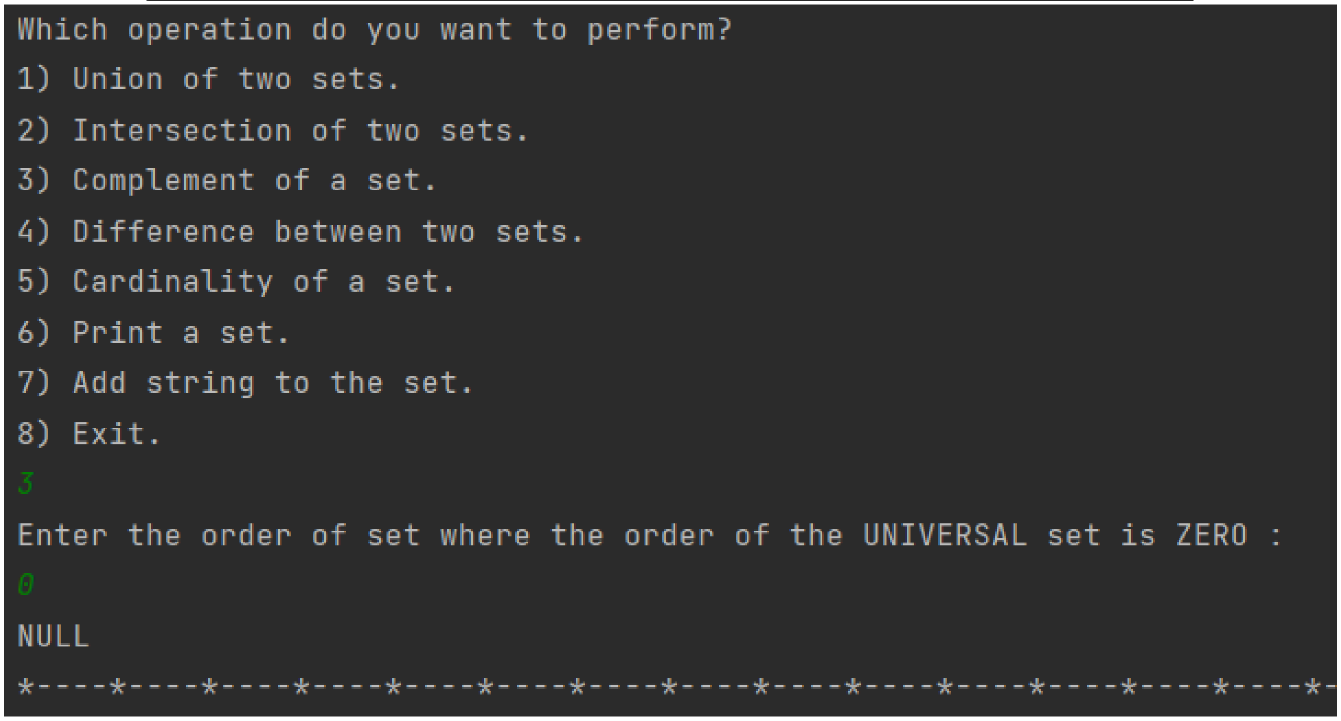


```
*----*----*----*----*----*----*----*----*----*----*----*----*----*----*----*-
Which operation do you want to perform?
1) Union of two sets.
2) Intersection of two sets.
3) Complement of a set.
4) Difference between two sets.
5) Cardinality of a set.
6) Print a set.
7) Add string to the set.
8) Exit.
2
Enter the order of TWO sets where the order of the UNIVERSAL set is ZERO :
2
4
No intersection!
*----*----*----*----*----*----*----*----*----*----*----*----*----*----*----*-
Which operation do you want to perform?
1) Union of two sets.
2) Intersection of two sets.
3) Complement of a set.
4) Difference between two sets.
5) Cardinality of a set.
6) Print a set.
7) Add string to the set.
8) Exit.
2
Enter the order of TWO sets where the order of the UNIVERSAL set is ZERO :
1 3
c e
*----*----*----*----*----*----*----*----*----*----*----*----*----*----*----*-
```

⇒ Complement operation:

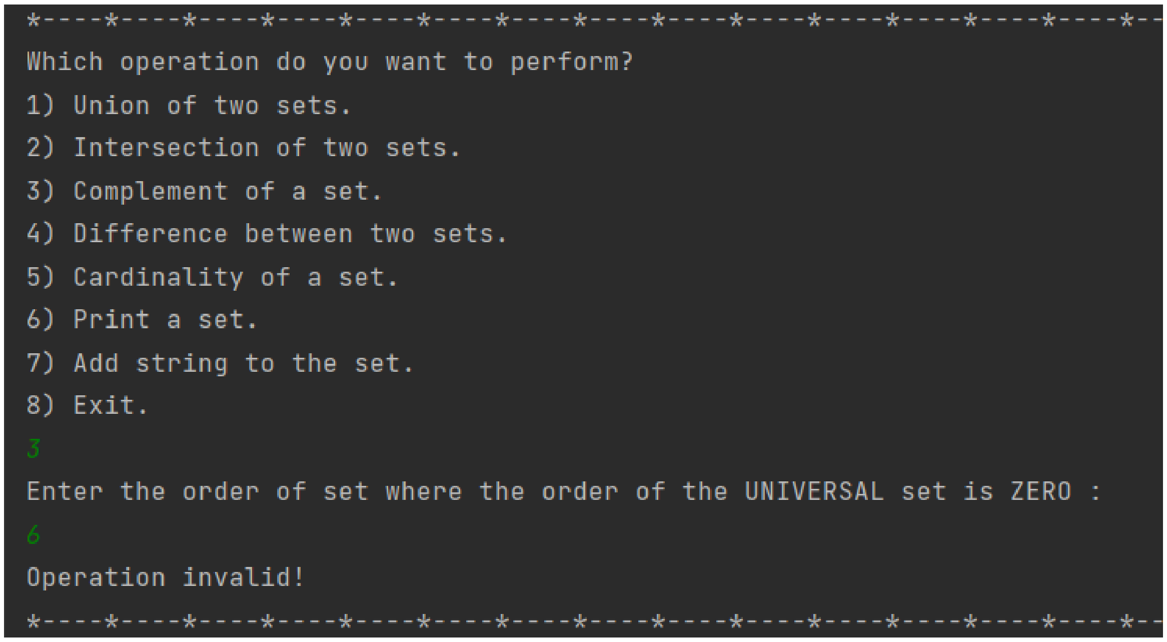


```
*----*----*----*----*----*----*----*----*----*----*----*----*----*-
Which operation do you want to perform?
1) Union of two sets.
2) Intersection of two sets.
3) Complement of a set.
4) Difference between two sets.
5) Cardinality of a set.
6) Print a set.
7) Add string to the set.
8) Exit.
3
Enter the order of set where the order of the UNIVERSAL set is ZERO :
3
a g i j
*----*----*----*----*----*----*----*----*----*----*----*----*----*-
Which operation do you want to perform?
1) Union of two sets.
2) Intersection of two sets.
3) Complement of a set.
4) Difference between two sets.
5) Cardinality of a set.
6) Print a set.
7) Add string to the set.
8) Exit.
3
Enter the order of set where the order of the UNIVERSAL set is ZERO :
2
a b c d e f g h i
*----*----*----*----*----*----*----*----*----*----*----*----*----*-
```



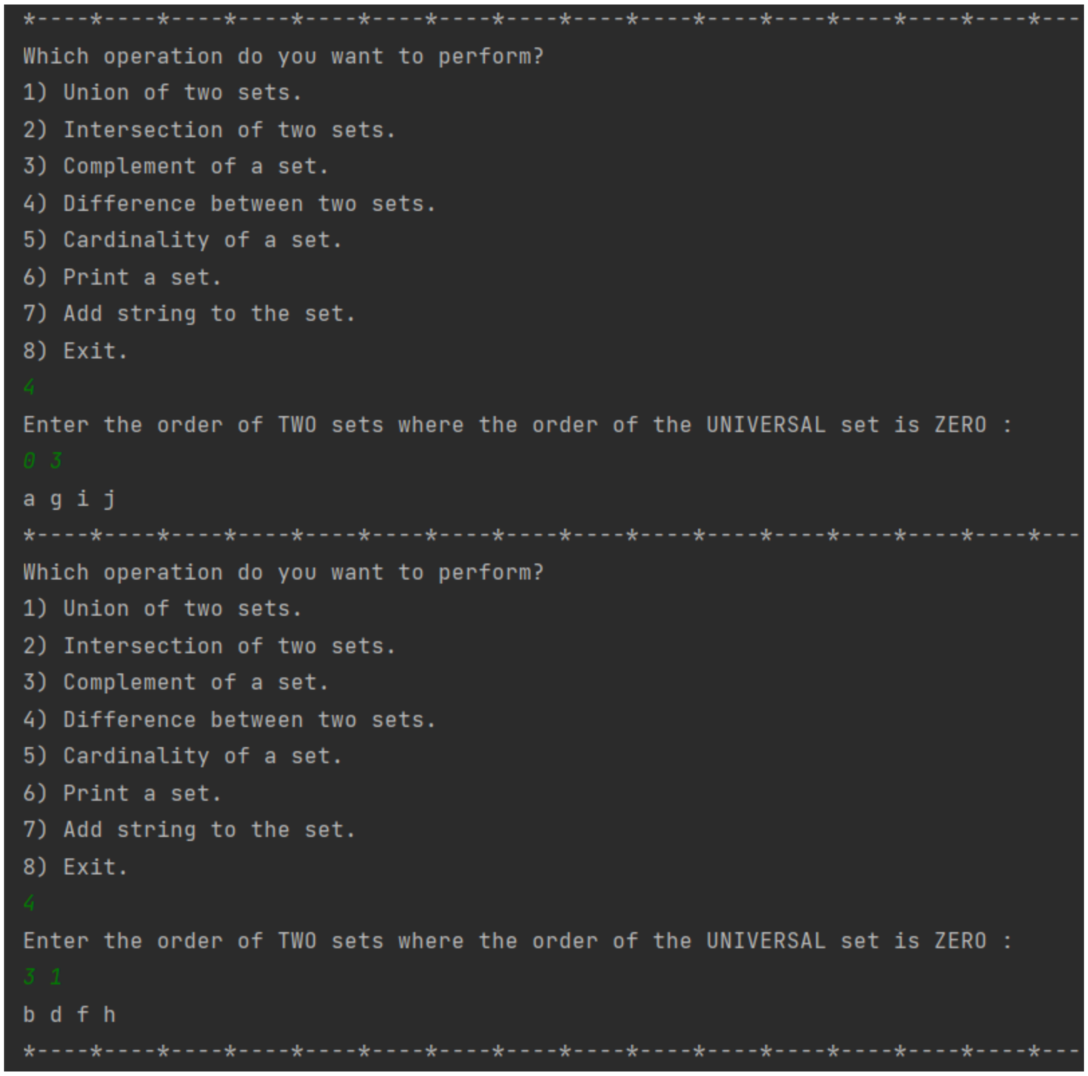
```
Which operation do you want to perform?
1) Union of two sets.
2) Intersection of two sets.
3) Complement of a set.
4) Difference between two sets.
5) Cardinality of a set.
6) Print a set.
7) Add string to the set.
8) Exit.
3
Enter the order of set where the order of the UNIVERSAL set is ZERO :
0
NULL
*----*----*----*----*----*----*----*----*----*----*----*----*----*----*-
```

⇒ Entering invalid operation:



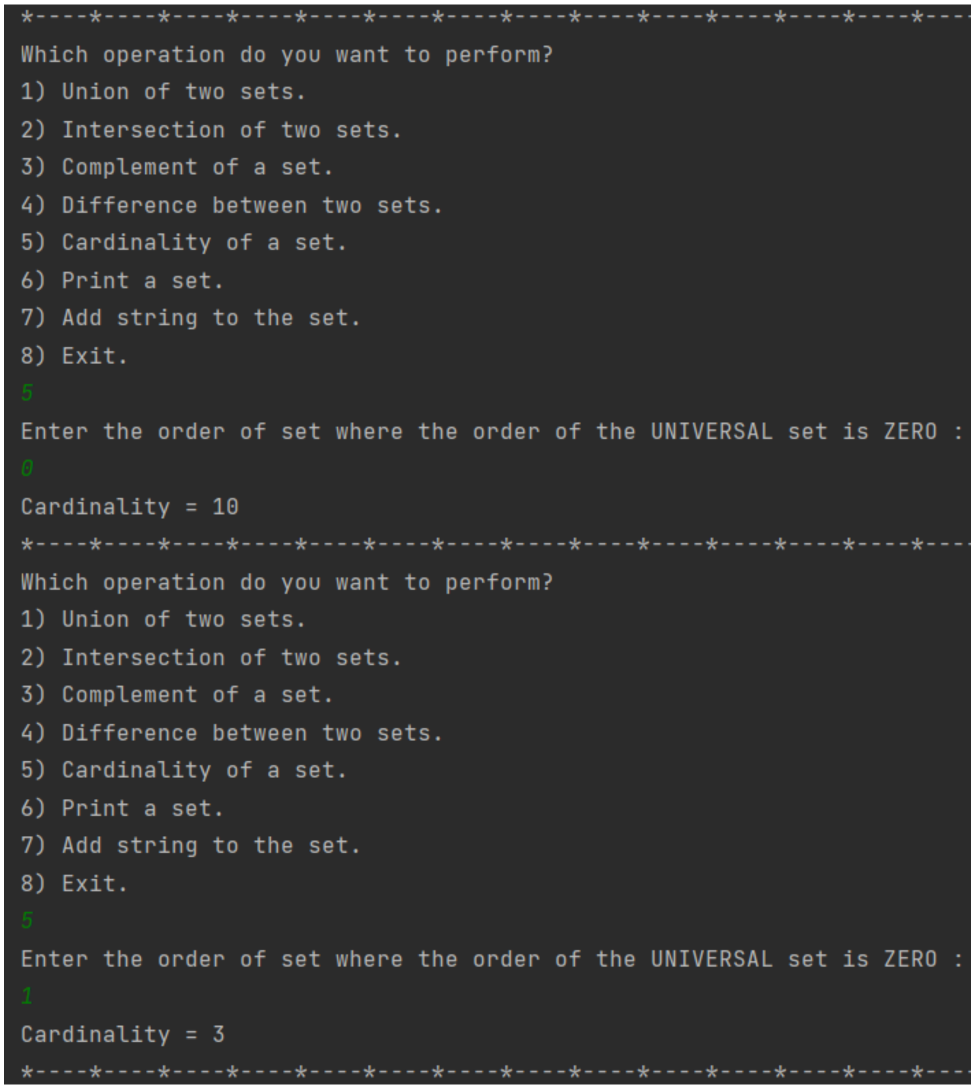
```
*----*----*----*----*----*----*----*----*----*----*----*----*----*----*--
Which operation do you want to perform?
1) Union of two sets.
2) Intersection of two sets.
3) Complement of a set.
4) Difference between two sets.
5) Cardinality of a set.
6) Print a set.
7) Add string to the set.
8) Exit.
3
Enter the order of set where the order of the UNIVERSAL set is ZERO :
6
Operation invalid!
*----*----*----*----*----*----*----*----*----*----*----*----*----*----*--
```

⇒ Difference operation:

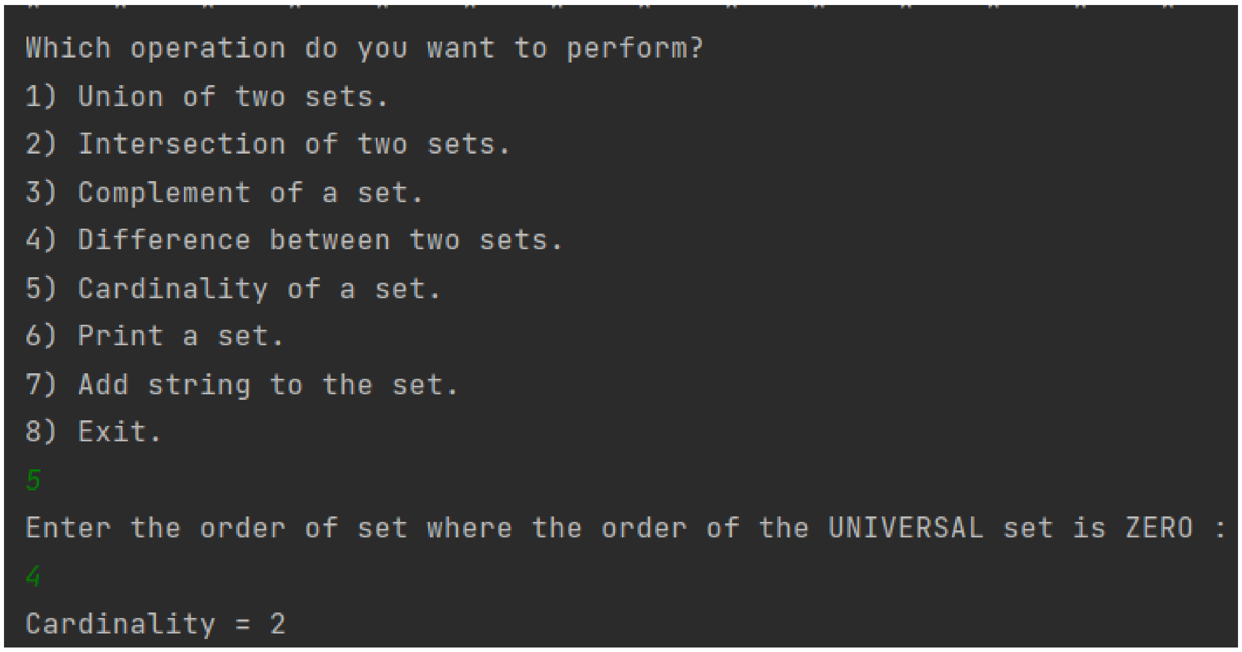


```
*----*----*----*----*----*----*----*----*----*----*----*----*----*----*---
Which operation do you want to perform?
1) Union of two sets.
2) Intersection of two sets.
3) Complement of a set.
4) Difference between two sets.
5) Cardinality of a set.
6) Print a set.
7) Add string to the set.
8) Exit.
4
Enter the order of TWO sets where the order of the UNIVERSAL set is ZERO :
0 3
a g i j
*----*----*----*----*----*----*----*----*----*----*----*----*----*----*---
Which operation do you want to perform?
1) Union of two sets.
2) Intersection of two sets.
3) Complement of a set.
4) Difference between two sets.
5) Cardinality of a set.
6) Print a set.
7) Add string to the set.
8) Exit.
4
Enter the order of TWO sets where the order of the UNIVERSAL set is ZERO :
3 1
b d f h
*----*----*----*----*----*----*----*----*----*----*----*----*----*----*---
```

⇒ Cardinality operation:

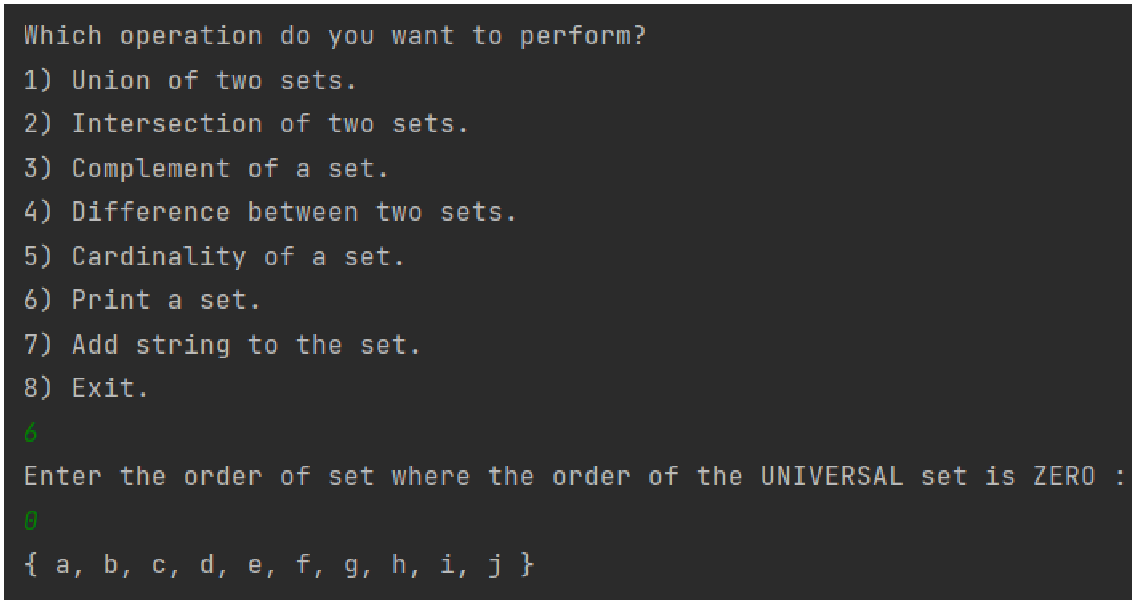


```
*----*----*----*----*----*----*----*----*----*----*----*---
Which operation do you want to perform?
1) Union of two sets.
2) Intersection of two sets.
3) Complement of a set.
4) Difference between two sets.
5) Cardinality of a set.
6) Print a set.
7) Add string to the set.
8) Exit.
5
Enter the order of set where the order of the UNIVERSAL set is ZERO :
0
Cardinality = 10
*----*----*----*----*----*----*----*----*----*----*----*---
Which operation do you want to perform?
1) Union of two sets.
2) Intersection of two sets.
3) Complement of a set.
4) Difference between two sets.
5) Cardinality of a set.
6) Print a set.
7) Add string to the set.
8) Exit.
5
Enter the order of set where the order of the UNIVERSAL set is ZERO :
1
Cardinality = 3
*----*----*----*----*----*----*----*----*----*----*----*---
```

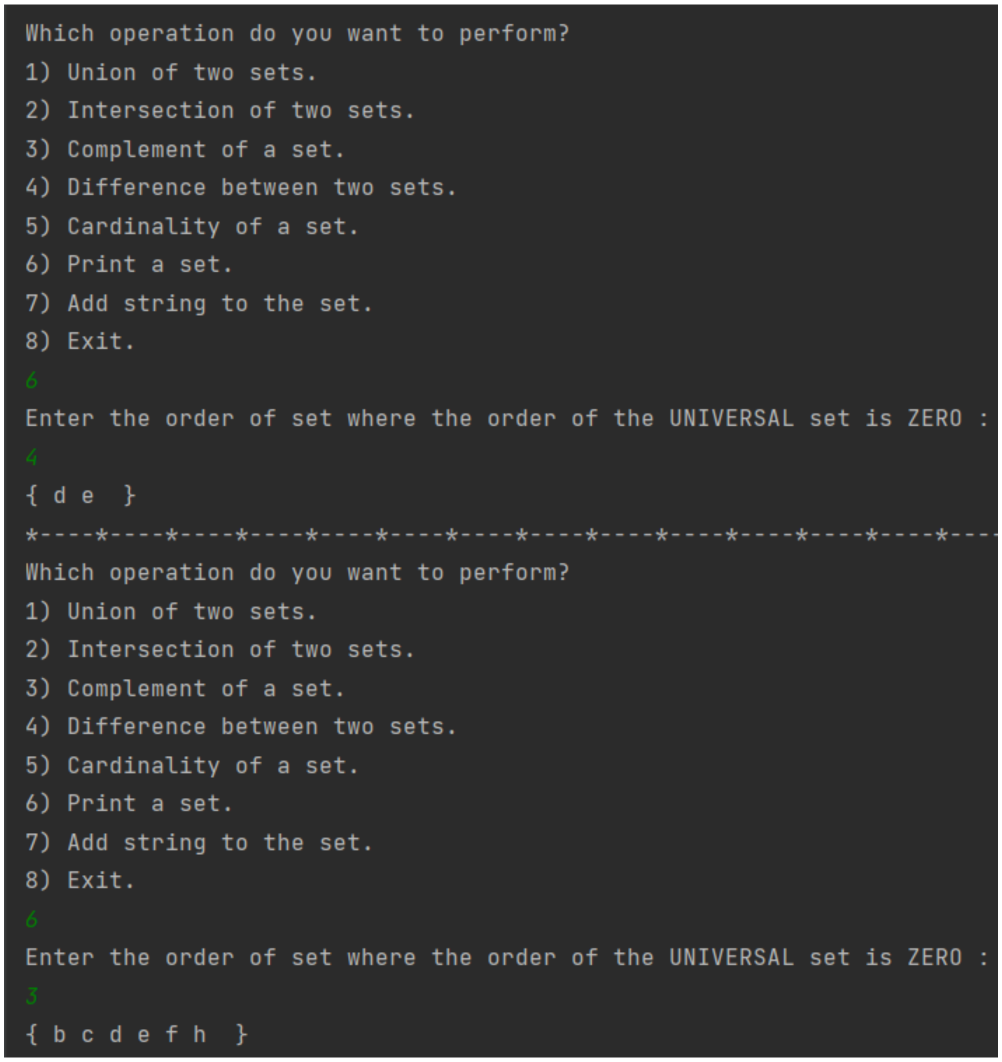


```
Which operation do you want to perform?
1) Union of two sets.
2) Intersection of two sets.
3) Complement of a set.
4) Difference between two sets.
5) Cardinality of a set.
6) Print a set.
7) Add string to the set.
8) Exit.
5
Enter the order of set where the order of the UNIVERSAL set is ZERO :
4
Cardinality = 2
```

⇒ Printing a set:

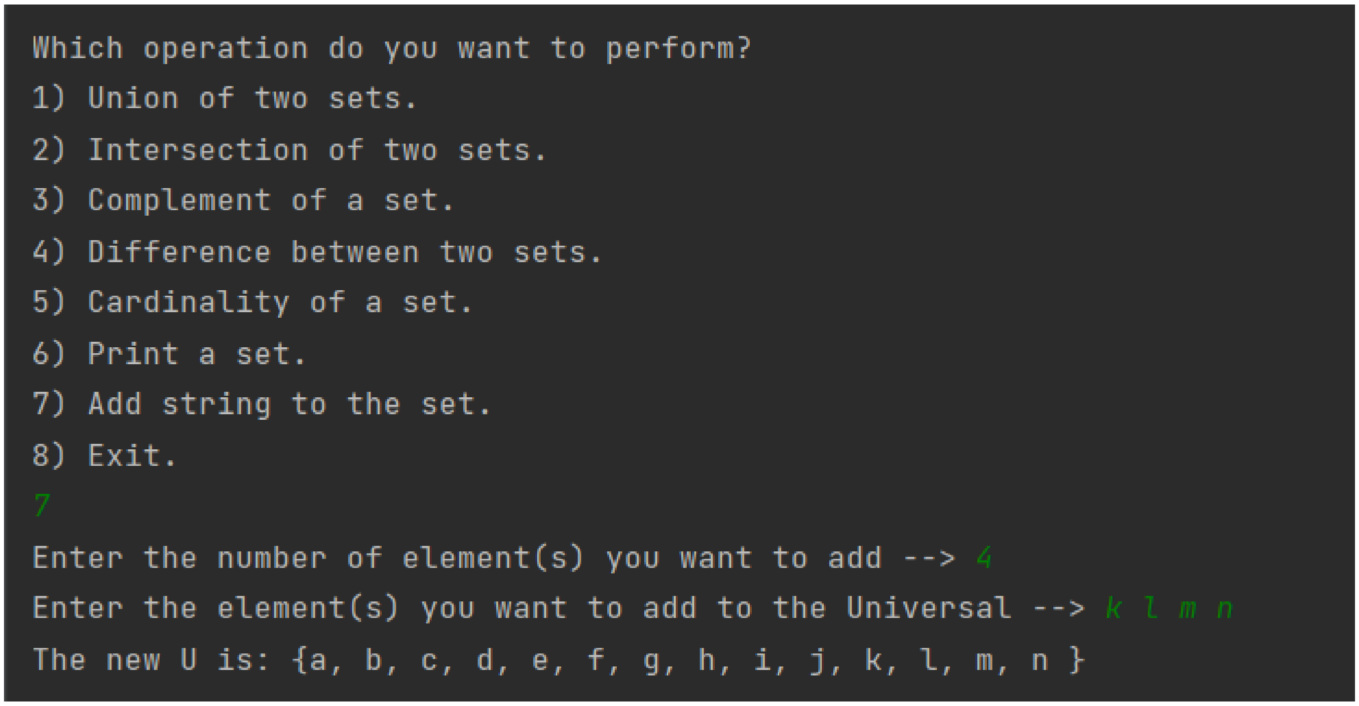


```
Which operation do you want to perform?
1) Union of two sets.
2) Intersection of two sets.
3) Complement of a set.
4) Difference between two sets.
5) Cardinality of a set.
6) Print a set.
7) Add string to the set.
8) Exit.
6
Enter the order of set where the order of the UNIVERSAL set is ZERO :
0
{ a, b, c, d, e, f, g, h, i, j }
```



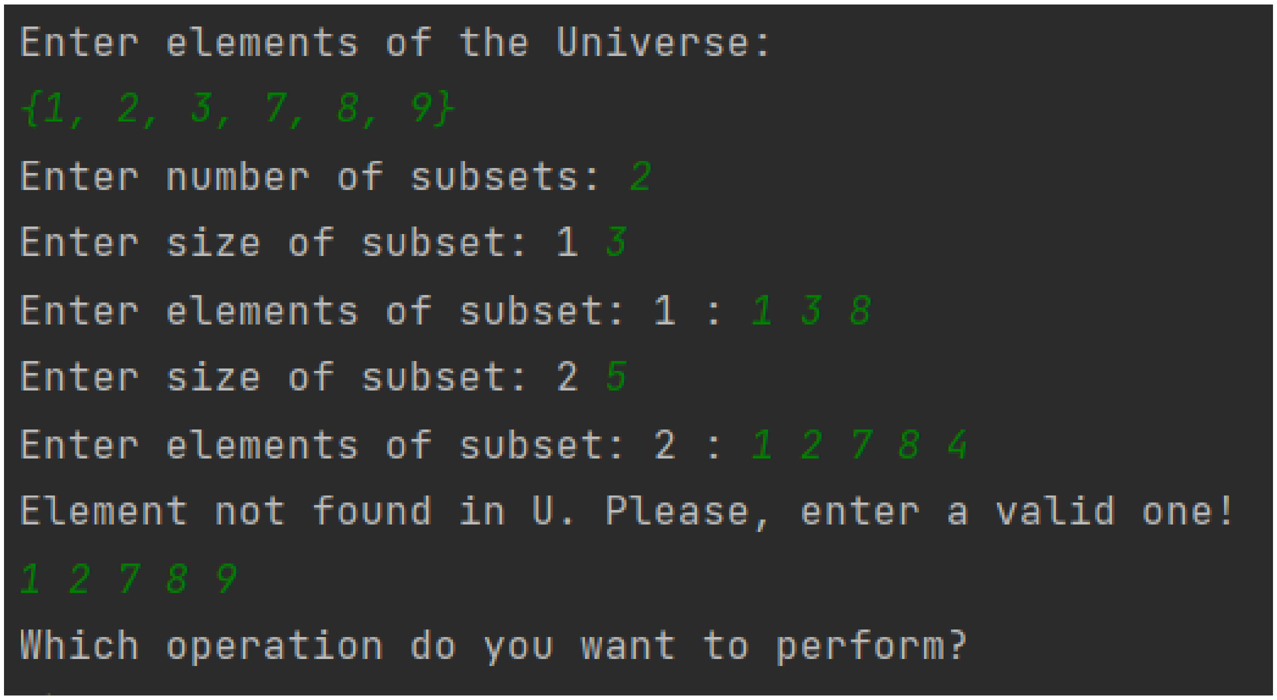
```
Which operation do you want to perform?
1) Union of two sets.
2) Intersection of two sets.
3) Complement of a set.
4) Difference between two sets.
5) Cardinality of a set.
6) Print a set.
7) Add string to the set.
8) Exit.
6
Enter the order of set where the order of the UNIVERSAL set is ZERO :
4
{ d e  }
*----*----*----*----*----*----*----*----*----*----*----*----*----*---
Which operation do you want to perform?
1) Union of two sets.
2) Intersection of two sets.
3) Complement of a set.
4) Difference between two sets.
5) Cardinality of a set.
6) Print a set.
7) Add string to the set.
8) Exit.
6
Enter the order of set where the order of the UNIVERSAL set is ZERO :
3
{ b c d e f h  }
```

⇒ **Adding string to a set:**



```
Which operation do you want to perform?
1) Union of two sets.
2) Intersection of two sets.
3) Complement of a set.
4) Difference between two sets.
5) Cardinality of a set.
6) Print a set.
7) Add string to the set.
8) Exit.
7
Enter the number of element(s) you want to add --> 4
Enter the element(s) you want to add to the Universal --> k l m n
The new U is: {a, b, c, d, e, f, g, h, i, j, k, l, m, n }
```

⇒ **Entering invalid element in a subset:**



```
Enter elements of the Universe:
{1, 2, 3, 7, 8, 9}
Enter number of subsets: 2
Enter size of subset: 1 3
Enter elements of subset: 1 : 1 3 8
Enter size of subset: 2 5
Enter elements of subset: 2 : 1 2 7 8 4
Element not found in U. Please, enter a valid one!
1 2 7 8 9
Which operation do you want to perform?
```

(5) <u>Assumptions and details you find necessary to be clarified:</u>

The elements entered to the set are distinct.

Part 3: Applications for Bits Manipulation

1. Problem statement:

1) Write a function that takes a non-empty array of integers nums, where every element appears twice except for one integer, and returns the unique integer. You must implement a solution with a linear runtime complexity and use only constant extra space. you must think for your solution using bits manipulation operation.
   (a) [Bonus] Assume there are two unique integers in the array. Implement a function that prints these two unique integers. You must solve it using bitwise operations.
2) Write a function that takes an unsigned integer and returns the number of '1' bits it.

2. Used data structures:

- **Arrays:**
  a) Name: array.
     Data type: int
     Functionality: To store elements of the array of the first part.

3. Algorithms documented using flow charts or pseudo code:

- **Documenting algorithms through pseudocode:**

  First: 4 functions are made.

  (1) **singleOccurence():** Which is given an array as a parameter and using BITWISE XOR, the single element which didn't appear twice is returned.
  (2) **found():** By looping on a given array as a parameter, we check if a certain element is found in the array.
  (3) **doubleOccurence():** 2 elements are returned, those which didn't appear twice.
  (4) **numberOnes():** which returns the number of ones in a given number by AND-ing the number with one, then, performing a right shift by one to the number till it equals zero. On each AND-ing process, if the result = 1, a count variable is incremented.

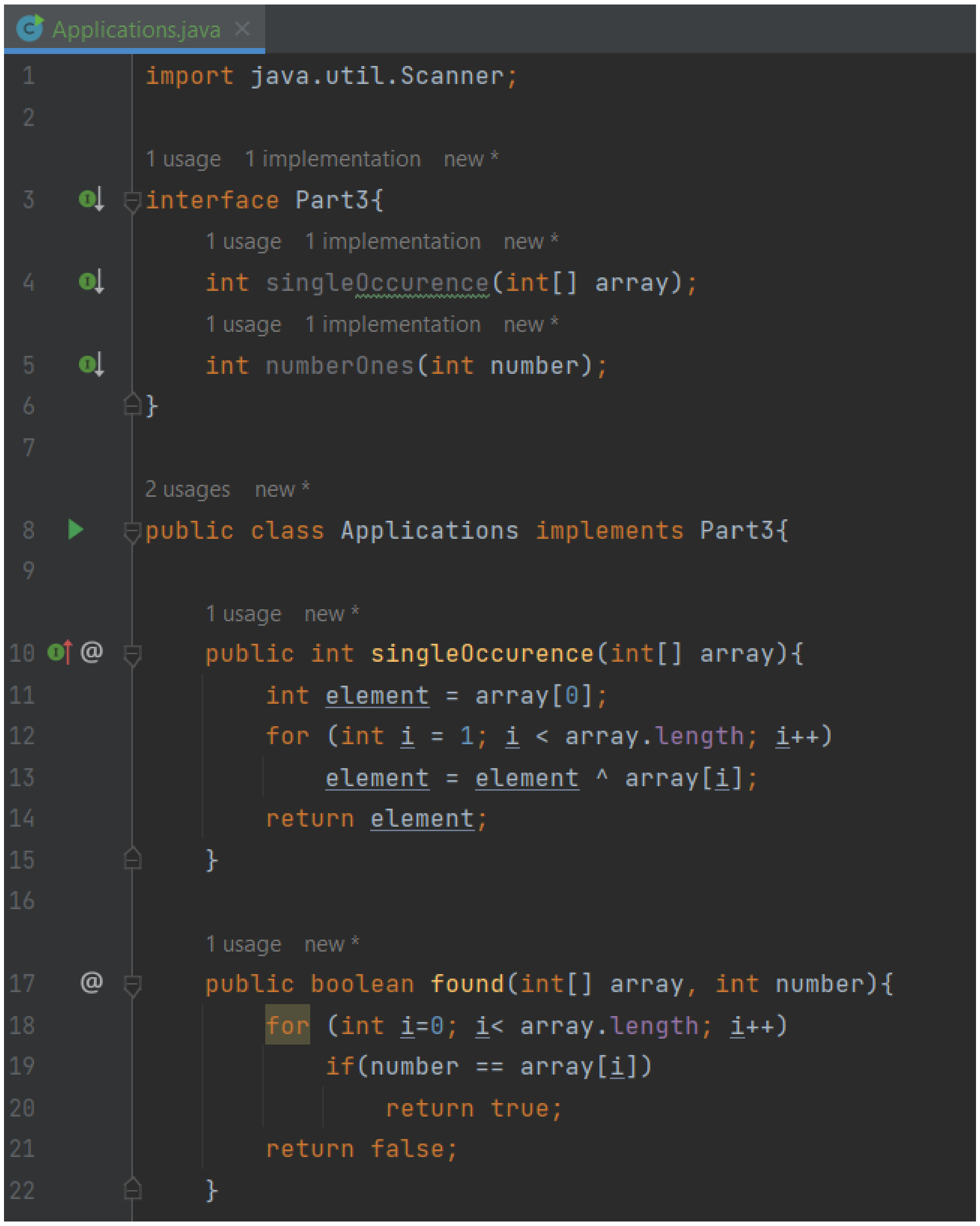
  In the main function:

  1. The menu is printed which asks the user to choose one of the following operations: 1) The single occurence in an array. 2) Number of 1s. 3)Exit.
  2. The user enters a number corresponding to the function he wants to perform and that number is stored in a variable named "function".
  3. As long as the entered function is bigger than zero and smaller than 3, a switch case works.
  4. If the user enters "1", he is asked to enter array size and elements. The singleOccurence() function is called with the array given as a parameter. If the answer was found, through the

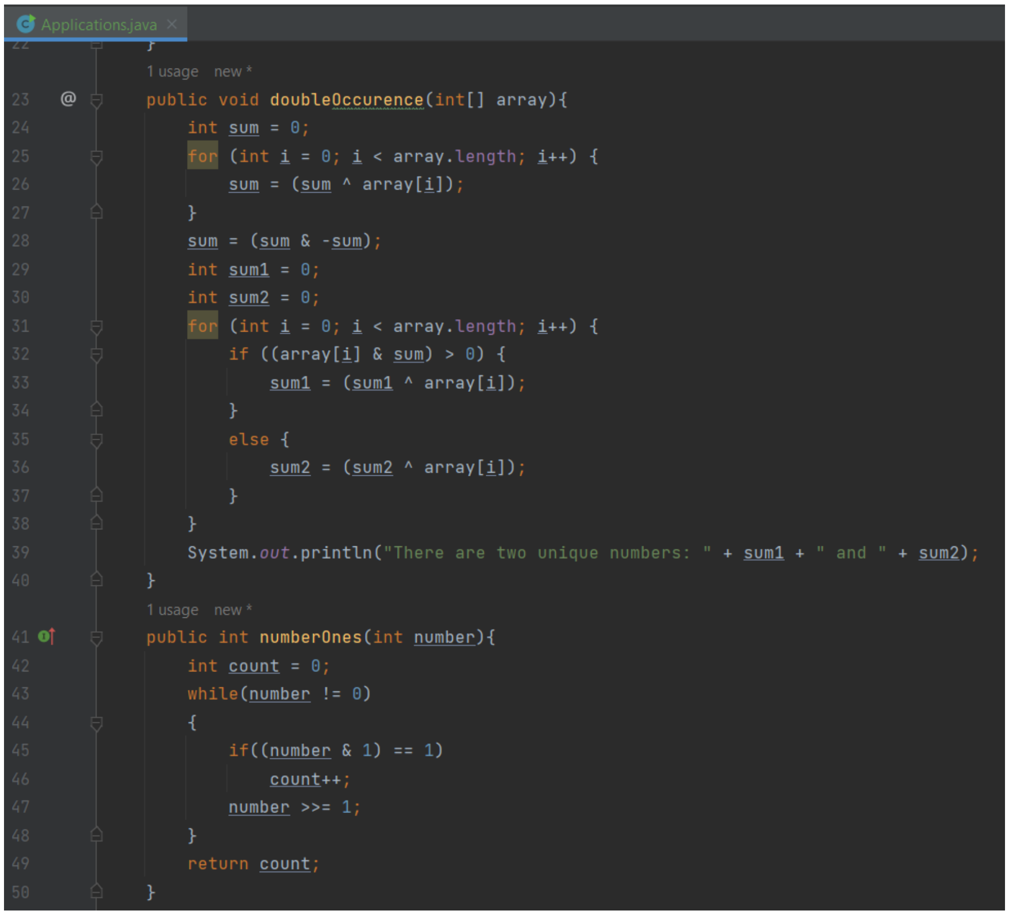
"found()" function, in the original array, then the result is printed. If not, then the doubleOccurence() function is called with the array given as a parameter. 2 numbers are printed, then.

5. If the user enters "2", he is asked to enter a number. Then, the "numberOnes()" function is called and the result is printed.
6. Every time an operation is performed, the menu is reprinted, the user is asked to enter the operation number he wishes to perform and the program continues on till the user terminates it.

4. Code Snippets:

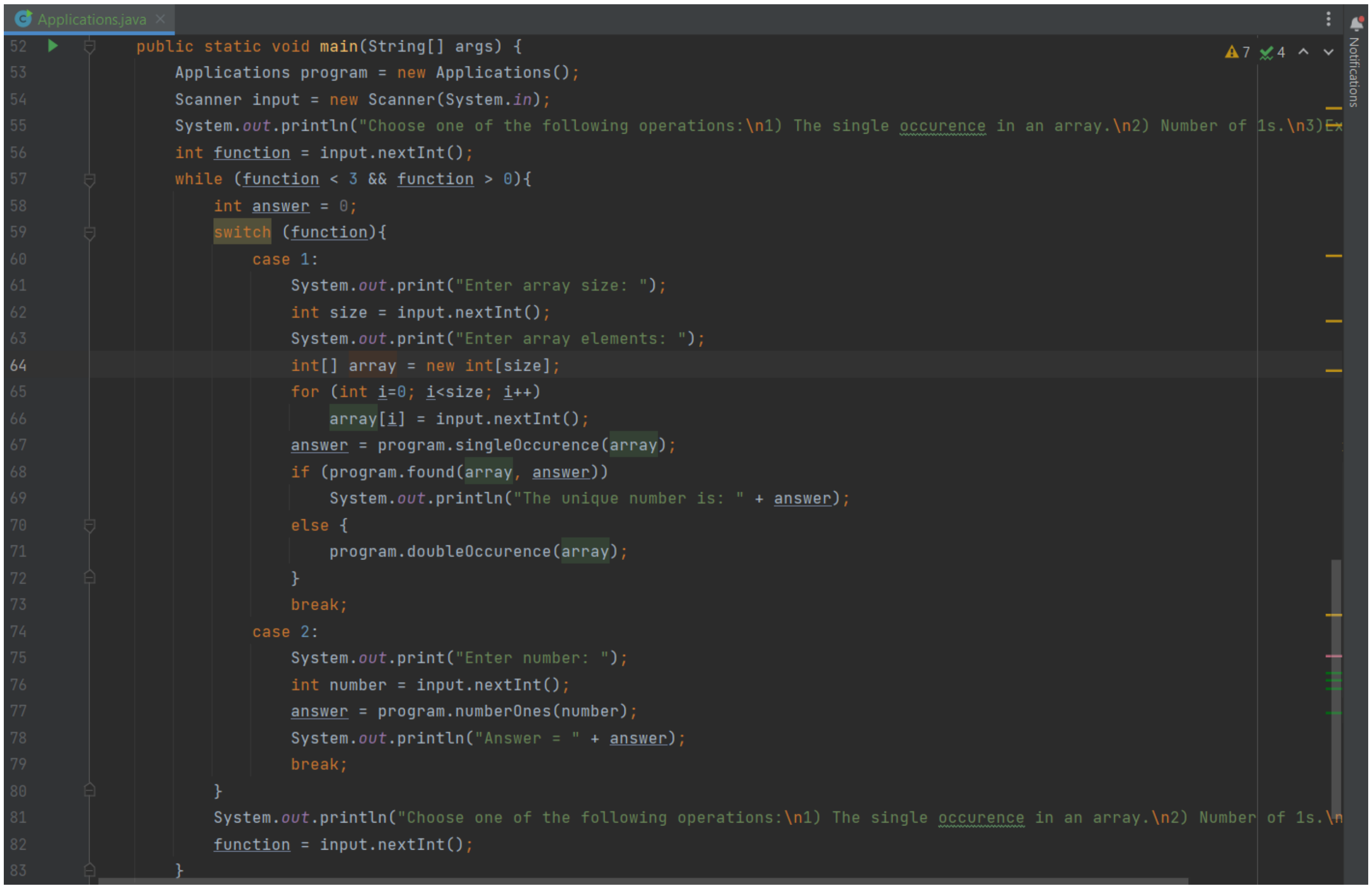

Applications.java

```java
import java.util.Scanner;

1 usage   1 implementation   new *
interface Part3{
    1 usage   1 implementation   new *
    int singleOccurence(int[] array);
    1 usage   1 implementation   new *
    int numberOnes(int number);
}

2 usages   new *
public class Applications implements Part3{

    1 usage   new *
    public int singleOccurence(int[] array){
        int element = array[0];
        for (int i = 1; i < array.length; i++)
            element = element ^ array[i];
        return element;
    }

    1 usage   new *
    public boolean found(int[] array, int number){
        for (int i=0; i< array.length; i++)
            if(number == array[i])
                return true;
        return false;
    }
```



Applications.java

```java
    }
    1 usage   new *
    public void doubleOccurence(int[] array){
        int sum = 0;
        for (int i = 0; i < array.length; i++) {
            sum = (sum ^ array[i]);
        }
        sum = (sum & -sum);
        int sum1 = 0;
        int sum2 = 0;
        for (int i = 0; i < array.length; i++) {
            if ((array[i] & sum) > 0) {
                sum1 = (sum1 ^ array[i]);
            }
            else {
                sum2 = (sum2 ^ array[i]);
            }
        }
        System.out.println("There are two unique numbers: " + sum1 + " and " + sum2);
    }
    1 usage   new *
    public int numberOnes(int number){
        int count = 0;
        while(number != 0)
        {
            if((number & 1) == 1)
                count++;
            number >>= 1;
        }
        return count;
    }
```

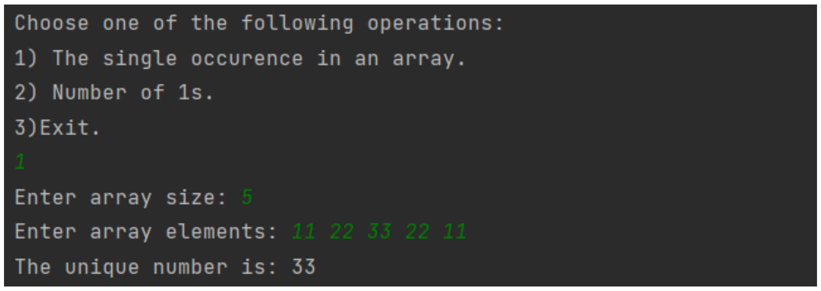


Applications.java

```java
    public static void main(String[] args) {
        Applications program = new Applications();
        Scanner input = new Scanner(System.in);
        System.out.println("Choose one of the following operations:\n1) The single occurence in an array.\n2) Number of 1s.\n3)E
        int function = input.nextInt();
        while (function < 3 && function > 0){
            int answer = 0;
            switch (function){
                case 1:
                    System.out.print("Enter array size: ");
                    int size = input.nextInt();
                    System.out.print("Enter array elements: ");
                    int[] array = new int[size];
                    for (int i=0; i<size; i++)
                        array[i] = input.nextInt();
                    answer = program.singleOccurence(array);
                    if (program.found(array, answer))
                        System.out.println("The unique number is: " + answer);
                    else {
                        program.doubleOccurence(array);
                    }
                    break;
                case 2:
                    System.out.print("Enter number: ");
                    int number = input.nextInt();
                    answer = program.numberOnes(number);
                    System.out.println("Answer = " + answer);
                    break;
            }
            System.out.println("Choose one of the following operations:\n1) The single occurence in an array.\n2) Number of 1s.\n
            function = input.nextInt();
        }
```

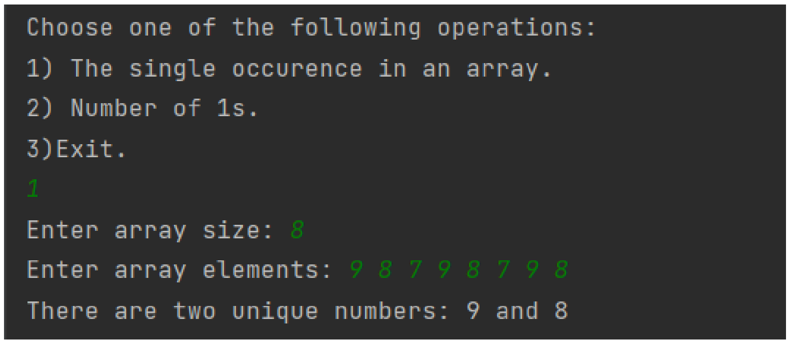
5. Sample runs and different test cases:

⇒ **First operation:**



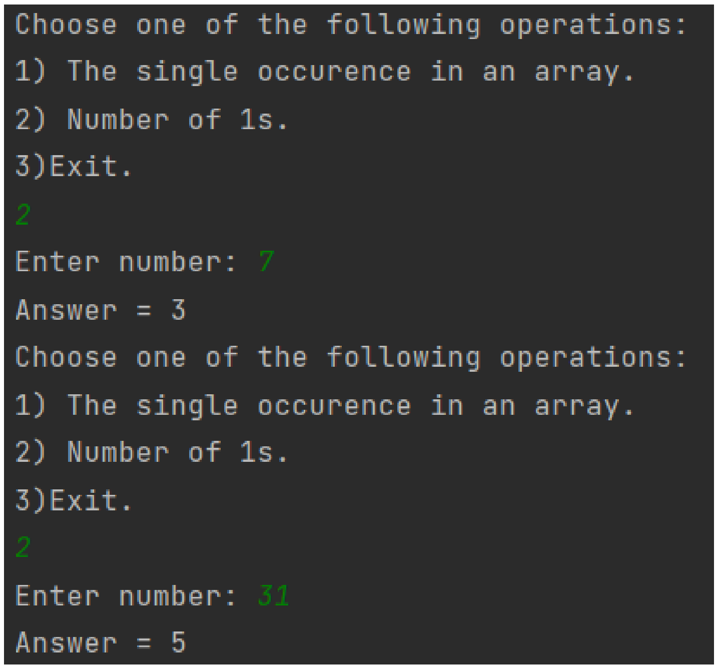
```
Choose one of the following operations:
1) The single occurence in an array.
2) Number of 1s.
3)Exit.
1
Enter array size: 5
Enter array elements: 11 22 33 22 11
The unique number is: 33
```

⇒ **The bonus part:**



```
Choose one of the following operations:
1) The single occurence in an array.
2) Number of 1s.
3)Exit.
1
Enter array size: 8
Enter array elements: 9 8 7 9 8 7 9 8
There are two unique numbers: 9 and 8
```

⇒ **The second operation:**



```
Choose one of the following operations:
1) The single occurence in an array.
2) Number of 1s.
3)Exit.
2
Enter number: 7
Answer = 3
Choose one of the following operations:
1) The single occurence in an array.
2) Number of 1s.
3)Exit.
2
Enter number: 31
Answer = 5
```

6. Assumptions and details you find necessary to be clarified:

No assumptions made.