Set Operations Using Bit Manipulation

Team Members

- Abdel-Rahman Amgad Hassan (ID: 22010871)
- Mahmoud Hesham Mohamed (ID: 22011201)

1. Problem Statement

The goal of this project is to implement a set data structure with all its operations using bit manipulation. The implementation should support:

- Basic set operations (union, intersection, complement, difference)
- Set membership testing
- Set cardinality calculation
- Efficient memory usage as we use bit representation

2. Data Structures Used

- **Bit Representation:** Sets are represented using integers where each bit position corresponds to an element in the universe
- **Universe Storage:** List<String> maintains the mapping between elements and their bit positions
- **SetOperations Class:** Encapsulates both the bit representation and operations

3. Code Logic

3.1 Bit Operations

The foundation of our implementation relies on four fundamental bit operations:

- **getBit(number, position):**

$$\text{result} = (number \gg position)\&1 \tag{1}$$

- **setBit(number, position):**

$$\text{result} = number \mid (1 \ll position) \tag{2}$$

- **clearBit(number, position):**

$$\text{result} = number \,\&\, \sim (1 \ll position) \tag{3}$$

- **updateBit(number, position, value):**

$$\text{result} = (number \,\&\, \sim (1 \ll position)) \mid (value \ll position) \tag{4}$$

3.2 Set Operations

The set operations are implemented using bitwise operations:

- **Union:** $A \cup B = A \mid B$
- **Intersection:** $A \cap B = A \,\&\, B$
- **Complement:** $\overline{A} = \sim A \,\&\, mask$
- **Difference:** $A - B = A \,\&\, \sim B$

4. Design Choices

- **Bit Manipulation:** Using bits to represent sets provides:
  - Space efficiency: One integer can represent up to 32 elements
  - Time efficiency: Set operations become simple bitwise operations
  - Ensures non repition in set members
- **Immutable Universe:**
  - The universe is fixed at creation time
  - Ensures consistency across all set operations
- **Error Handling:**
  - Robust validation for universe creation
  - Checks for invalid elements

5. Assumptions

- The universe size is limited to 32 elements (int size in Java), For larger universes we can use large int or use a string.

6. Test Results

```
Enter the size of the universe:
5
Enter 5 unique elements for the universe:
A1
B1
A2
B2
C3
Universe created: [A1, B1, A2, B2, C3]

Enter the number of sets you want to create:
2

Creating Set 1
Enter the number of elements in this set:
2
Enter 2 elements (must be from the universe):
A1
A2
Set 1 created: Set{elements=[A1, A2]}

Creating Set 2
Enter the number of elements in this set:
2
Enter 2 elements (must be from the universe):
B1
B2
Set 2 created: Set{elements=[B1, B2]}

Set Operations Menu:
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
0. Exit
Enter your choice:
```

Figure 1: Input example

```
Set Operations Menu:
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
0. Exit
Enter your choice:
1
Select first set:
1. Set{elements=[A1, A2]}
2. Set{elements=[B1, B2]}
1
Select second set:
1. Set{elements=[A1, A2]}
2. Set{elements=[B1, B2]}
2
Union result: Set{elements=[A1, B1, A2, B2]}
```

(a) Union operation test

```
Set Operations Menu:
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
0. Exit
Enter your choice:
3
Select a set:
1. Set{elements=[A1, A2]}
2. Set{elements=[B1, B2]}
1
Complement result: Set{elements=[B1, B2, C3]}
```

(b) Complement Operation Result

```
Set Operations Menu:
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
0. Exit
Enter your choice:
5
Select a set:
1. Set{elements=[A1, A2]}
2. Set{elements=[B1, B2]}
1
Cardinality: 2
```

(a) Cardinality

```
4. Difference between two sets
5. Cardinality of a set
6. Print a set
0. Exit
Enter your choice:
4
Select first set:
1. Set{elements=[A1, A2]}
2. Set{elements=[B1, B2]}
1
Select second set:
1. Set{elements=[A1, A2]}
2. Set{elements=[B1, B2]}
2
Difference result: Set{elements=[A1, A2]}
```

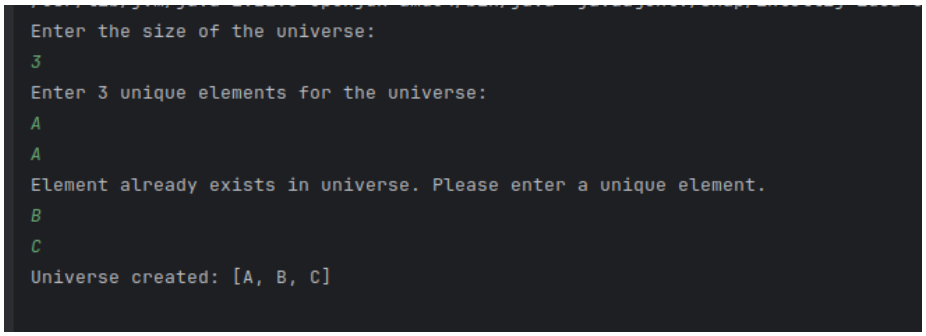
(b) Difference result

```
Set Operations Menu:
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
0. Exit
Enter your choice:
2
Select first set:
1. Set{elements=[A1, A2]}
2. Set{elements=[B1, B2]}
1
Select second set:
1. Set{elements=[A1, A2]}
2. Set{elements=[B1, B2]}
2
Intersection result: Set{elements=[]}
```

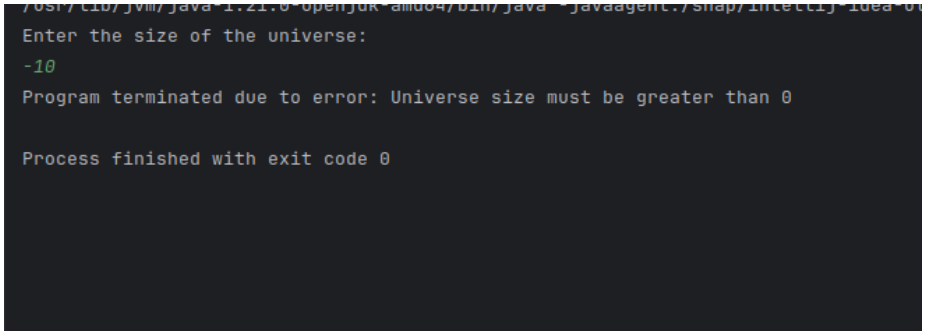
(a) Intersection Result

```
Set Operations Menu:
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
0. Exit
Enter your choice:
6
Select a set to print:
1. Set{elements=[A1, A2]}
2. Set{elements=[B1, B2]}
2
Set: Set{elements=[B1, B2]}
```

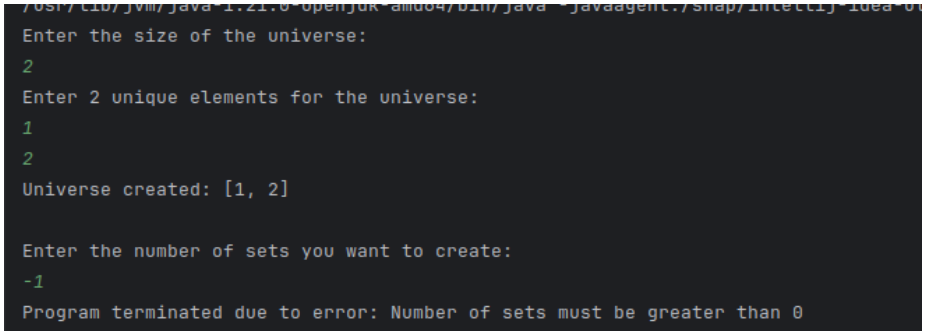
(b) Print set operation test



(a) User input validation



(a) User input validation



(a) User input validation