Name: Hussein Mohamed Mansour ID: 20010499
Name: Amr Ahmed Abd El Azim Gomaa ID: 20011037

Discrete: Lab I

1 - Problem statement:

Part 1:- Basic Bit Operations:-
You have to implement 4 bits operations, so your programe might allow user 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(an integer, 0 (false) or 1(true) in the number at position, according to its binary representation. The least significant bit in a number is position 0 and return number after update .

Part 2:- Sets Operations using Bits manipulation
Write a program that takes
1.An input a list of strings as a Universe
2.Then takes another input a number of sets (that are subsets of the universe)
3.Then ask the user about the operations they want to perform (3 required features to be implemented in this assignment):
(a)Union of two sets.
(b)Intersection of two sets
(c)Complement of a set

Part 3:- Applications for bits manipulation

1.Given a non-empty array of integers nums, every element appears twice except for one. Find that single one.

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.

2.Write a function that takes an unsigned integer and returns the number of ’1’ bits it.

2- Used Data Structures:

Part 1: No used data structures.

Part 2: One-Dimensional Arrays to represent the universe and the sets:

String[] universe: represent the values of the universe as strings.

Int[] sets: represent the values of each set at index i with its representative value.

ArrayList<String> ans: represent the result set after doing set operation on it.

Part 3:

Int[] arr : used to represent the elements of the array to get the number that has a single occurrence.

3- Algorithms used documented using flow chars and pseudo code

Part 1:

- getBit function: returns the bit either 0 , 1 of a given number.
  Function takes two parameters: the number and the position.

  ```
  Int bit = 0
  Int val = (1 << position) & number
  If val > 0 then bit = 1;
  else bit = 0;
  ```

- setBit function: returns the number after setting the bit.
  Function takes two parameters: the number and the position.

  ```
  number = number | (1 << position)
  ```

- clearBit function: returns the number after clearing the bit.
  Function takes two parameters: the number and the position.

  ```
  number = number & ~( 1 << position)
  ```

- updateBit function: returns the number after setting or clearing the bit at a certain position.

  ```
  If value != 0 then
    number = setBit(number , position)
  else
    Number = clearBit(number, position)
  ```

Part 2:

- Hashing set function takes the universe array and arr of the set. This function returns a number which represents the set.

```
Int num = 0
for(int j in arr)
        for(int k in universe)
                if(universe[k] == arr[j] )
                        num = setBIt(num,universe.length - 1- k)
                        Break
 return num
```

- Union function: returns the union of two numbers
  This function takes two numbers by using bitwise or.

```
number = num1 | num2
```

- Intersection function returns the intersection of two numbers
  This function takes two numbers by using bitwise and.

```
number = num1 & num2
```

- Complement function returns the complement of a number
  This function takes a number and complements it using bitwise not.

```
number = ~num
```

Part 3:

- singleOccurence function returns the number that have occurred once. This function takes one parameter: the array of the elements.

  ```
  Int res = 0;
  for(int i in arr)
        res ^= arr[i]
  return res
  ```

- NumOfBits function returns the number of 1 bits in a number This function takes one parameter: a number.

  ```
  Int cnt = 0;
  while( number > 0 )
        cnt += (number & 1)
        number = number >> 1
  Return number
  ```

4- Code snippets:-

Part 1:-

```
public class  BitOperations implements IOperations {

    @Override
    public int getBit(int number, int position) {

        int bit = 0;
        int val = (1 << position) & number; // 1011000
        if (val > 0) {
            bit = 1;
        }
        return bit;

    }
```

```
    @Override
    public int clearBit(int number, int position) {

        number &= ~(1 << position);
        return number;

    }

    @Override
    public int updateBit(int number, int position, boolean value) {

        if (value) {
            number = setBit(number, position);
        }

        else {
            number = clearBit(number, position);
        }
        return number;

    }
}
```

```java
@Override
public int setBit(int number, int position) {

    number |= (1 << position);
    return number;

}
```

Part 2:-

```java
@Override
public int Union(int num1, int num2) {
    return num1 | num2;
}

@Override
public int Intersection(int num1, int num2) {
    return num1 & num2;
}

@Override
public int Complement(int num){
    return ~num;
}
```

Part 3:-

```java
import java.util.Scanner;

public class BitsManipulation {


    public int singleOcurrence(int[] arr){

        int res = 0;
        for(int i=0;i<arr.length;i++){
            res ^= arr[i];
        }
        return res;

    }

    public int NumOfBits(int number){

        int cnt = 0;
        while(number > 0){
            cnt += (number & 1);
            number >>= 1;
        }
        return cnt;

    }

}
```

5- Sample runs and test cases:

- Part 1:

```
1 : getBit
2 : setBit
3 : clearBit
4 : updateBit
5 : exit
Enter the number of a function you want : 2
enter the number: 8
enter the position: 0
The result equal : 9
1 : getBit
2 : setBit
3 : clearBit
4 : updateBit
5 : exit
Enter the number of a function you want : 4
enter the number: 9
enter the position: 0
enter the value (true or false) : false
The result equal : 8
```

- Part 2:

```
1 : Union of two sets
2 : Intersection of two sets
3 : Complement of a set
4 : exit
Enter the number of a function you want : 1
Enter the number of the first set: 1
Enter the number of the second set: 3
The result of the operation:
1 2 5 6
1 : Union of two sets
2 : Intersection of two sets
3 : Complement of a set
4 : exit
Enter the number of a function you want : 2
Enter the number of the first set: 3
Enter the number of the secong set: 4
The result of the operation:
```

- Part 3:

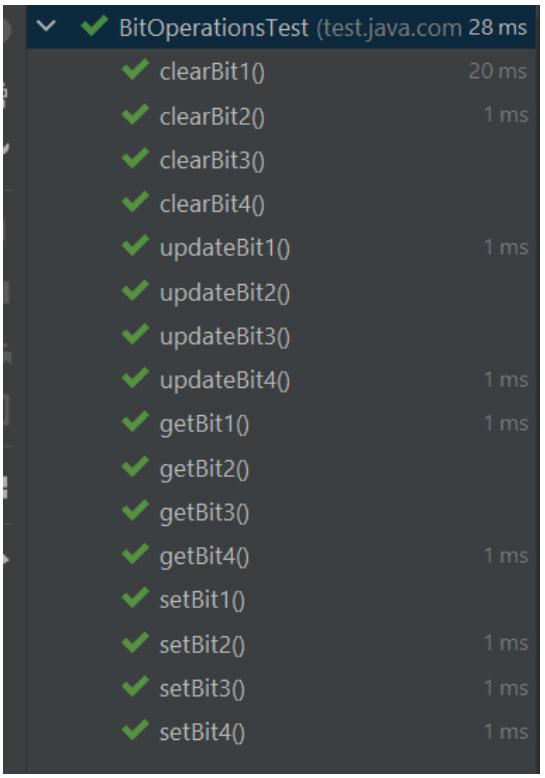
```
1 : Get single ocuurence of a number
2 : Number of set bits
3 : Exit
Enter the number of a function you want : 1
Enter the size of the array : 5
Enter the elements of the array :1 2 1 2 3
The result equal : 3
1 : Get single ocuurence of a number
2 : Number of set bits
3 : Exit
Enter the number of a function you want : 2
Enter the number: 18
The result equal : 2
1 : Get single ocuurence of a number
2 : Number of set bits
3 : Exit
Enter the number of a function you want :
```

- **<u>Test Cases:</u>**

  We have used Junit testing to test our classes and functions.

  - Part 1:

```
hussainmansour *
class BitOperationsTest {

    hussainmansour *
    @Test
    // Test getBit FunctionA
    void getBit1() {

        BitOperations operation = new BitOperations();
        int result = operation.getBit( number: 8, position: 100);
        Assertions.assertEquals(result, actual: 0);

    }

    hussainmansour *
    @Test
    // Test getBit Function
    void getBit2() {

        BitOperations operation = new BitOperations();
        int result = operation.getBit( number: 31, position: 4);
        Assertions.assertEquals(result, actual: 1);

    }
```

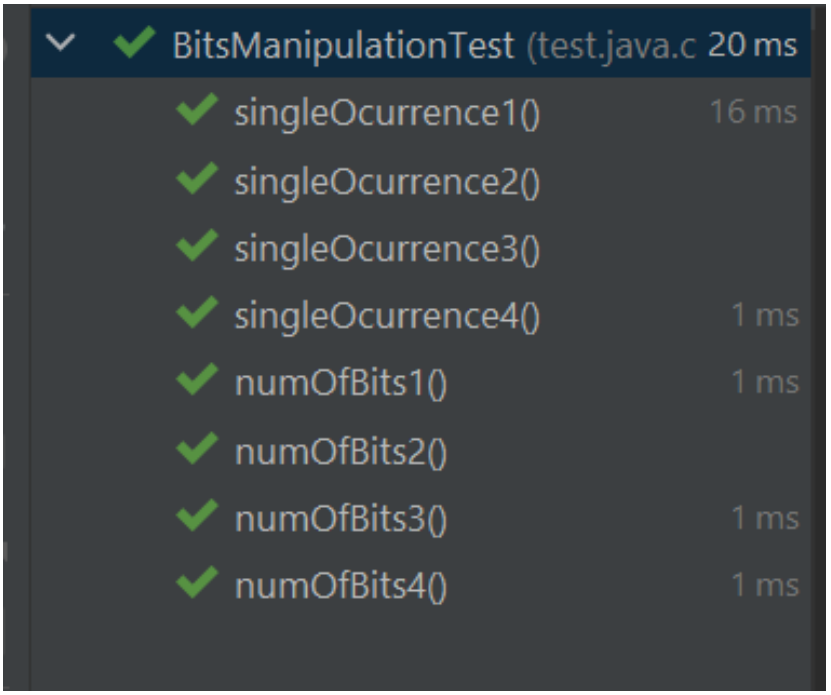


- Part 2:

```java
void UnionTest_1() {

    BitOperations op = new BitOperations();
    String[] universe = {"1", "2", "3", "4", "5","6"};
    String[] s1 = {"2","5","6"};
    String[] s2 = {"1","4","5","6"};
    int set1 = Menu.hashing_set(universe,s1) , set2 = Menu.hashing_set(universe,s2);
    int res = op.Union(set1,set2);
    ArrayList<String> arr = Main.convert(res,universe);
    Assertions.assertEquals(arr, new ArrayList<String>(Arrays.asList("1", "2", "4", "5", "6")));

}

hussainmansour
@Test
void UnionTest_2() {

    BitOperations op = new BitOperations();
    String[] universe = {"Amr", "Belal", "Hussien", "Kareem", "Mohamed"};
    String[] s1 = {"Amr","Hussien","Mohamed"};
    String[] s2 = {"Belal","Kareem"};
    int set1 = Menu.hashing_set(universe,s1) , set2 = Menu.hashing_set(universe,s2);
    int res = op.Union(set1,set2);
    ArrayList<String> arr = Main.convert(res,universe);
    Assertions.assertEquals(arr, new ArrayList<String>(Arrays.asList("Amr", "Belal", "Hussien", "Kareem", "Mohamed")));
```

- Part 3:

```java
AmrAhmed119
class BitsManipulationTest {

    AmrAhmed119
    @Test
    void singleOcurrence1() {

        BitsManipulation op = new BitsManipulation();
        int[] arr = {5, 7, 9 ,10 , 11, 9, 7, 5, 10};
        int result = op.singleOcurrence(arr);
        Assertions.assertEquals(result, actual: 11);

    }

    AmrAhmed119
    @Test
    void singleOcurrence2() {

        BitsManipulation op = new BitsManipulation();
        int[] arr = {5, 7, 9 ,10 , 11, 9, 7, 5, 10};
        int result = op.singleOcurrence(arr);
        Assertions.assertEquals(result, actual: 11);

    }
```



6- Assumptions:

- We have assumed that the universe set will not exceed 31 elements so we can represent each set with a number.

- We have assumed the correctness of the user input.

7- Links:

Repo link: https://github.com/AmrAhmed119/Ser-Operations