IT 314 – Software Engineering

Lab 7

Akshat Jindal | 202201299

Quadratic Probing Code

Program Inspection

Question 1. How many errors are there in the program? Mention the errors you have identified.

The error in the code was only due to the logical error in the insert, remove and get statements.

Question 2. Which category of program inspection would you find more effective?

For this question, the Category D of the program inspection, the Comparison errors were the most effective, as the failure of the code was due to the use of logic.

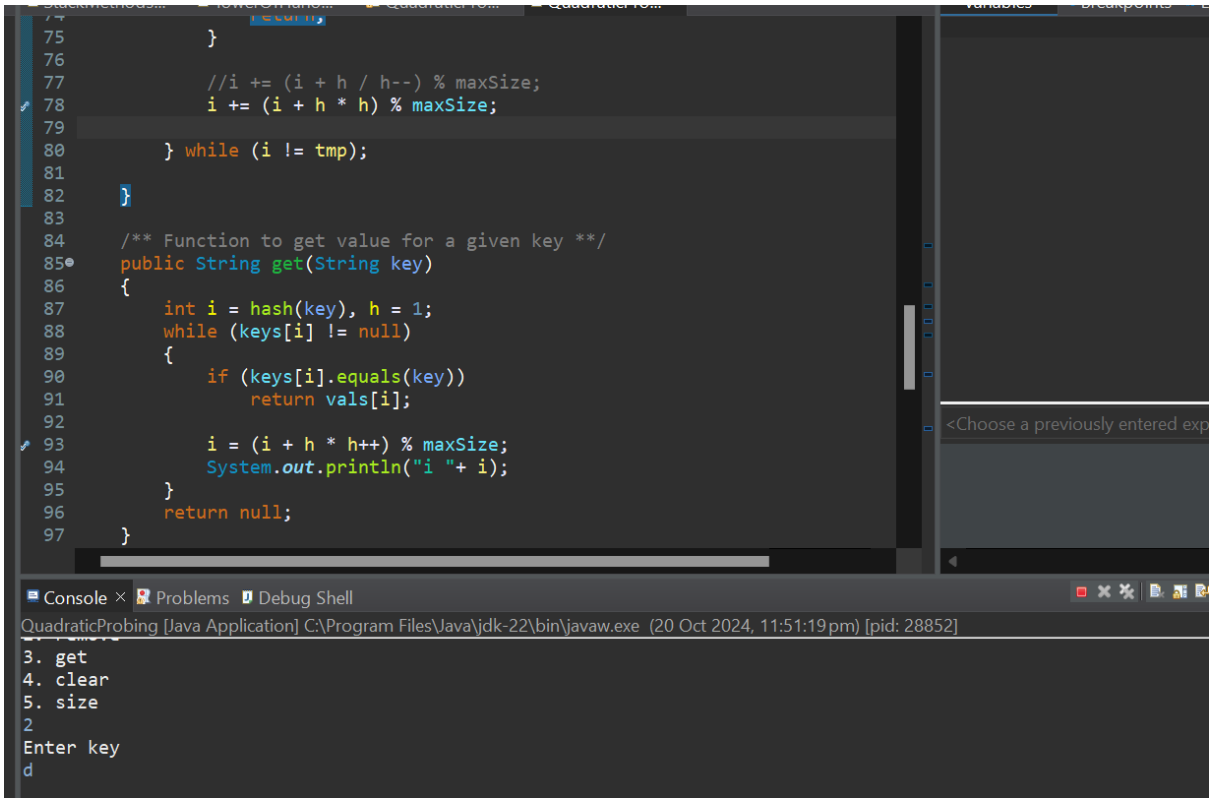
Question 3. Which type of error you are not able to identified using the program inspection?

In this example, all the errors in the document were identifiable using the program inspection method.

Question 4. Is the program inspection technique is worth applicable?

Due to the long length, it was very difficult to solve this problem using the problem inspection method.

Code Debugging



Question 1. How many errors are there in the program? Mention the errors you have identified.

There were 3 errors in the code, which were mostly based on the logical implementations of the inset, get and remove statements. This can be resolved using counter counters in the h variable.

Question 2. How many breakpoints you need to fix those errors? What are the steps you have taken to fix the error you identified in the code fragment?

The errors can be fixed by adding the debugging points on each and every function call in the hash table class.

Question 3. Submit your complete executable code.

```java
package DebugQuadraticProbing;
import java.util.Scanner;

public class QuadraticProbing {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner scan = new Scanner(System.in);
        System.out.println("Hash Table Test\n\n");
        System.out.println("Enter size");

        QuadraticProbingHashTable qpht = new QuadraticProbingHashTable(scan.nextInt() ); /** maxSizeake object of Quadrat



        char ch;

        /**  Perform QuadraticProbingHashTable operations  **/
        do{
            System.out.println("\nHash Table Operations\n");
            System.out.println("1. insert ");
            System.out.println("2. remove");
            System.out.println("3. get");
            System.out.println("4. clear");
            System.out.println("5. size");

            int choice = scan.nextInt();

            switch(choice)
            {
                case 1 :
                    System.out.println("Enter key and value");
                    qpht.insert(scan.next(), scan.next() );
                    break;

                case 2 :
                    System.out.println("Enter key");
                    qpht.remove( scan.next() );
                    break;

                case 3 :
                    System.out.println("Enter key");
                    System.out.println("Value = "+ qpht.get( scan.next() ));
                    break;
```

```java
                case 4 :
                    qpht.makeEmpty();
                    System.out.println("Hash Table Cleared\n");
                    break;

                case 5 :
                    System.out.println("Size = "+ qpht.getSize() );
                    break;

                default :
                    System.out.println("Wrong Entry \n ");
                    break;

            }

            qpht.printHashTable();
            System.out.println("\nDo you want to continue (Type y or n) \n");
            ch = scan.next().charAt(0);
        }while (ch == 'Y'|| ch == 'y');
    }
}
```

Console × Problems Debug Shell

```
<terminated> QuadraticProbing [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (20 Oct 2024, 11:44:43 pm – 11:45:13 pm) [pid: 32196]
5. size
1
Enter key and value
d desktop

Hash Table:
d desktop
c computer
```

MatrixMultip... SortingArray... × StackImplem... StackMethods... TowerOfHano... QuadraticPro... QuadraticPro... ×

```java
package DebugQuadraticProbing;

public class QuadraticProbingHashTable {

    private int currentSize, maxSize;
    private String[] keys;
    private String[] vals;

    /** Constructor **/
    public QuadraticProbingHashTable(int capacity)
    {
        currentSize = 0;
        maxSize = capacity;
        keys = new String[maxSize];
        vals = new String[maxSize];
    }

    /** Function to clear hash table **/
    public void makeEmpty()
    {
        currentSize = 0;
        keys = new String[maxSize];
        vals = new String[maxSize];
    }

    /** Function to get size of hash table **/
    public int getSize()
    {
        return currentSize;
    }

    /** Function to check if hash table is full **/
    public boolean isFull()
    {
        return currentSize == maxSize;
    }

    /** Function to check if hash table is empty **/
    public boolean isEmpty()
    {
        return getSize() == 0;
    }

    /** Function to check if hash table contains a key **/
    public boolean contains(String key)
    {
        return get(key) != null;
```

```java
    /** Function to check if hash table contains a key **/
    public boolean contains(String key)
    {
        return get(key) !=  null;
    }

    /** Function to get hash code of a given key **/
    private int hash(String key)
    {
        return key.hashCode() % maxSize;
    }

    /** Function to insert key-value pair **/
    public void insert(String key, String val)
    {
        int tmp = hash(key);
        int i = tmp, h = 1;
        do
        {
            if (keys[i] == null)
            {
                keys[i] = key;
                vals[i] = val;
```

```java
        /** find position key and delete **/
        int i = hash(key), h = 1;
        while (!key.equals(keys[i]))
            i = (i + h * h++) % maxSize;

        keys[i] = vals[i] = null;

        /** rehash all keys **/
        for (i = (i + h * h++) % maxSize; keys[i] != null; i = (i + h * h++) % maxSize)
        {
            String tmp1 = keys[i], tmp2 = vals[i];
            keys[i] = vals[i] = null;
            currentSize--;
            insert(tmp1, tmp2);
        }
        currentSize--;
    }

    /** Function to print HashTable **/
    public void printHashTable()
```

```java
            insert(tmp1, tmp2);
        }
        currentSize--;
    }

    /** Function to print HashTable **/
    public void printHashTable()
    {
        System.out.println("\nHash Table: ");
        for (int i = 0; i < maxSize; i++)
            if (keys[i] != null)
                System.out.println(keys[i] +" "+ vals[i]);

        System.out.println();
    }
}
```