Tugas Kecil 1 IF2211 Strategi Algoritma

Penyelesaian Permainan Kartu 24 dengan Algoritma Brute Force

Semester II tahun 2022/2023

Saddam Annais Shaquille
13521121
Teknik Informatika 2021

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

I. Algoritma *Brute Force* pada Penyelesaian Permainan Kartu 24

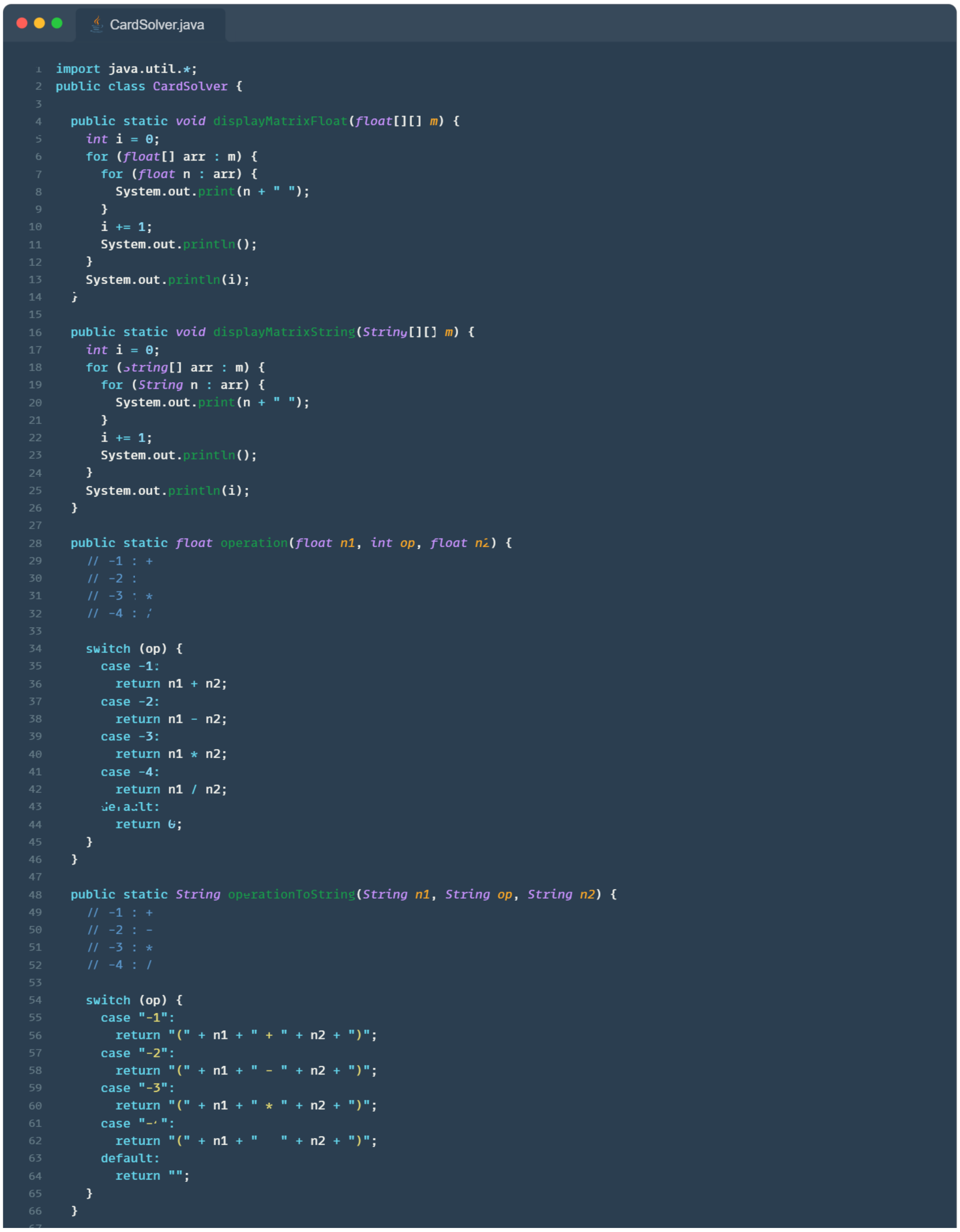
Permainan kartu 24 adalah permainan menggunakan kartu untuk memecahkan teka-teki matematika. Teka teki ini menggunakan 4 angka acak, biasanya direpresentasi menggunakan 4 kartu, yang kemudian dioperasikan menggunakan penjumlahan, pengurangan, perkalian, atau pembagian sampai terbentuk hasil akhir yaitu 24. Permainan ini dapat dikomputasikan menggunakan algoritma *brute force*, yaitu dengan cara mencoba setiap kemungkinan kombinasi angka dan operasi sampai solusi ditemukan.

1. Pertama-tama, akan dibuat array 2 dimensi yang setiap baris elemennya akan berisi kombinasi dari 4 angka yang diambil dari kartu yang digunakan dalam permainan. Hal ini dilakukan untuk mencari setiap kombinasi yang mungkin dari 4 angka tersebut sehingga semua kemungkinan solusi dapat dihitung. Jumlah kombinasi yang akan terbuat adalah 4! atau 24 buah.
2. Kedua, setelah membuat array 2 dimensi yang berisi kombinasi dari 4 angka, akan dibuat lagi sebuah array 2 dimensi lain yang setiap elemen barisnya akan berisi kombinasi dari 3 buah operasi matematika. Operasi-operasi tersebut terdiri dari penjumlahan (+), pengurangan (-), perkalian (*), atau pembagian (/). Hal ini dilakukan dengan tujuan yang sama seperti langkah pertama, yaitu untuk mencari setiap kombinasi yang mungkin dari 3 operasi tersebut, sehingga semua kemungkinan solusi dapat dihitung. Jumlah kombinasi yang akan terbuat adalah 4^3 atau 64 buah karena setiap operasi memiliki 4 pilihan dan setiap pilihan berbeda akan menghasilkan kombinasi yang berbeda pula.
3. Ketiga, setelah dibuat array 2 dimensi yang berisi kombinasi dari 4 angka dan kombinasi dari 3 operasi matematika, urutan evaluasi ekspresi matematika harus diperharikan. Urutan evaluasi ekspresi ditentukan oleh penggunaan tanda kurung "(" dan ")", yang akan mempengaruhi hasil perhitungan. Ada 5 kombinasi urutan evaluasi ekspresi yang mungkin pada 4 buah angka, yaitu:
   a. ((X o X) o X) o X
   b. (X o X) o (X o X)
   c. (X o (X o X)) o X
   d. X o ((X o X) o X)
   e. X o (X o (X o X))

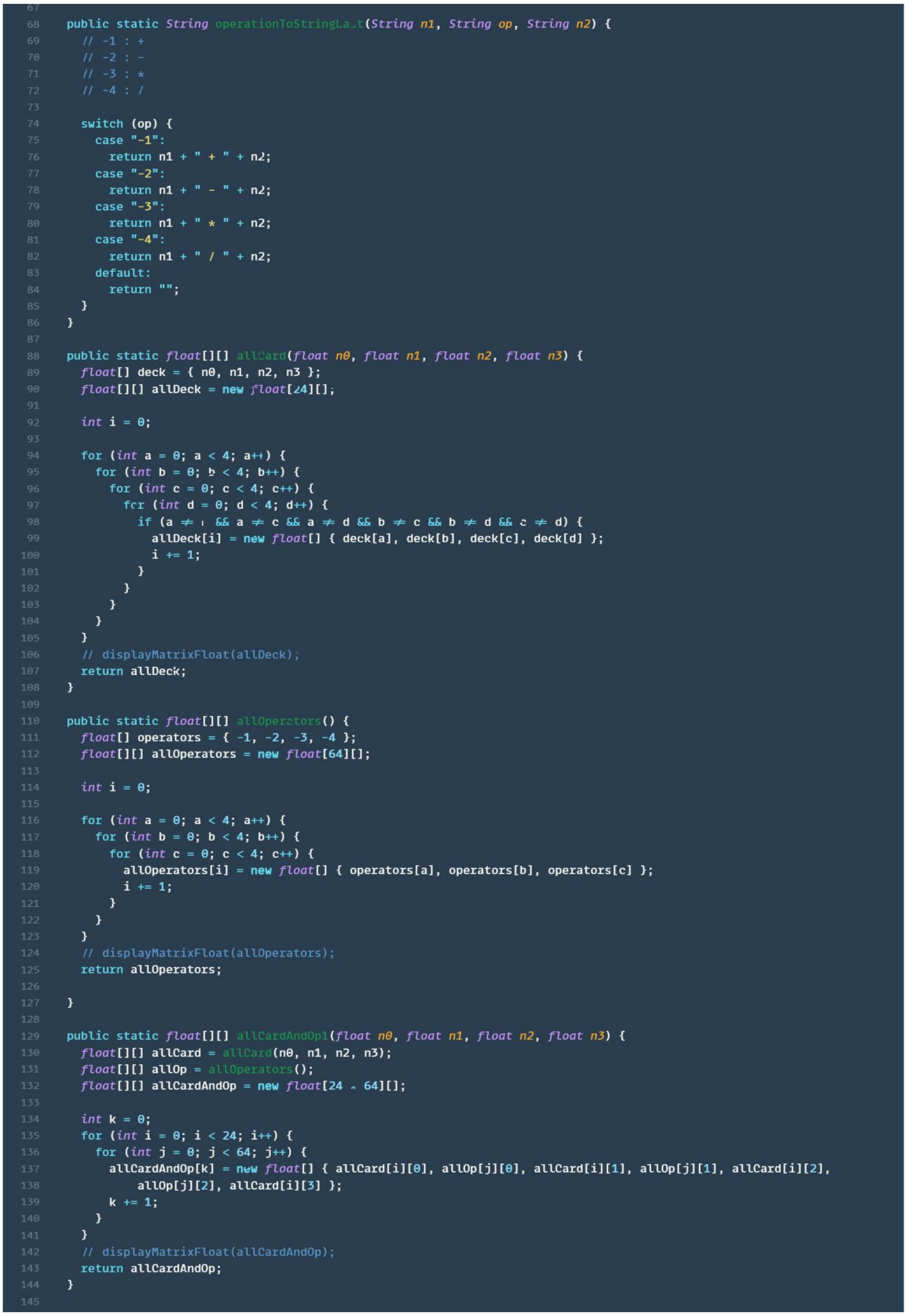
   Misal X adalah angka dan o adalah operasi
4. Selanjutnya, tiap baris array yang berisi kombinasi angka akan dioperasikan dengan semua baris array operasi sehingga akan terdapat 24 * 64 buah ekspresi. Tiap ekspresi tersebut memiliki 5 kombinasi bentuk urutan evaluasi ekspresi sehingga jumlah ekspresi yang diperoleh adalah 24 * 64 * 5 atau 7680 hasil ekspresi yang berbeda.
5. Setelah itu, tiap kombinasi ekspresi akan dievaluasi. Jika hasil perhitungan ekspresi tersebut sama dengan 24, solusi untuk teka-teki ini ditemukan.

II. Source Program dalam Bahasa Java

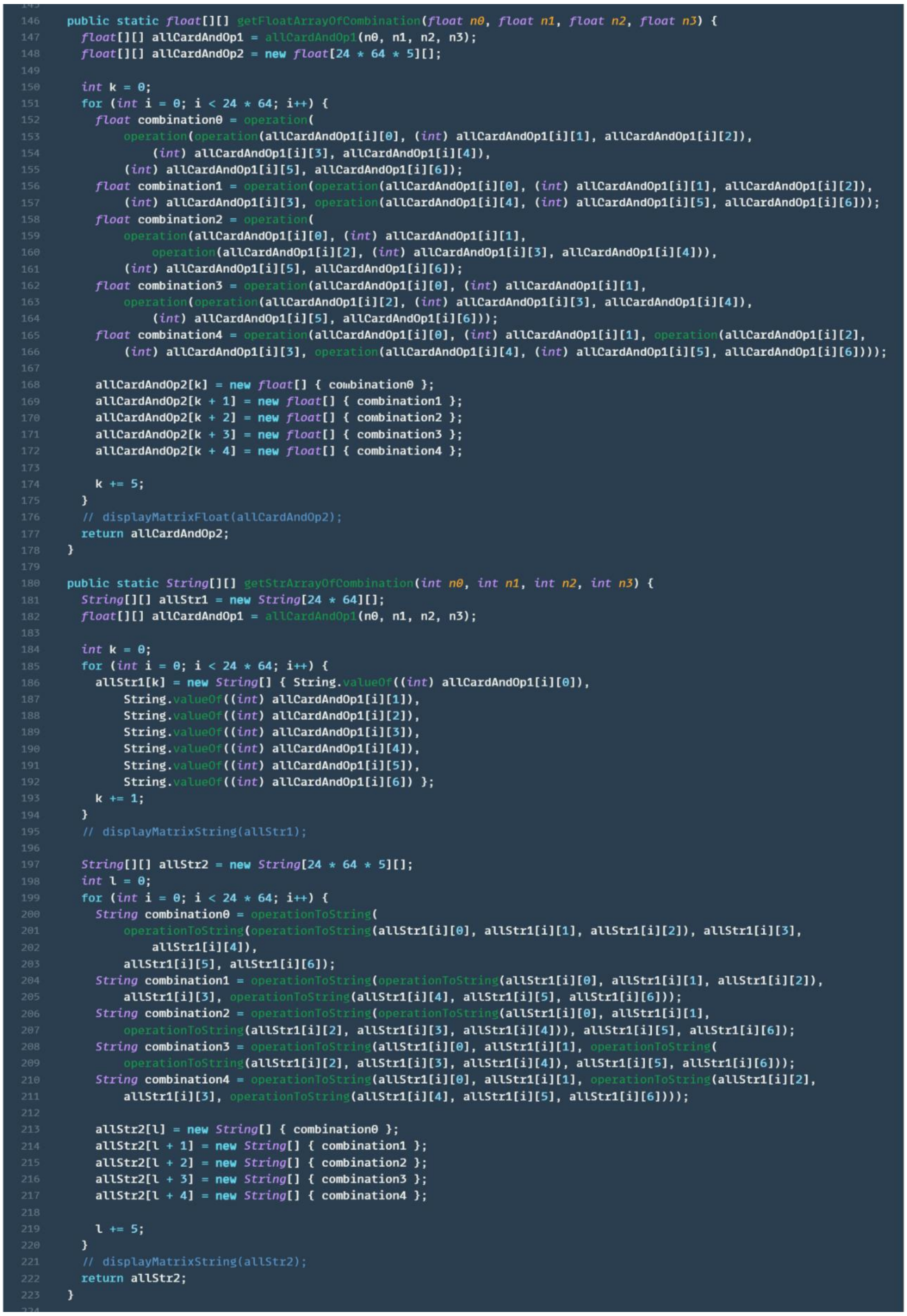
1. CardSolver.java



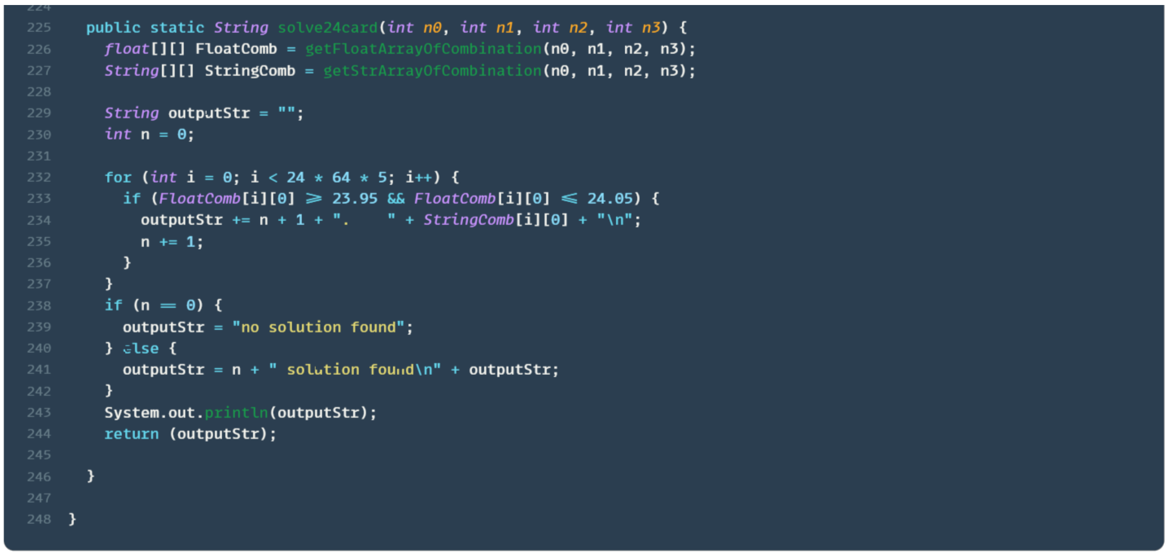
```java
import java.util.*;
public class CardSolver {

  public static void displayMatrixFloat(float[][] m) {
    int i = 0;
    for (float[] arr : m) {
      for (float n : arr) {
        System.out.print(n + " ");
      }
      i += 1;
      System.out.println();
    }
    System.out.println(i);
  }

  public static void displayMatrixString(String[][] m) {
    int i = 0;
    for (String[] arr : m) {
      for (String n : arr) {
        System.out.print(n + " ");
      }
      i += 1;
      System.out.println();
    }
    System.out.println(i);
  }

  public static float operation(float n1, int op, float n2) {
    // -1 : +
    // -2 :
    // -3 : *
    // -4 : /

    switch (op) {
      case -1:
        return n1 + n2;
      case -2:
        return n1 - n2;
      case -3:
        return n1 * n2;
      case -4:
        return n1 / n2;
      default:
        return 0;
    }
  }

  public static String operationToString(String n1, String op, String n2) {
    // -1 : +
    // -2 : -
    // -3 : *
    // -4 : /

    switch (op) {
      case "-1":
        return "(" + n1 + " + " + n2 + ")";
      case "-2":
        return "(" + n1 + " - " + n2 + ")";
      case "-3":
        return "(" + n1 + " * " + n2 + ")";
      case "-4":
        return "(" + n1 + "   " + n2 + ")";
      default:
        return "";
    }
  }
```



```java
public static String operationToStringLast(String n1, String op, String n2) {
  // -1 : +
  // -2 : -
  // -3 : *
  // -4 : /

  switch (op) {
    case "-1":
      return n1 + " + " + n2;
    case "-2":
      return n1 + " - " + n2;
    case "-3":
      return n1 + " * " + n2;
    case "-4":
      return n1 + " / " + n2;
    default:
      return "";
  }
}

public static float[][] allCard(float n0, float n1, float n2, float n3) {
  float[] deck = { n0, n1, n2, n3 };
  float[][] allDeck = new float[24][];

  int i = 0;

  for (int a = 0; a < 4; a++) {
    for (int b = 0; b < 4; b++) {
      for (int c = 0; c < 4; c++) {
        for (int d = 0; d < 4; d++) {
          if (a ≠ b && a ≠ c && a ≠ d && b ≠ c && b ≠ d && c ≠ d) {
            allDeck[i] = new float[] { deck[a], deck[b], deck[c], deck[d] };
            i += 1;
          }
        }
      }
    }
  }
  // displayMatrixFloat(allDeck);
  return allDeck;
}

public static float[][] allOperators() {
  float[] operators = { -1, -2, -3, -4 };
  float[][] allOperators = new float[64][];

  int i = 0;

  for (int a = 0; a < 4; a++) {
    for (int b = 0; b < 4; b++) {
      for (int c = 0; c < 4; c++) {
        allOperators[i] = new float[] { operators[a], operators[b], operators[c] };
        i += 1;
      }
    }
  }
  // displayMatrixFloat(allOperators);
  return allOperators;

}

public static float[][] allCardAndOp1(float n0, float n1, float n2, float n3) {
  float[][] allCard = allCard(n0, n1, n2, n3);
  float[][] allOp = allOperators();
  float[][] allCardAndOp = new float[24 * 64][];

  int k = 0;
  for (int i = 0; i < 24; i++) {
    for (int j = 0; j < 64; j++) {
      allCardAndOp[k] = new float[] { allCard[i][0], allOp[j][0], allCard[i][1], allOp[j][1], allCard[i][2],
          allOp[j][2], allCard[i][3] };
      k += 1;
    }
  }
  // displayMatrixFloat(allCardAndOp);
  return allCardAndOp;
}
```

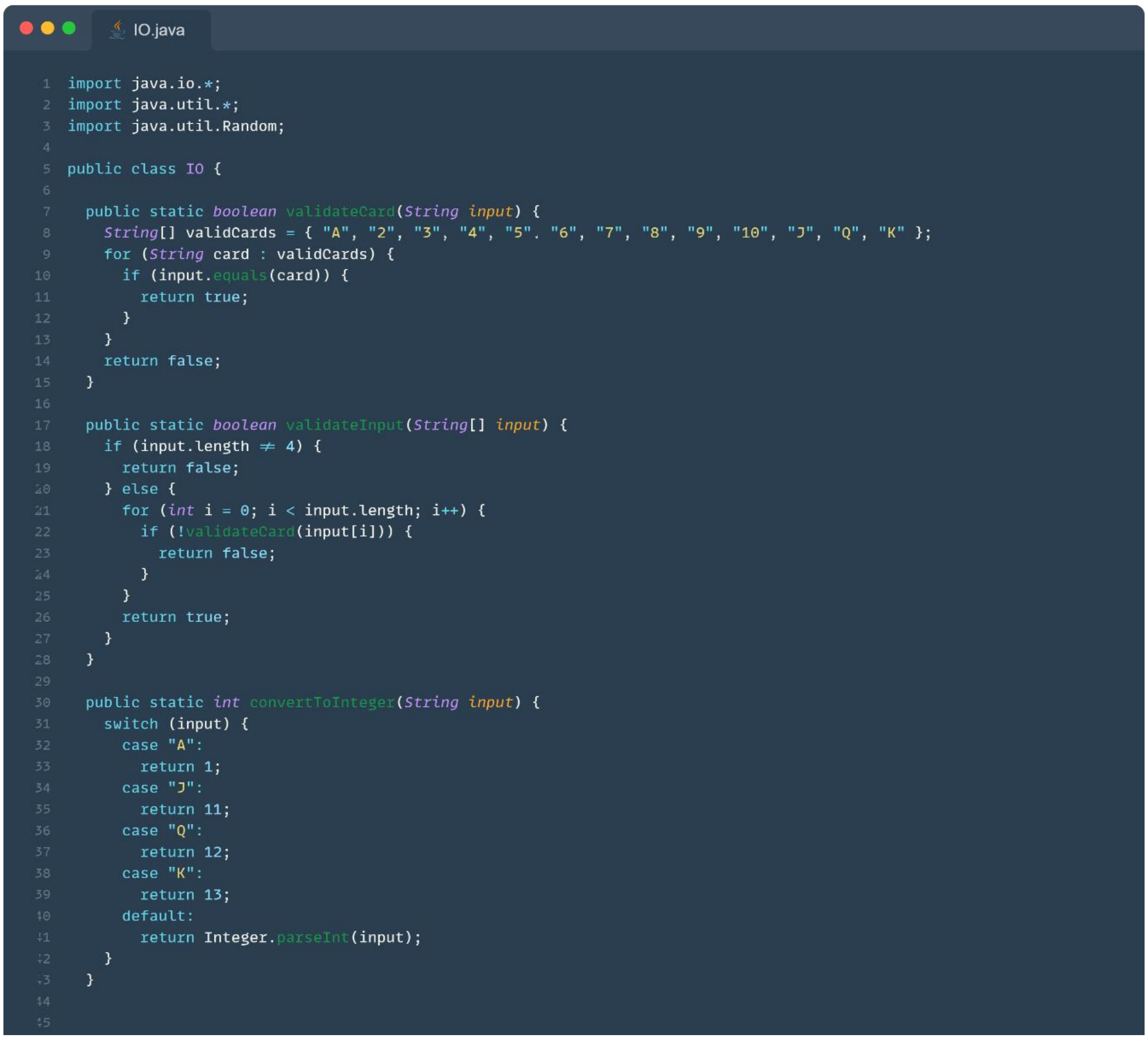


```java
public static float[][] getFloatArrayOfCombination(float n0, float n1, float n2, float n3) {
  float[][] allCardAndOp1 = allCardAndOp1(n0, n1, n2, n3);
  float[][] allCardAndOp2 = new float[24 * 64 * 5][];

  int k = 0;
  for (int i = 0; i < 24 * 64; i++) {
    float combination0 = operation(
        operation(operation(allCardAndOp1[i][0], (int) allCardAndOp1[i][1], allCardAndOp1[i][2]),
            (int) allCardAndOp1[i][3], allCardAndOp1[i][4]),
        (int) allCardAndOp1[i][5], allCardAndOp1[i][6]);
    float combination1 = operation(operation(allCardAndOp1[i][0], (int) allCardAndOp1[i][1], allCardAndOp1[i][2]),
        (int) allCardAndOp1[i][3], operation(allCardAndOp1[i][4], (int) allCardAndOp1[i][5], allCardAndOp1[i][6]));
    float combination2 = operation(
        operation(allCardAndOp1[i][0], (int) allCardAndOp1[i][1],
            operation(allCardAndOp1[i][2], (int) allCardAndOp1[i][3], allCardAndOp1[i][4])),
        (int) allCardAndOp1[i][5], allCardAndOp1[i][6]);
    float combination3 = operation(allCardAndOp1[i][0], (int) allCardAndOp1[i][1],
        operation(operation(allCardAndOp1[i][2], (int) allCardAndOp1[i][3], allCardAndOp1[i][4]),
            (int) allCardAndOp1[i][5], allCardAndOp1[i][6]));
    float combination4 = operation(allCardAndOp1[i][0], (int) allCardAndOp1[i][1], operation(allCardAndOp1[i][2],
        (int) allCardAndOp1[i][3], operation(allCardAndOp1[i][4], (int) allCardAndOp1[i][5], allCardAndOp1[i][6])));

    allCardAndOp2[k] = new float[] { combination0 };
    allCardAndOp2[k + 1] = new float[] { combination1 };
    allCardAndOp2[k + 2] = new float[] { combination2 };
    allCardAndOp2[k + 3] = new float[] { combination3 };
    allCardAndOp2[k + 4] = new float[] { combination4 };

    k += 5;
  }
  // displayMatrixFloat(allCardAndOp2);
  return allCardAndOp2;
}

public static String[][] getStrArrayOfCombination(int n0, int n1, int n2, int n3) {
  String[][] allStr1 = new String[24 * 64][];
  float[][] allCardAndOp1 = allCardAndOp1(n0, n1, n2, n3);

  int k = 0;
  for (int i = 0; i < 24 * 64; i++) {
    allStr1[k] = new String[] { String.valueOf((int) allCardAndOp1[i][0]),
        String.valueOf((int) allCardAndOp1[i][1]),
        String.valueOf((int) allCardAndOp1[i][2]),
        String.valueOf((int) allCardAndOp1[i][3]),
        String.valueOf((int) allCardAndOp1[i][4]),
        String.valueOf((int) allCardAndOp1[i][5]),
        String.valueOf((int) allCardAndOp1[i][6]) };
    k += 1;
  }
  // displayMatrixString(allStr1);

  String[][] allStr2 = new String[24 * 64 * 5][];
  int l = 0;
  for (int i = 0; i < 24 * 64; i++) {
    String combination0 = operationToString(
        operationToString(operationToString(allStr1[i][0], allStr1[i][1], allStr1[i][2]), allStr1[i][3],
            allStr1[i][4]),
        allStr1[i][5], allStr1[i][6]);
    String combination1 = operationToString(operationToString(allStr1[i][0], allStr1[i][1], allStr1[i][2]),
        allStr1[i][3], operationToString(allStr1[i][4], allStr1[i][5], allStr1[i][6]));
    String combination2 = operationToString(operationToString(allStr1[i][0], allStr1[i][1],
        operationToString(allStr1[i][2], allStr1[i][3], allStr1[i][4])), allStr1[i][5], allStr1[i][6]);
    String combination3 = operationToString(allStr1[i][0], allStr1[i][1], operationToString(
        operationToString(allStr1[i][2], allStr1[i][3], allStr1[i][4]), allStr1[i][5], allStr1[i][6]));
    String combination4 = operationToString(allStr1[i][0], allStr1[i][1], operationToString(allStr1[i][2],
        allStr1[i][3], operationToString(allStr1[i][4], allStr1[i][5], allStr1[i][6])));

    allStr2[l] = new String[] { combination0 };
    allStr2[l + 1] = new String[] { combination1 };
    allStr2[l + 2] = new String[] { combination2 };
    allStr2[l + 3] = new String[] { combination3 };
    allStr2[l + 4] = new String[] { combination4 };

    l += 5;
  }
  // displayMatrixString(allStr2);
  return allStr2;
}
```



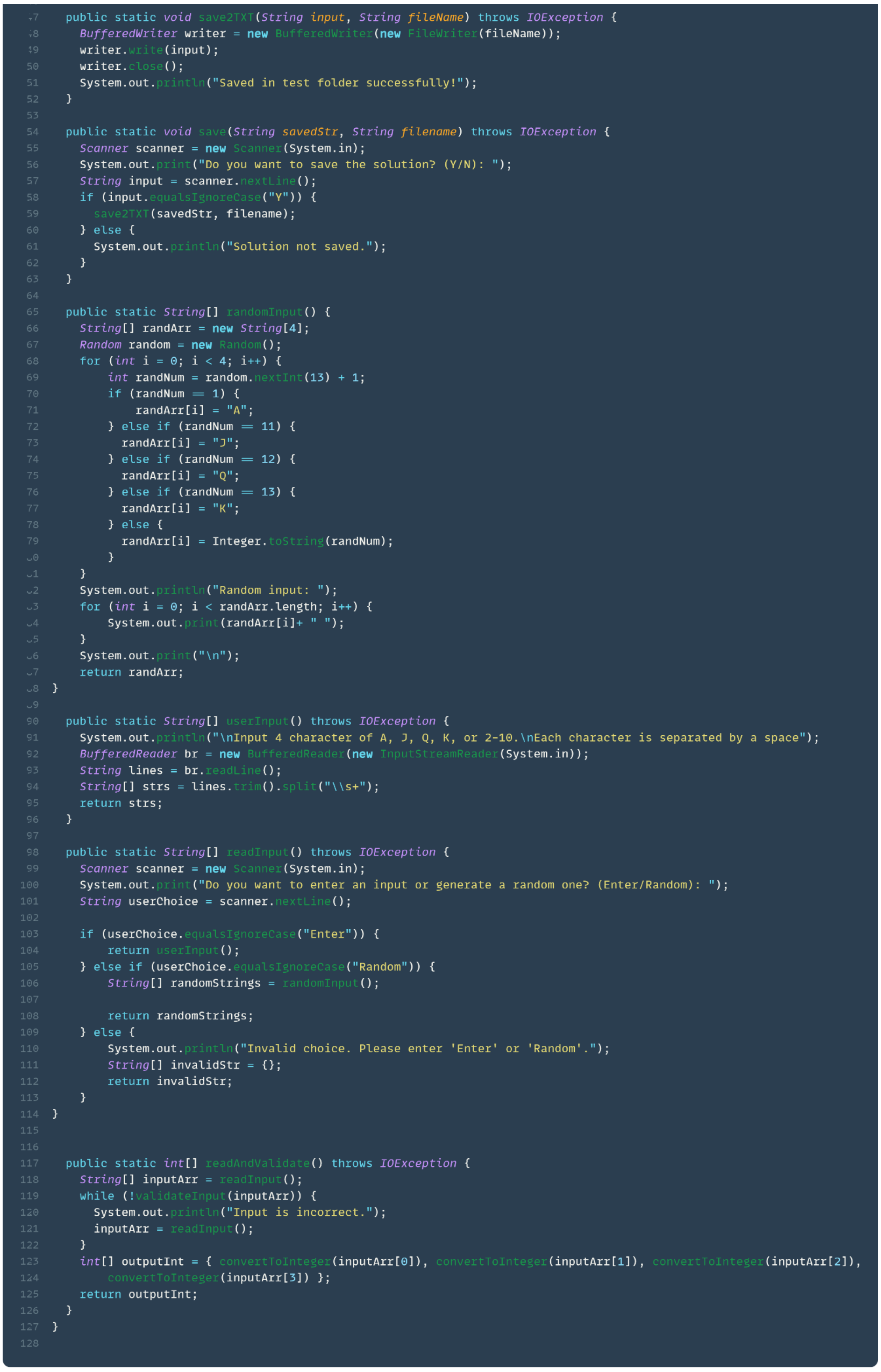
```java
  public static String solve24card(int n0, int n1, int n2, int n3) {
    float[][] FloatComb = getFloatArrayOfCombination(n0, n1, n2, n3);
    String[][] StringComb = getStrArrayOfCombination(n0, n1, n2, n3);

    String outputStr = "";
    int n = 0;

    for (int i = 0; i < 24 * 64 * 5; i++) {
      if (FloatComb[i][0] >= 23.95 && FloatComb[i][0] <= 24.05) {
        outputStr += n + 1 + ".    " + StringComb[i][0] + "\n";
        n += 1;
      }
    }
    if (n == 0) {
      outputStr = "no solution found";
    } else {
      outputStr = n + " solution found\n" + outputStr;
    }
    System.out.println(outputStr);
    return (outputStr);

  }

}
```

2. IO.java



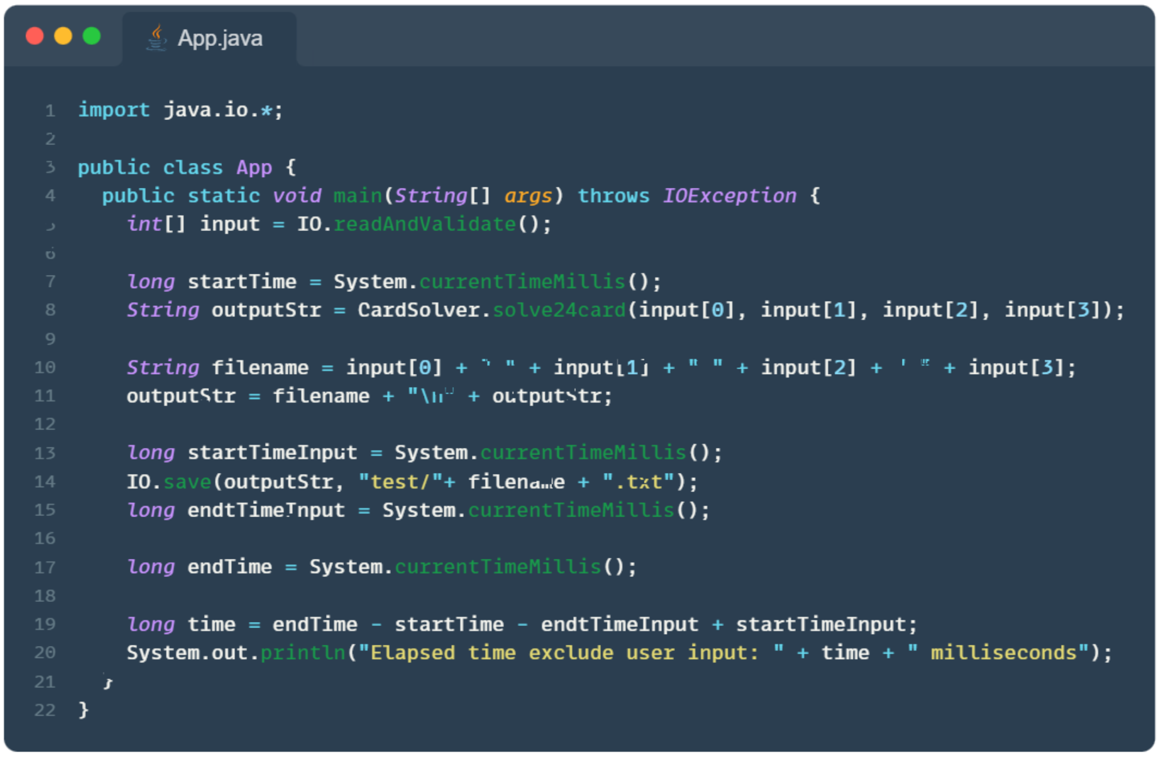
IO.java

```java
import java.io.*;
import java.util.*;
import java.util.Random;

public class IO {

  public static boolean validateCard(String input) {
    String[] validCards = { "A", "2", "3", "4", "5". "6", "7", "8", "9", "10", "J", "Q", "K" };
    for (String card : validCards) {
      if (input.equals(card)) {
        return true;
      }
    }
    return false;
  }

  public static boolean validateInput(String[] input) {
    if (input.length != 4) {
      return false;
    } else {
      for (int i = 0; i < input.length; i++) {
        if (!validateCard(input[i])) {
          return false;
        }
      }
      return true;
    }
  }

  public static int convertToInteger(String input) {
    switch (input) {
      case "A":
        return 1;
      case "J":
        return 11;
      case "Q":
        return 12;
      case "K":
        return 13;
      default:
        return Integer.parseInt(input);
    }
  }
```



```java
  public static void save2TXT(String input, String fileName) throws IOException {
    BufferedWriter writer = new BufferedWriter(new FileWriter(fileName));
    writer.write(input);
    writer.close();
    System.out.println("Saved in test folder successfully!");
  }

  public static void save(String savedStr, String filename) throws IOException {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Do you want to save the solution? (Y/N): ");
    String input = scanner.nextLine();
    if (input.equalsIgnoreCase("Y")) {
      save2TXT(savedStr, filename);
    } else {
      System.out.println("Solution not saved.");
    }
  }

  public static String[] randomInput() {
    String[] randArr = new String[4];
    Random random = new Random();
    for (int i = 0; i < 4; i++) {
        int randNum = random.nextInt(13) + 1;
        if (randNum == 1) {
            randArr[i] = "A";
        } else if (randNum == 11) {
          randArr[i] = "J";
        } else if (randNum == 12) {
          randArr[i] = "Q";
        } else if (randNum == 13) {
          randArr[i] = "K";
        } else {
          randArr[i] = Integer.toString(randNum);
        }
    }
    System.out.println("Random input: ");
    for (int i = 0; i < randArr.length; i++) {
        System.out.print(randArr[i]+ " ");
    }
    System.out.print("\n");
    return randArr;
}

  public static String[] userInput() throws IOException {
    System.out.println("\nInput 4 character of A, J, Q, K, or 2-10.\nEach character is separated by a space");
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    String lines = br.readLine();
    String[] strs = lines.trim().split("\\s+");
    return strs;
  }

  public static String[] readInput() throws IOException {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Do you want to enter an input or generate a random one? (Enter/Random): ");
    String userChoice = scanner.nextLine();

    if (userChoice.equalsIgnoreCase("Enter")) {
        return userInput();
    } else if (userChoice.equalsIgnoreCase("Random")) {
        String[] randomStrings = randomInput();

        return randomStrings;
    } else {
        System.out.println("Invalid choice. Please enter 'Enter' or 'Random'.");
        String[] invalidStr = {};
        return invalidStr;
    }
}


  public static int[] readAndValidate() throws IOException {
    String[] inputArr = readInput();
    while (!validateInput(inputArr)) {
      System.out.println("Input is incorrect.");
      inputArr = readInput();
    }
    int[] outputInt = { convertToInteger(inputArr[0]), convertToInteger(inputArr[1]), convertToInteger(inputArr[2]),
        convertToInteger(inputArr[3]) };
    return outputInt;
  }
}
```

3. App.java

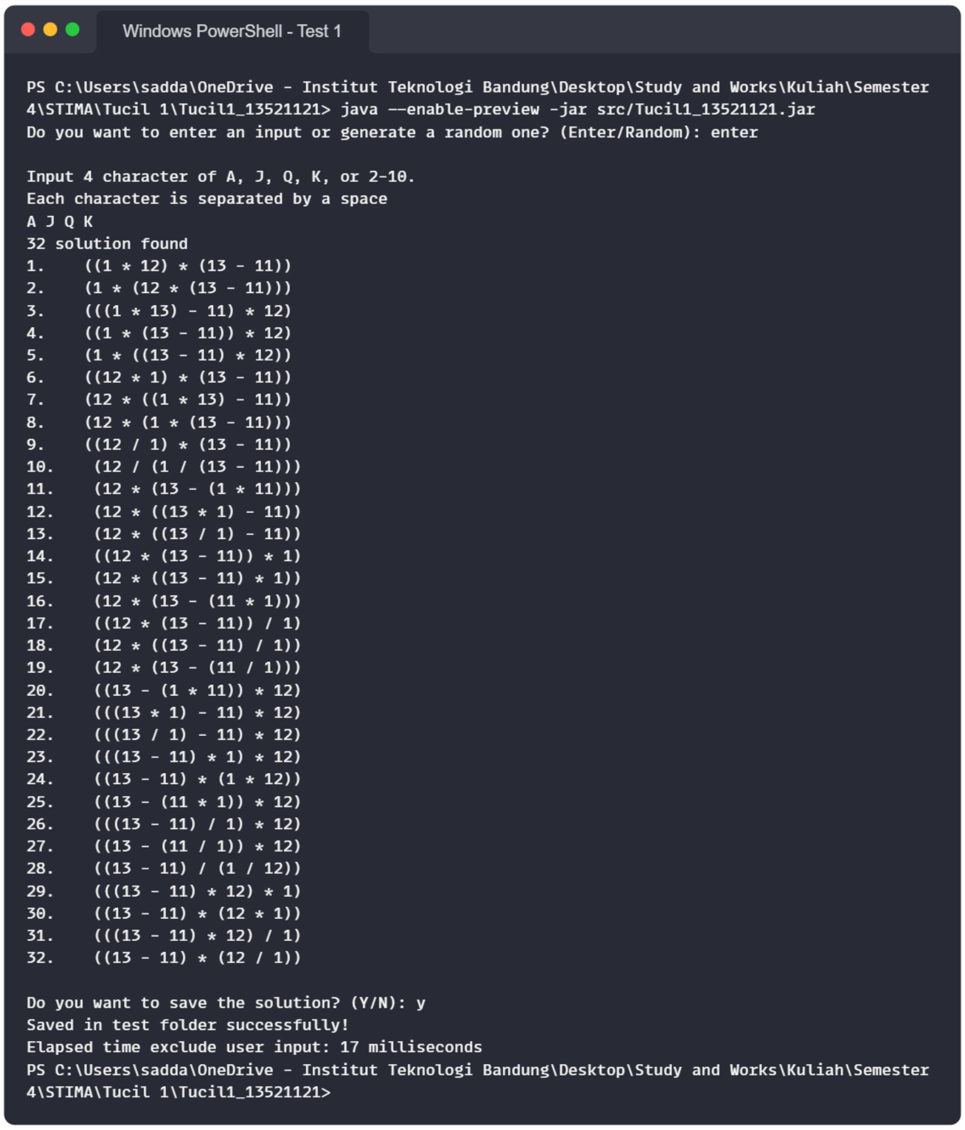


App.java

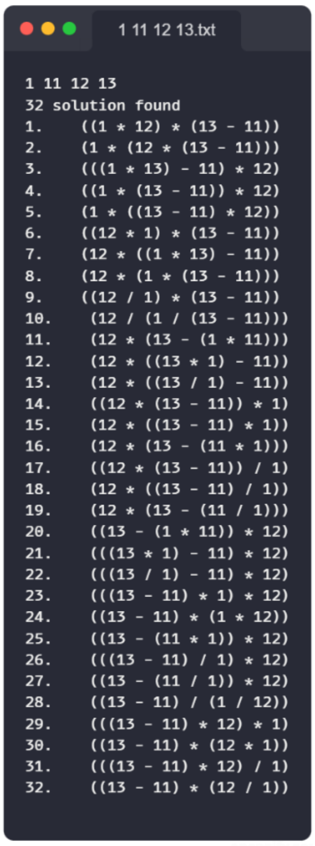
```java
import java.io.*;

public class App {
  public static void main(String[] args) throws IOException {
    int[] input = IO.readAndValidate();

    long startTime = System.currentTimeMillis();
    String outputStr = CardSolver.solve24card(input[0], input[1], input[2], input[3]);

    String filename = input[0] + " " + input[1] + " " + input[2] + " " + input[3];
    outputStr = filename + "\n" + outputStr;

    long startTimeInput = System.currentTimeMillis();
    IO.save(outputStr, "test/"+ filename + ".txt");
    long endtTimeInput = System.currentTimeMillis();

    long endTime = System.currentTimeMillis();

    long time = endTime - startTime - endtTimeInput + startTimeInput;
    System.out.println("Elapsed time exclude user input: " + time + " milliseconds");
  }
}
```

III. *Screenshot* Pengujian Input dan Output

1. Test 1

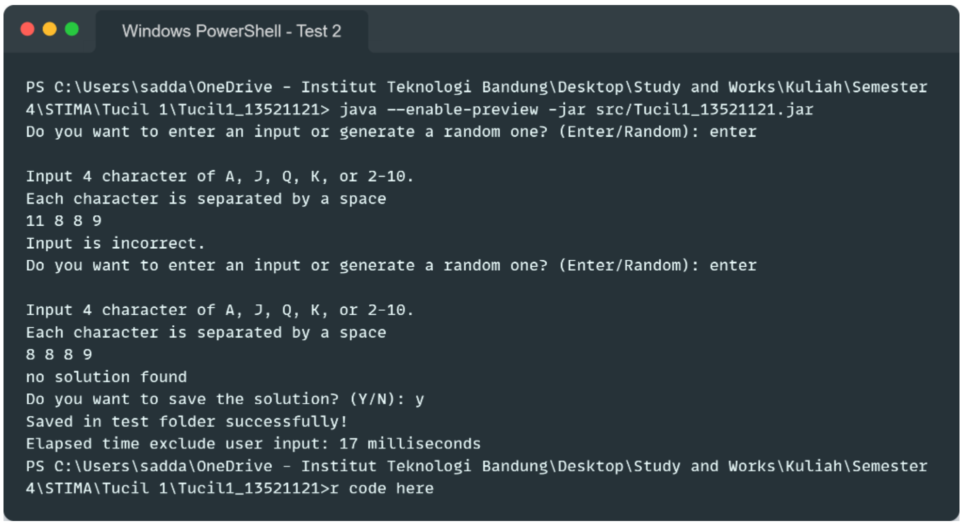

Windows PowerShell - Test 1

```
PS C:\Users\sadda\OneDrive - Institut Teknologi Bandung\Desktop\Study and Works\Kuliah\Semester
4\STIMA\Tucil 1\Tucil1_13521121> java --enable-preview -jar src/Tucil1_13521121.jar
Do you want to enter an input or generate a random one? (Enter/Random): enter

Input 4 character of A, J, Q, K, or 2-10.
Each character is separated by a space
A J Q K
32 solution found
1.    ((1 * 12) * (13 - 11))
2.    (1 * (12 * (13 - 11)))
3.    (((1 * 13) - 11) * 12)
4.    ((1 * (13 - 11)) * 12)
5.    (1 * ((13 - 11) * 12))
6.    ((12 * 1) * (13 - 11))
7.    (12 * ((1 * 13) - 11))
8.    (12 * (1 * (13 - 11)))
9.    ((12 / 1) * (13 - 11))
10.    (12 / (1 / (13 - 11)))
11.    (12 * (13 - (1 * 11)))
12.    (12 * ((13 * 1) - 11))
13.    (12 * ((13 / 1) - 11))
14.    ((12 * (13 - 11)) * 1)
15.    (12 * ((13 - 11) * 1))
16.    (12 * (13 - (11 * 1)))
17.    ((12 * (13 - 11)) / 1)
18.    (12 * ((13 - 11) / 1))
19.    (12 * (13 - (11 / 1)))
20.    ((13 - (1 * 11)) * 12)
21.    (((13 * 1) - 11) * 12)
22.    (((13 / 1) - 11) * 12)
23.    (((13 - 11) * 1) * 12)
24.    ((13 - 11) * (1 * 12))
25.    ((13 - (11 * 1)) * 12)
26.    (((13 - 11) / 1) * 12)
27.    ((13 - (11 / 1)) * 12)
28.    ((13 - 11) / (1 / 12))
29.    (((13 - 11) * 12) * 1)
30.    ((13 - 11) * (12 * 1))
31.    (((13 - 11) * 12) / 1)
32.    ((13 - 11) * (12 / 1))

Do you want to save the solution? (Y/N): y
Saved in test folder successfully!
Elapsed time exclude user input: 17 milliseconds
PS C:\Users\sadda\OneDrive - Institut Teknologi Bandung\Desktop\Study and Works\Kuliah\Semester
4\STIMA\Tucil 1\Tucil1_13521121>
```

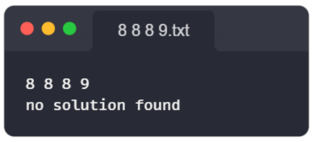

1 11 12 13.txt

```
1 11 12 13
32 solution found
1.    ((1 * 12) * (13 - 11))
2.    (1 * (12 * (13 - 11)))
3.    (((1 * 13) - 11) * 12)
4.    ((1 * (13 - 11)) * 12)
5.    (1 * ((13 - 11) * 12))
6.    ((12 * 1) * (13 - 11))
7.    (12 * ((1 * 13) - 11))
8.    (12 * (1 * (13 - 11)))
9.    ((12 / 1) * (13 - 11))
10.    (12 / (1 / (13 - 11)))
11.    (12 * (13 - (1 * 11)))
12.    (12 * ((13 * 1) - 11))
13.    (12 * ((13 / 1) - 11))
14.    ((12 * (13 - 11)) * 1)
15.    (12 * ((13 - 11) * 1))
16.    (12 * (13 - (11 * 1)))
17.    ((12 * (13 - 11)) / 1)
18.    (12 * ((13 - 11) / 1))
19.    (12 * (13 - (11 / 1)))
20.    ((13 - (1 * 11)) * 12)
21.    (((13 * 1) - 11) * 12)
22.    (((13 / 1) - 11) * 12)
23.    (((13 - 11) * 1) * 12)
24.    ((13 - 11) * (1 * 12))
25.    ((13 - (11 * 1)) * 12)
26.    (((13 - 11) / 1) * 12)
27.    ((13 - (11 / 1)) * 12)
28.    ((13 - 11) / (1 / 12))
29.    (((13 - 11) * 12) * 1)
30.    ((13 - 11) * (12 * 1))
31.    (((13 - 11) * 12) / 1)
32.    ((13 - 11) * (12 / 1))
```

2. Test 2

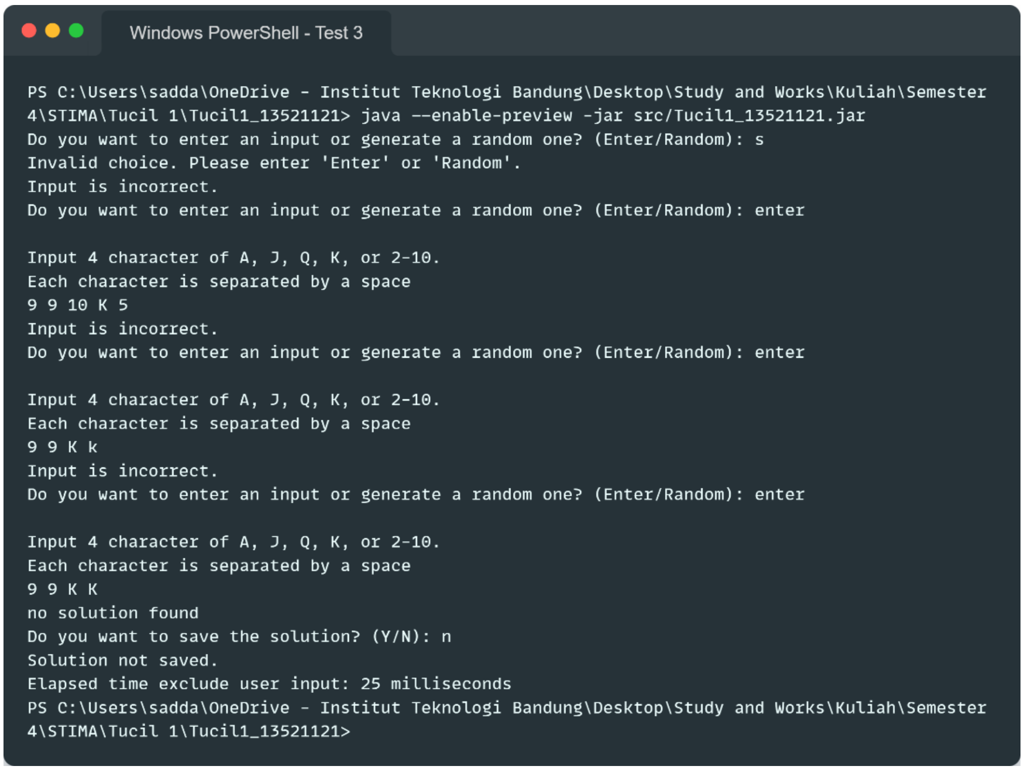

Windows PowerShell - Test 2

```
PS C:\Users\sadda\OneDrive - Institut Teknologi Bandung\Desktop\Study and Works\Kuliah\Semester
4\STIMA\Tucil 1\Tucil1_13521121> java --enable-preview -jar src/Tucil1_13521121.jar
Do you want to enter an input or generate a random one? (Enter/Random): enter

Input 4 character of A, J, Q, K, or 2-10.
Each character is separated by a space
11 8 8 9
Input is incorrect.
Do you want to enter an input or generate a random one? (Enter/Random): enter

Input 4 character of A, J, Q, K, or 2-10.
Each character is separated by a space
8 8 8 9
no solution found
Do you want to save the solution? (Y/N): y
Saved in test folder successfully!
Elapsed time exclude user input: 17 milliseconds
PS C:\Users\sadda\OneDrive - Institut Teknologi Bandung\Desktop\Study and Works\Kuliah\Semester
4\STIMA\Tucil 1\Tucil1_13521121>r code here
```


8 8 8 9.txt

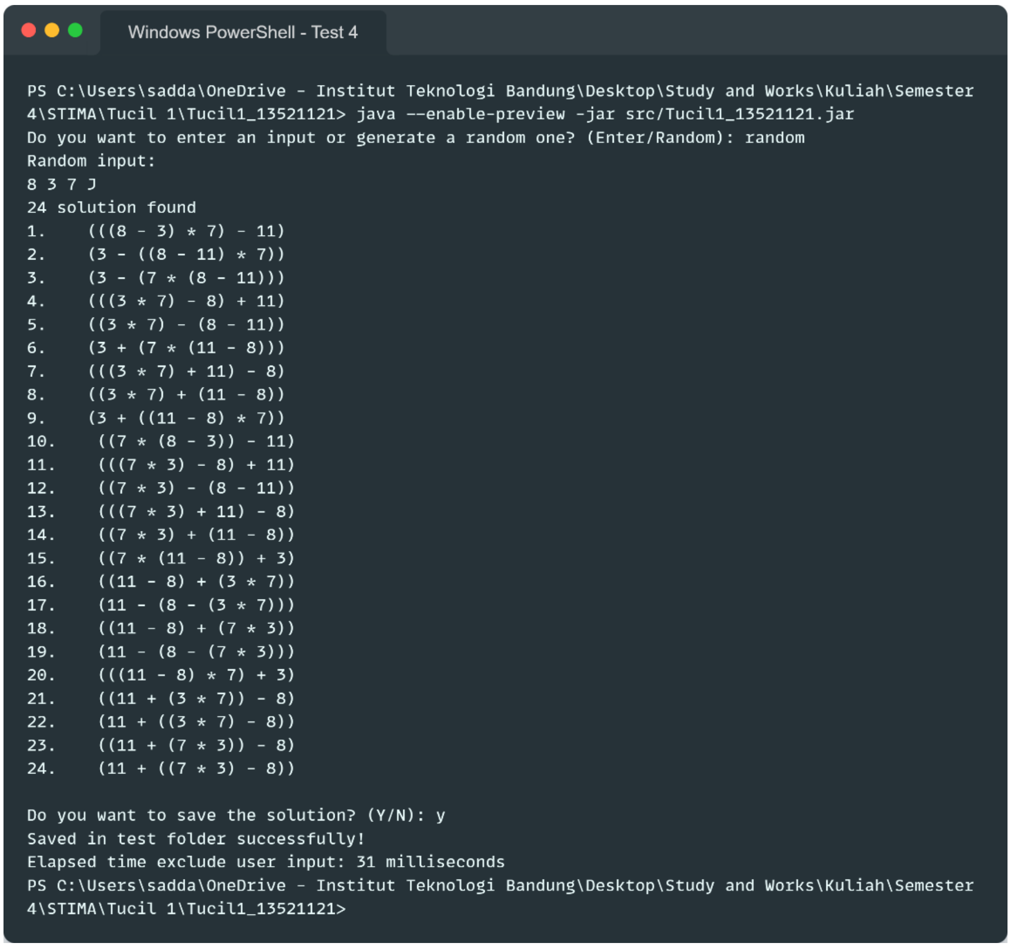
```
8 8 8 9
no solution found
```

3. Test 3

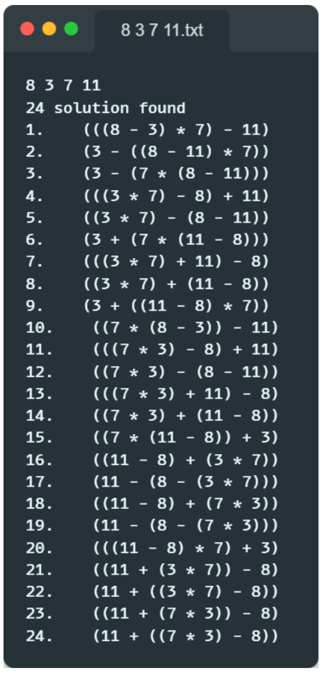

Windows PowerShell - Test 3

```
PS C:\Users\sadda\OneDrive - Institut Teknologi Bandung\Desktop\Study and Works\Kuliah\Semester
4\STIMA\Tucil 1\Tucil1_13521121> java --enable-preview -jar src/Tucil1_13521121.jar
Do you want to enter an input or generate a random one? (Enter/Random): s
Invalid choice. Please enter 'Enter' or 'Random'.
Input is incorrect.
Do you want to enter an input or generate a random one? (Enter/Random): enter

Input 4 character of A, J, Q, K, or 2-10.
Each character is separated by a space
9 9 10 K 5
Input is incorrect.
Do you want to enter an input or generate a random one? (Enter/Random): enter

Input 4 character of A, J, Q, K, or 2-10.
Each character is separated by a space
9 9 K k
Input is incorrect.
Do you want to enter an input or generate a random one? (Enter/Random): enter

Input 4 character of A, J, Q, K, or 2-10.
Each character is separated by a space
9 9 K K
no solution found
Do you want to save the solution? (Y/N): n
Solution not saved.
Elapsed time exclude user input: 25 milliseconds
PS C:\Users\sadda\OneDrive - Institut Teknologi Bandung\Desktop\Study and Works\Kuliah\Semester
4\STIMA\Tucil 1\Tucil1_13521121>
```

4. Test 4

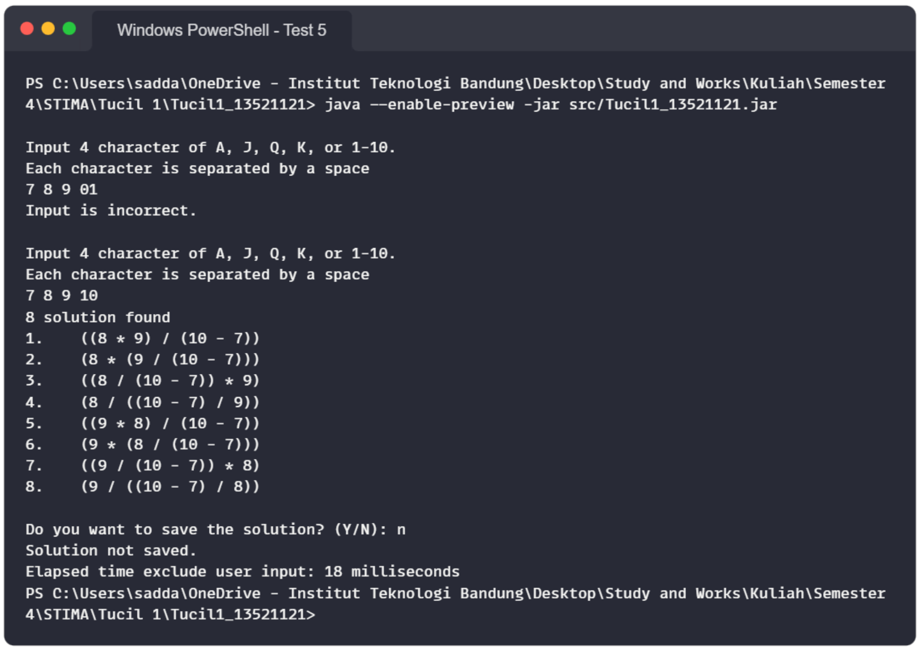

Windows PowerShell - Test 4

```
PS C:\Users\sadda\OneDrive - Institut Teknologi Bandung\Desktop\Study and Works\Kuliah\Semester
4\STIMA\Tucil 1\Tucil1_13521121> java --enable-preview -jar src/Tucil1_13521121.jar
Do you want to enter an input or generate a random one? (Enter/Random): random
Random input:
8 3 7 J
24 solution found
1.    (((8 - 3) * 7) - 11)
2.    (3 - ((8 - 11) * 7))
3.    (3 - (7 * (8 - 11)))
4.    (((3 * 7) - 8) + 11)
5.    ((3 * 7) - (8 - 11))
6.    (3 + (7 * (11 - 8)))
7.    (((3 * 7) + 11) - 8)
8.    ((3 * 7) + (11 - 8))
9.    (3 + ((11 - 8) * 7))
10.    ((7 * (8 - 3)) - 11)
11.    (((7 * 3) - 8) + 11)
12.    ((7 * 3) - (8 - 11))
13.    (((7 * 3) + 11) - 8)
14.    ((7 * 3) + (11 - 8))
15.    ((7 * (11 - 8)) + 3)
16.    ((11 - 8) + (3 * 7))
17.    (11 - (8 - (3 * 7)))
18.    ((11 - 8) + (7 * 3))
19.    (11 - (8 - (7 * 3)))
20.    (((11 - 8) * 7) + 3)
21.    ((11 + (3 * 7)) - 8)
22.    (11 + ((3 * 7) - 8))
23.    ((11 + (7 * 3)) - 8)
24.    (11 + ((7 * 3) - 8))

Do you want to save the solution? (Y/N): y
Saved in test folder successfully!
Elapsed time exclude user input: 31 milliseconds
PS C:\Users\sadda\OneDrive - Institut Teknologi Bandung\Desktop\Study and Works\Kuliah\Semester
4\STIMA\Tucil 1\Tucil1_13521121>
```


8 3 7 11.txt

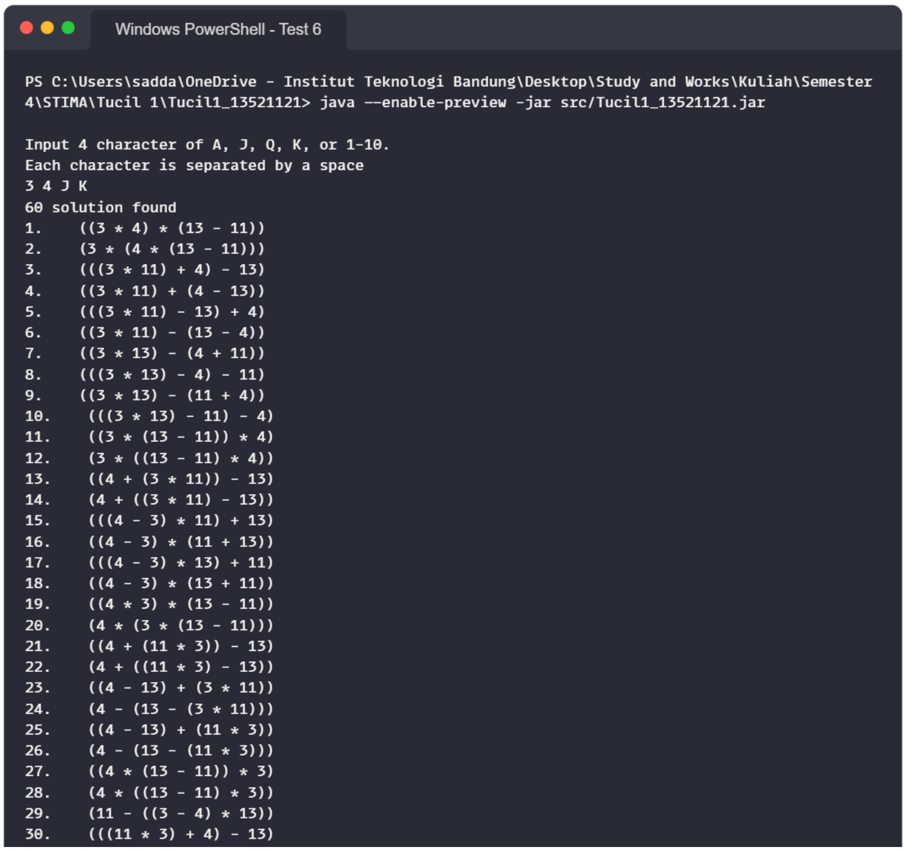
```
8 3 7 11
24 solution found
1.    (((8 - 3) * 7) - 11)
2.    (3 - ((8 - 11) * 7))
3.    (3 - (7 * (8 - 11)))
4.    (((3 * 7) - 8) + 11)
5.    ((3 * 7) - (8 - 11))
6.    (3 + (7 * (11 - 8)))
7.    (((3 * 7) + 11) - 8)
8.    ((3 * 7) + (11 - 8))
9.    (3 + ((11 - 8) * 7))
10.    ((7 * (8 - 3)) - 11)
11.    (((7 * 3) - 8) + 11)
12.    ((7 * 3) - (8 - 11))
13.    (((7 * 3) + 11) - 8)
14.    ((7 * 3) + (11 - 8))
15.    ((7 * (11 - 8)) + 3)
16.    ((11 - 8) + (3 * 7))
17.    (11 - (8 - (3 * 7)))
18.    ((11 - 8) + (7 * 3))
19.    (11 - (8 - (7 * 3)))
20.    (((11 - 8) * 7) + 3)
21.    ((11 + (3 * 7)) - 8)
22.    (11 + ((3 * 7) - 8))
23.    ((11 + (7 * 3)) - 8)
24.    (11 + ((7 * 3) - 8))
```

5. Test 5

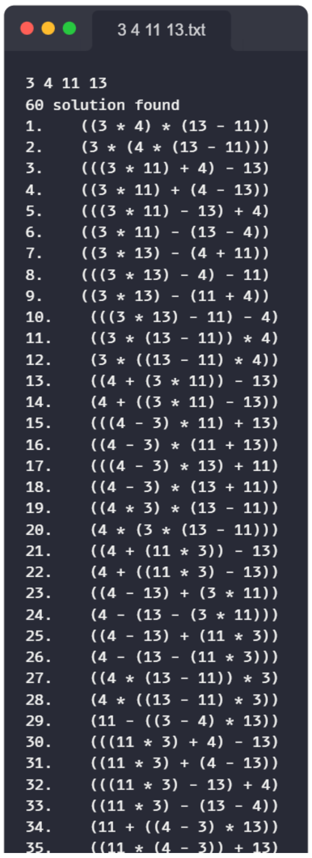


```
PS C:\Users\sadda\OneDrive - Institut Teknologi Bandung\Desktop\Study and Works\Kuliah\Semester
4\STIMA\Tucil 1\Tucil1_13521121> java --enable-preview -jar src/Tucil1_13521121.jar

Input 4 character of A, J, Q, K, or 1-10.
Each character is separated by a space
7 8 9 01
Input is incorrect.

Input 4 character of A, J, Q, K, or 1-10.
Each character is separated by a space
7 8 9 10
8 solution found
1.    ((8 * 9) / (10 - 7))
2.    (8 * (9 / (10 - 7)))
3.    ((8 / (10 - 7)) * 9)
4.    (8 / ((10 - 7) / 9))
5.    ((9 * 8) / (10 - 7))
6.    (9 * (8 / (10 - 7)))
7.    ((9 / (10 - 7)) * 8)
8.    (9 / ((10 - 7) / 8))

Do you want to save the solution? (Y/N): n
Solution not saved.
Elapsed time exclude user input: 18 milliseconds
PS C:\Users\sadda\OneDrive - Institut Teknologi Bandung\Desktop\Study and Works\Kuliah\Semester
4\STIMA\Tucil 1\Tucil1_13521121>
```

6. Test 6

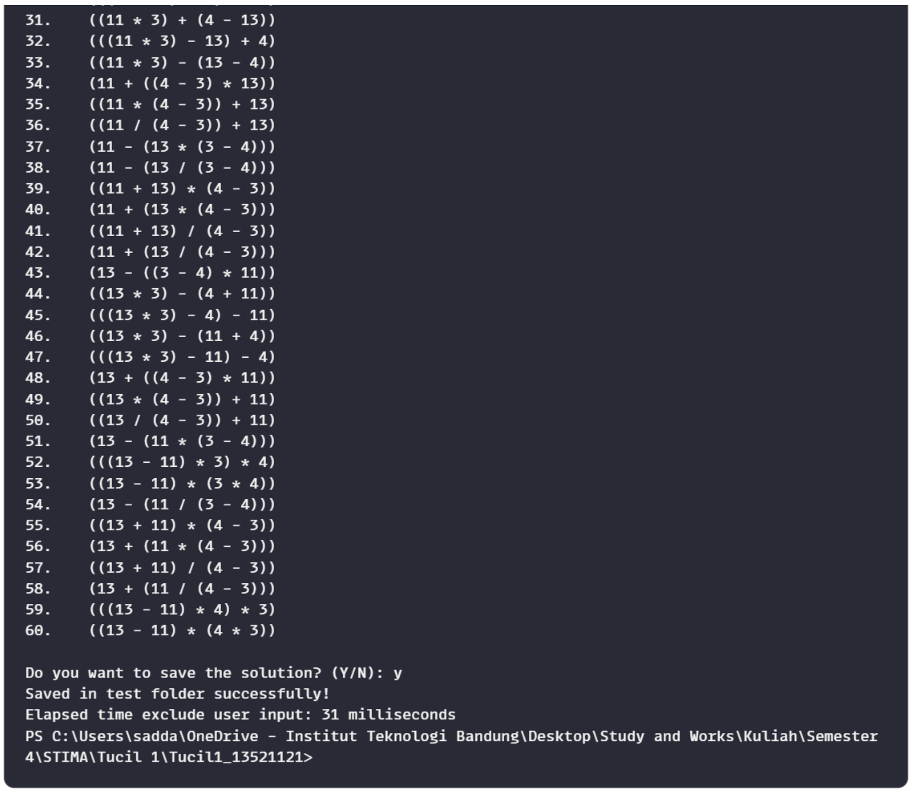


```
PS C:\Users\sadda\OneDrive - Institut Teknologi Bandung\Desktop\Study and Works\Kuliah\Semester
4\STIMA\Tucil 1\Tucil1_13521121> java --enable-preview -jar src/Tucil1_13521121.jar

Input 4 character of A, J, Q, K, or 1-10.
Each character is separated by a space
3 4 J K
60 solution found
1.    ((3 * 4) * (13 - 11))
2.    (3 * (4 * (13 - 11)))
3.    (((3 * 11) + 4) - 13)
4.    ((3 * 11) + (4 - 13))
5.    (((3 * 11) - 13) + 4)
6.    ((3 * 11) - (13 - 4))
7.    ((3 * 13) - (4 + 11))
8.    (((3 * 13) - 4) - 11)
9.    ((3 * 13) - (11 + 4))
10.    (((3 * 13) - 11) - 4)
11.    ((3 * (13 - 11)) * 4)
12.    (3 * ((13 - 11) * 4))
13.    ((4 + (3 * 11)) - 13)
14.    (4 + ((3 * 11) - 13))
15.    (((4 - 3) * 11) + 13)
16.    ((4 - 3) * (11 + 13))
17.    (((4 - 3) * 13) + 11)
18.    ((4 - 3) * (13 + 11))
19.    ((4 * 3) * (13 - 11))
20.    (4 * (3 * (13 - 11)))
21.    ((4 + (11 * 3)) - 13)
22.    (4 + ((11 * 3) - 13))
23.    ((4 - 13) + (3 * 11))
24.    (4 - (13 - (3 * 11)))
25.    ((4 - 13) + (11 * 3))
26.    (4 - (13 - (11 * 3)))
27.    ((4 * (13 - 11)) * 3)
28.    (4 * ((13 - 11) * 3))
29.    (11 - ((3 - 4) * 13))
30.    (((11 * 3) + 4) - 13)
```

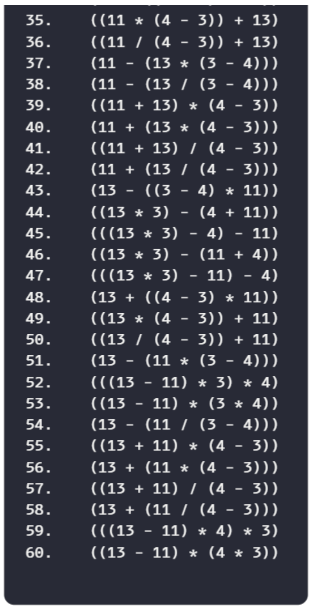


3 4 11 13.txt

```
3 4 11 13
60 solution found
1.    ((3 * 4) * (13 - 11))
2.    (3 * (4 * (13 - 11)))
3.    (((3 * 11) + 4) - 13)
4.    ((3 * 11) + (4 - 13))
5.    (((3 * 11) - 13) + 4)
6.    ((3 * 11) - (13 - 4))
7.    ((3 * 13) - (4 + 11))
8.    (((3 * 13) - 4) - 11)
9.    ((3 * 13) - (11 + 4))
10.    (((3 * 13) - 11) - 4)
11.    ((3 * (13 - 11)) * 4)
12.    (3 * ((13 - 11) * 4))
13.    ((4 + (3 * 11)) - 13)
14.    (4 + ((3 * 11) - 13))
15.    (((4 - 3) * 11) + 13)
16.    ((4 - 3) * (11 + 13))
17.    (((4 - 3) * 13) + 11)
18.    ((4 - 3) * (13 + 11))
19.    ((4 * 3) * (13 - 11))
20.    (4 * (3 * (13 - 11)))
21.    ((4 + (11 * 3)) - 13)
22.    (4 + ((11 * 3) - 13))
23.    ((4 - 13) + (3 * 11))
24.    (4 - (13 - (3 * 11)))
25.    ((4 - 13) + (11 * 3))
26.    (4 - (13 - (11 * 3)))
27.    ((4 * (13 - 11)) * 3)
28.    (4 * ((13 - 11) * 3))
29.    (11 - ((3 - 4) * 13))
30.    (((11 * 3) + 4) - 13)
31.    ((11 * 3) + (4 - 13))
32.    (((11 * 3) - 13) + 4)
33.    ((11 * 3) - (13 - 4))
34.    (11 + ((4 - 3) * 13))
35.    ((11 * (4 - 3)) + 13)
```



```
31.    ((11 * 3) + (4 - 13))
32.    (((11 * 3) - 13) + 4)
33.    ((11 * 3) - (13 - 4))
34.    (11 + ((4 - 3) * 13))
35.    ((11 * (4 - 3)) + 13)
36.    ((11 / (4 - 3)) + 13)
37.    (11 - (13 * (3 - 4)))
38.    (11 - (13 / (3 - 4)))
39.    ((11 + 13) * (4 - 3))
40.    (11 + (13 * (4 - 3)))
41.    ((11 + 13) / (4 - 3))
42.    (11 + (13 / (4 - 3)))
43.    (13 - ((3 - 4) * 11))
44.    ((13 * 3) - (4 + 11))
45.    (((13 * 3) - 4) - 11)
46.    ((13 * 3) - (11 + 4))
47.    (((13 * 3) - 11) - 4)
48.    (13 + ((4 - 3) * 11))
49.    ((13 * (4 - 3)) + 11)
50.    ((13 / (4 - 3)) + 11)
51.    (13 - (11 * (3 - 4)))
52.    (((13 - 11) * 3) * 4)
53.    ((13 - 11) * (3 * 4))
54.    (13 - (11 / (3 - 4)))
55.    ((13 + 11) * (4 - 3))
56.    (13 + (11 * (4 - 3)))
57.    ((13 + 11) / (4 - 3))
58.    (13 + (11 / (4 - 3)))
59.    (((13 - 11) * 4) * 3)
60.    ((13 - 11) * (4 * 3))

Do you want to save the solution? (Y/N): y
Saved in test folder successfully!
Elapsed time exclude user input: 31 milliseconds
PS C:\Users\sadda\OneDrive - Institut Teknologi Bandung\Desktop\Study and Works\Kuliah\Semester
4\STIMA\Tucil 1\Tucil1_13521121>
```



```
35.    ((11 * (4 - 3)) + 13)
36.    ((11 / (4 - 3)) + 13)
37.    (11 - (13 * (3 - 4)))
38.    (11 - (13 / (3 - 4)))
39.    ((11 + 13) * (4 - 3))
40.    (11 + (13 * (4 - 3)))
41.    ((11 + 13) / (4 - 3))
42.    (11 + (13 / (4 - 3)))
43.    (13 - ((3 - 4) * 11))
44.    ((13 * 3) - (4 + 11))
45.    (((13 * 3) - 4) - 11)
46.    ((13 * 3) - (11 + 4))
47.    (((13 * 3) - 11) - 4)
48.    (13 + ((4 - 3) * 11))
49.    ((13 * (4 - 3)) + 11)
50.    ((13 / (4 - 3)) + 11)
51.    (13 - (11 * (3 - 4)))
52.    (((13 - 11) * 3) * 4)
53.    ((13 - 11) * (3 * 4))
54.    (13 - (11 / (3 - 4)))
55.    ((13 + 11) * (4 - 3))
56.    (13 + (11 * (4 - 3)))
57.    ((13 + 11) / (4 - 3))
58.    (13 + (11 / (4 - 3)))
59.    (((13 - 11) * 4) * 3)
60.    ((13 - 11) * (4 * 3))
```

IV. *Check List* Program

Poin	Ya	Tidak
1. Program dikompilasi tanpa kesalahan	✓	
2. Program berhasil running	✓	
3. Program dapat membaca input / generate sendiri dan memberikan luaran	✓	
4. Solusi yang diberikan program memenuhi (berhasil mencapai 24)	✓	
5. Program dapat menyimpan solusi dalam file teks	✓	

V. Tautan Repository Github

https://github.com/SaddamAnnais/Tucil1_13521121