Practice Assignment 3

Phạm Trần Gia Hưng – 2331200153

Save the test code, and coverage report as per question numbers. Make a zip folder of all the files and upload in Moodle.

Test Coverage:

TestJacoco

TestJacoco

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
org.example		100%		98%	1	58	0	102	0	13	0	4
Total	0 of 475	100%	1 of 87	98%	1	58	0	102	0	13	0	4

org.example

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
NextDate		100%		98%	1	36	0	60	0	5	0	1
TwoNumbersSum		100%		100%	0	7	0	18	0	2	0	1
CaesarShiftCipher		100%		100%	0	7	0	14	0	2	0	1
CheckSTR		100%		100%	0	8	0	10	0	4	0	1
Total	0 of 475	100%	1 of 87	98%	1	58	0	102	0	13	0	4

Question 1: addTwoNumbers

TwoNumbersSum

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
addTwoNumbers(ArrayList, ArrayList)		100%		100%	0	6	0	17	0	1
TwoNumbersSum()		100%		n/a	0	1	0	1	0	1
Total	0 of 78	100%	0 of 10	100%	0	7	0	18	0	2

Question 2: CaesarShiftCipher

CaesarShiftCipher

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
CaesarShift(String, int)		100%		100%	0	6	0	13	0	1
CaesarShiftCipher()		100%		n/a	0	1	0	1	0	1
Total	0 of 66	100%	0 of 10	100%	0	7	0	14	0	2

Question 3: NextDate

NextDate

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
next_date()		100%		97%	1	24	0	49	0	1
check(int, int)		100%		100%	0	6	0	3	0	1
isleap(int)		100%		100%	0	4	0	3	0	1
setYearmonthdate(int, int, int)		100%		n/a	0	1	0	4	0	1
NextDate()		100%		n/a	0	1	0	1	0	1
Total	0 of 276	100%	1 of 59	98%	1	36	0	60	0	5

Question 4: ChekcSTR

CheckSTR

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
total(String)		100%		100%	0	3	0	3	0	1
binarise(int)		100%		100%	0	3	0	5	0	1
convert(String)		100%		n/a	0	1	0	1	0	1
CheckSTR()		100%		n/a	0	1	0	1	0	1
Total	0 of 55	100%	0 of 8	100%	0	8	0	10	0	4

1. The method addTwoNumbers in TwoNumbersSum.java file should add two non-negative integers from which the digits are stored separately in an arraylist. For example, the number 12 would be stored as ArrayList(1,2). You can assume that there are no zeros in the beginning of the ArrayList i.e: not ArrayList(0, 0, 2).

a. Test the attached source code (TwoNumbersSum.java) in order to achieve 100% Branch Coverage. Upload the **test code** and **coverage report** in Moodle. **15 Points**

b. There are bugs in this implementation. If you find the bug, just write the description of bugs as a comment in the TwoNumbersSum.java file. Upload **the file alone with the bug-description** in Moodle. **10 Points.**

```java
package org.example;

import ...

// error version dont add final carry digit, so if adding numbers like 999 + 1 would give 000 ins
public class TwoNumbersSum {  21 usages  Phạm Trần Gia Hưng

    public ArrayList<Integer> addTwoNumbers(ArrayList<Integer> first, ArrayList<Integer> second)
        Collections.reverse(first);
        Collections.reverse(second);

        int complement = 0;
        ArrayList<Integer> result = new ArrayList<>();

        for(int i = 0; i < Math.max(first.size(), second.size()); i++){
            int firstVal = i < first.size() ? first.get(i) : 0;
            int secondVal = i < second.size() ? second.get(i) : 0;
            int total = firstVal + secondVal + complement;
            complement = 0;
            if (total >= 10) {
                complement = 1;
                total -= 10;
            }
            result.add(i, total);
        }
        // add complement when it more than 0
        if (complement > 0) {
            result.add(complement);
        }
        Collections.reverse(result);
        return result;
    }
```

2. The CaesarShiftCipher class is responsible for applying the encryption method Caesar cipher. The rule of this method is to shift letters of a message by a given offset.

Let us say we want to shift the alphabet by 3, then letter a would be transformed to letter d, b to e, c to f, and so on.

Here is the complete matching between original and transformed letters for an offset of 3:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

As we can see, once the transformation goes beyond the letter Z, we go back to the start of the alphabet, so that X, Y and Z are transformed into A, B and C, respectively.

public String CaesarShift(String message, int shift) - this method takes input the “message” which have to be encrypted and “shift” which represents the offset. This method will return the encrypted message. For example:

Input: message: ATTACKATONCE, Shift: 4

Output: EXXEGOEXSRGI

The program only takes uppercase alphabets as input.

a. Execute test cases that give you 100% branch coverage for this source code. Upload **the test code** and **coverage report** in Moodle. **15 Points**

b. There are bugs in the implementation. Find the bug and write the description of bugs in CaesarShiftCipher.java file as a comment. Upload **the file alone with the bug-description** in Moodle. **10 Points**

```java
package org.example;

//bug
//sb.append(currentChar);
// else if ((char) (currentChar + shift) >= 'Z')
//else if ((char) (currentChar + shift) <= 'A')
public class CaesarShiftCipher {  25 usages  Phạm Trần Gia Hưng

public String CaesarShift(String message, int shift){  12 usages  Phạm Trần Gia Hưng
    StringBuilder sb = new StringBuilder();
    char currentChar;
    int length = message.length();

    shift = shift%26;   Hưng, Today • push

    for(int i = 0; i < length; i++){
        currentChar = message.charAt(i);

        // sb.append(currentChar); -- cause the output to appends before anything run properly

        if (currentChar > 'Z' || currentChar < 'A') {
            return "invalid";
        }

        // else if ((char) (currentChar + shift) >= 'Z') {
        // >=  wrap the character 'Z' not needed
        else if ((char) (currentChar + shift) > 'Z') {
            currentChar = (char) (currentChar - 26);
        }

        // else if ((char) (currentChar + shift) <= 'A'){ - wrong
        // <=  wrap the character 'A' not needed
        else if ((char) (currentChar + shift) < 'A'){
            currentChar = (char) (currentChar + 26);
        }
        sb.append((char) (currentChar + shift));
    }

    return sb.toString();
}
```

3. NextDate is a function with three variables: month, day, year. It returns the date of the day after the input date. Limitation: 1800-2025

Treatment Summary: If it is not the last day of the month, the next date function will simply, increment the day value. At the end of a month, the next day is 1 and the month is incremented. At the end of the year, both the day and the month are reset to 1, and the year incremented. Finally, the problem of leap year makes determining the last day of a month interesting.

a. Execute test cases that give you 100% branch coverage for this source code. Upload the **test code** and **coverage report** in Moodle. **15 Points**

b. There are bugs in the implementation. Find the bug and write the description of bugs in Next_Date.java file as a comment. Upload the file alone with the bug-description in Moodle. **10 Points**

```
//public void setYearmonthdate(int year, int date, int month) {
//parameter order is wrong should be year,month, date
public void setYearmonthdate(int year, int month, int date) {  30 usages  Phạm Trần Gi
    this.year = year;
```

```
}
// else if ((isleap(year) == 0) && day >= 28) {
//Should be day > 28 not day >= 28
else if ((isleap(year) == 0) && day > 28) {
    s1 = "invalid date input for not a leap year";
    return s1;
}
```

```
case 12:
    //if (day <= 31)
    //correct is day < 31 not day <= 31
    if (day < 31)
```

4. The program in ChekcSTR.java file converts an ASCII string to the sum of its characters represented as a Binary string i.e. adds up the values (ASCII) of strings, character by character, and show the sum as a binary digit.

a. Create and execute test cases to achieve 100% branch coverage. Save the test code as .java file. **15 Points**

b. The implementation of ChekcStr.java has bugs. Identify bugs and write the description of the bugs in the ChekcStr.java file as a comment. Upload the file alone with the bug-description in Moodle. **10 Points.**

```java
package org.example;

// bug
// return 1+binarise(givenstrvalue/2);
// return "0"+binarise(givenstrvalue/2);

public class CheckSTR {  11 usages  Phạm Trần Gia Hưng *  Hưng, Today • push
    public String convert(String str) { return binarise(total(str)); }

        public int total(String str) {  2 usages  Phạm Trần Gia Hưng
        if(str=="") return 0;
        if(str.length()==1) return ((int)(str.charAt(0)));
        return ((int)(str.charAt(0)))+total(str.substring( beginIndex: 1));
        }

        public String binarise(int givenstrvalue) {  3 usages  Phạm Trần Gia Hưng *
        if(givenstrvalue==0)
            return "";
        if(givenstrvalue %2==1)
        //return 1+binarise(givenstrvalue/2);
            //creates a reversed binary string because prepend 1
            return binarise( givenstrvalue: givenstrvalue/2) + "1";
        //return "0"+binarise(givenstrvalue/2);
        // creates a reversed binary string because prepend 0
        return binarise( givenstrvalue: givenstrvalue/2) + "0";
        }
}
```