IT602: Object-Oriented Programming

Lab Assignment-2

Sagar Variya

202112114

Question #1:

On a space expedition, a space explorers' ship crashed on Mars! The designated SOS message is "MAYDAY". The team sends a series of SOS messages to Earth for help. However, due to the cosmic radiation, letters in some of the SOS messages are altered. The signal received by an Earth station is a string. As a space scientist, your task is to determine how many letters of the SOS message have been changed by the radiation so that the actual message can be retrieved.
Example Input: "MAYDXYMAYDAY"
Example Output: 1 (i.e., Only 5th character is altered)

Code #1:

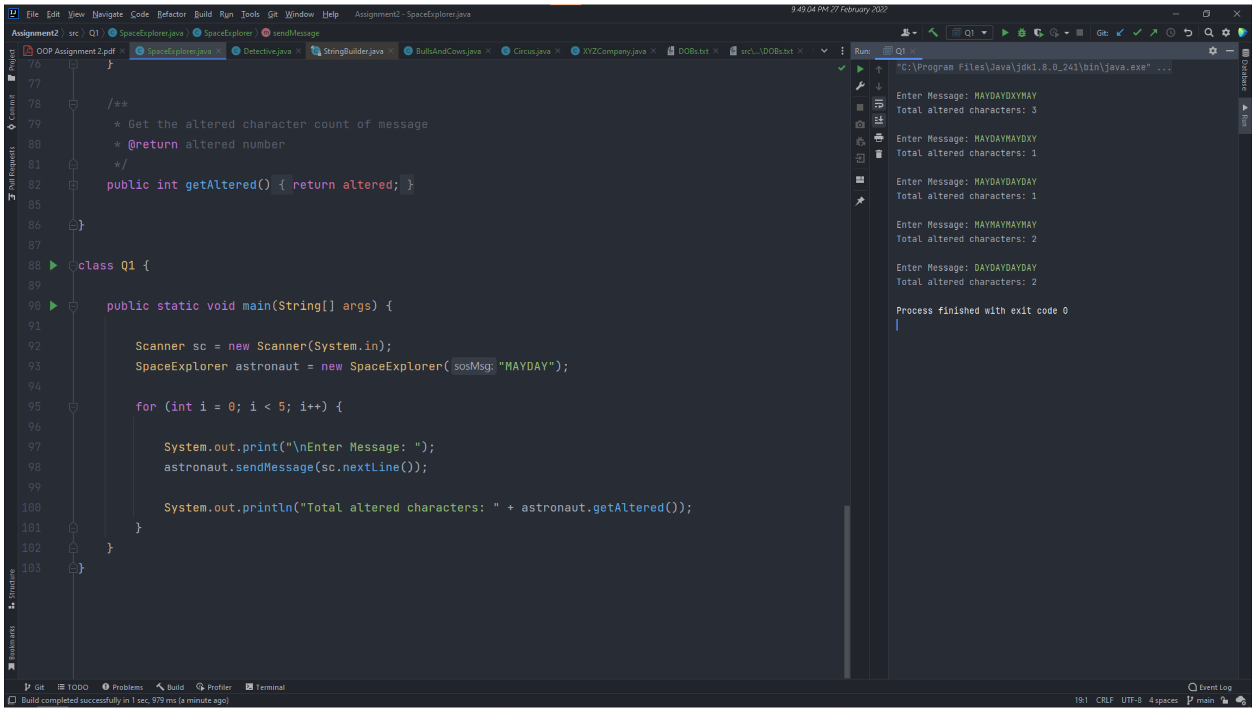
```
package Q1;

import java.util.Scanner;

/**
 * Space Explorer class can check altered characters of messages come from
astronauts.
 * @version 22.2.26
 * @author SaGaR VaRiyA
 */
public class SpaceExplorer {

   /**
    * The original msg and count of altered  characters.
    */
   private final String sosMsg;
   private int altered;
```

```
/**
 * Creates a new Space Explorer with the given original message.
 * @param sosMsg This original sos message
 */
public SpaceExplorer(String sosMsg) {
   this.sosMsg = sosMsg;
}

/**
 * Correct the message from astronaut.
 * @param msg message for correction
 * @return corrected message
 */
private String messageCorrection(String msg) {

   int strLen = msg.length();
   int sosMsgLen = sosMsg.length();
   int missingChars = strLen % sosMsgLen;

   if (missingChars != 0) {
      missingChars = sosMsgLen - missingChars;
   }

   if (missingChars > 0) {
      StringBuilder strBuilder = new StringBuilder(msg);
      while (missingChars-- > 0) strBuilder.append("_");
      msg = strBuilder.toString();
   }

   return msg;
}

/**
 * see {@link #messageCorrection(String)}
 * Astronaut sending message to earth
```

```
     * @see #messageCorrection(String)
     * @param msg This is astronaut's message
     */
    public void sendMessage(String msg) {

        altered = 0;
        int strLen = msg.length();
        msg = messageCorrection(msg);
        int sosMsgLen = sosMsg.length();
        int startIndex = 0, msgCount = strLen / sosMsgLen;

        while (msgCount-- > 0) {

            String tempMsg = msg.substring(startIndex, startIndex += sosMsgLen);

            if (!sosMsg.equals(tempMsg)) {
                for (int i = 0; i < sosMsgLen; i++) {
                    if (sosMsg.charAt(i) != tempMsg.charAt(i)) {
                        altered++;
                    }
                }
            }
        }
    }

    /**
     * Get the altered character count of message
     * @return altered number
     */
    public int getAltered() {
        return altered;
    }

}
```

```
class Q1 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        SpaceExplorer astronaut = new SpaceExplorer("MAYDAY");

        for (int i = 0; i < 5; i++) {

            System.out.print("\nEnter Message: ");
            astronaut.sendMessage(sc.nextLine());

            System.out.println("Total altered characters: " + astronaut.getAltered());
        }
    }
}
```

Output #1:



```java
    }

    /**
     * Get the altered character count of message
     * @return altered number
     */
    public int getAltered() { return altered; }

}

class Q1 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        SpaceExplorer astronaut = new SpaceExplorer( sosMsg: "MAYDAY");

        for (int i = 0; i < 5; i++) {

            System.out.print("\nEnter Message: ");
            astronaut.sendMessage(sc.nextLine());

            System.out.println("Total altered characters: " + astronaut.getAltered());
        }
    }
}
```

```
"C:\Program Files\Java\jdk1.8.0_241\bin\java.exe" ...

Enter Message: MAYDAYDXYMAY
Total altered characters: 3

Enter Message: MAYDAYMAYDXY
Total altered characters: 1

Enter Message: MAYDAYDAYDAY
Total altered characters: 1

Enter Message: MAYMAYMAYMAY
Total altered characters: 2

Enter Message: DAYDAYDAYDAY
Total altered characters: 2

Process finished with exit code 0
```

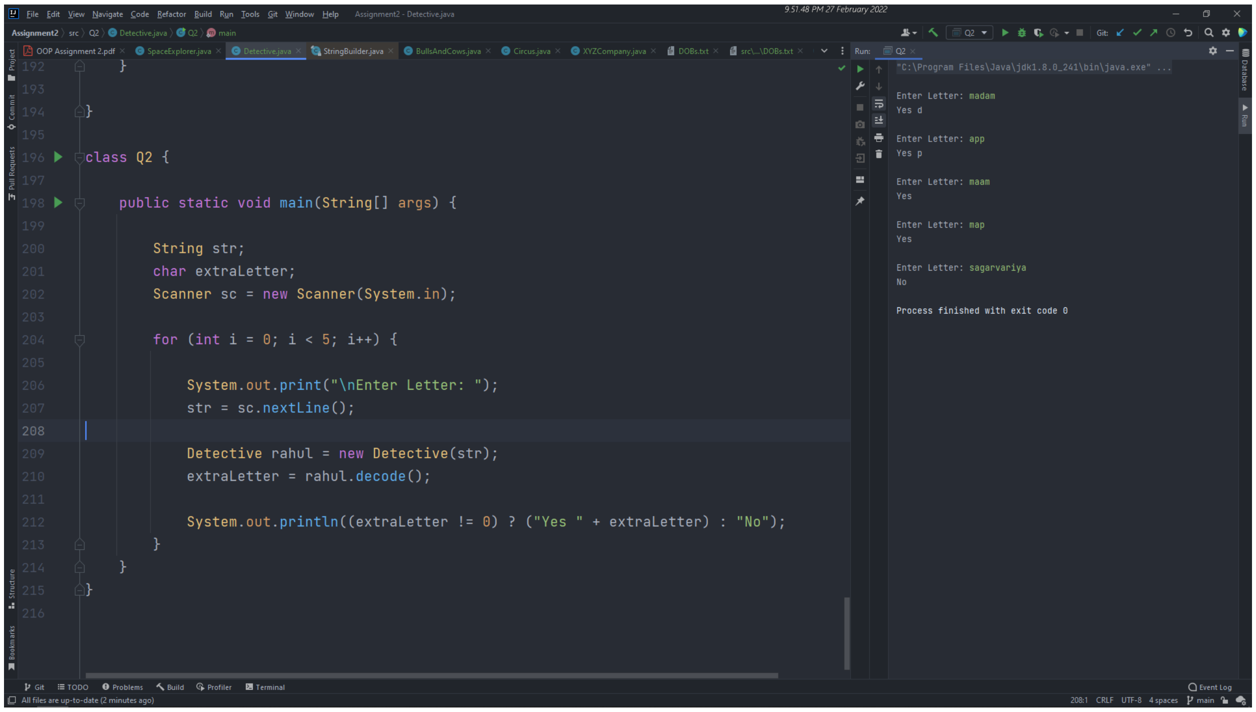
Question #2:

Rahul, a great detective, has to decode a letter to solve a mystery. He considers a string to be valid if all characters of the string appear the same number of times. He also assumes a string to be valid, if he can remove just 1 character at one index in the string, and the remaining characters will occur the same number of times. Given a string, determine if it is valid. If so, return YES, otherwise return NO. Example: “app” is a valid string as we can remove ‘p’ such that: ‘a’ occurs once and ‘p’ occurs once. If there is no extra character to be removed just print “YES”. Else also print character ch that got removed in this format “YES ”.
Input: “madam”
Output: YES d

Code #2:

```
package Q2;

import java.util.Scanner;

/**
 * A mystery solver decodes a letter from string.
 * String is valid if all the characters appears the same number of time.
 * for validation, it can remove just 1 character at one index in the string,
 * and the remaining characters will occur the same number of times.
 * @version 22.2.25
 * @author SaGaR VaRiyA
 */
public class Detective {

    /**
 * use arrLen to store the distinct character length.
 * use charCnt1 and charCnt2 to store 2 different character counts in string.
 * use letters to store the original string of letters.
 * use charArr to store all unique letters.
 * use charCountArr to store all unique letters.
 */
private int arrLen;
private int charCnt1;
private int charCnt2;
private final String letters;
private final char[] charArr;
private final int[] charCountArr;

/**
 * Construct a Detective with the original string of letters.
 * Create array according to letter length.
 * @param str original string of letters.
 */
public Detective(String str) {

   arrLen = 0;
   letters = str;
   charArr = new char[letters.length()];
   charCountArr = new int[letters.length()];
}

/**
 * Check if the character exists in counted array.
 * If character exist then increase the count of that character.
 * @param c Character to check existence.
 * @param len To check in between the length.
 * @return True if character exist else False.
 */
private boolean checkCharExist(char c, int len) {
    for (int j = 0; j < len; j++) {
        if (charArr[j] == c) {
            charCountArr[j]++;
            return true;
        }
    }
    return false;
}

/**
 * see {@link #checkCharExist(char, int)}
 * Calculate the unique characters and their number of appearance.
 * using checkCharExist it will take decision.
 * @see #checkCharExist(char, int)
 */
private void calcChars() {

    for (int i = 0; i < letters.length(); i++) {
        char tempChar = letters.charAt(i);
        boolean charExist = checkCharExist(tempChar, arrLen);

        if (!charExist) {
            charArr[arrLen] = tempChar;
            charCountArr[arrLen] = 1;
            arrLen++;
        }
    }
}

/**
 * Give how many times the number of character count appear in charCountArr.
 * @param n Get its occurrence.
 * @return Occurrence of the given number.
 */
private int giveOccurrence(int n) {
    int cnt = 0;
    for (int i = 0; i < arrLen; i++) {
        if (charCountArr[i] == n) {
            cnt++;
        }
    }
    return cnt;
}

/**
 * see {@link #calcChars()}
 * see {@link #giveOccurrence(int)}
 * A function to check if the given letter of string is valid or not.
 * @see #calcChars()
 * @see #giveOccurrence(int)
 * @return True if letters are valid else False.
 */
private boolean isValid() {

    calcChars();
    StringBuilder distinctCounts = new StringBuilder();

    for (int i = 0; i < arrLen; i++) {
        if (!distinctCounts.toString().contains(String.valueOf(charCountArr[i]))) {
            distinctCounts.append(charCountArr[i]);
        }
    }

    if (distinctCounts.length() == 1) return true;
    if (distinctCounts.length() > 2) return false;

    charCnt1 = Integer.parseInt(String.valueOf(distinctCounts.charAt(0)));
    charCnt2 = Integer.parseInt(String.valueOf(distinctCounts.charAt(1)));
    int difference = charCnt1 - charCnt2;
    difference = (difference < 0) ? difference * -1 : difference;

    if (difference == 1) {

        int occurrence1 = giveOccurrence(charCnt1);
        int occurrence2 = giveOccurrence(charCnt2);

        if (occurrence1 == 1 || occurrence2 == 1) {
            if (occurrence1 == 1 && (charCnt1 == 1 || charCnt1 - charCnt2 == 1)) {
                return true;
            } else return occurrence2 == 1 && (charCnt2 == 1 || charCnt2 - charCnt1 == 1);
        } else return false;
    }
    return false;
}

/**
 * Remove the character that appear give number of time in letters.
 * @param counts Appear count to remove.
 * @return The character that will be removed.
 */
private char removeChar(int counts) {

    for (int i = 0; i < arrLen; i++) {
        if (counts == charCountArr[i]) {
            return charArr[i];
        }
    }
    return 0;
}

/**
 * see {@link #removeChar(int)}
 * see {@link #giveOccurrence(int)}
 * Decide which character to remove based on its appearance.
 * @see #removeChar(int)
 * @see #giveOccurrence(int)
 * @return The character that will be removed.
 */
private char getRemovalChar() {

    char removedChar = ' ';
    if (charCnt1 == charCnt2) return removedChar;

    int occurrence1 = giveOccurrence(charCnt1);
    int occurrence2 = giveOccurrence(charCnt2);

    if (occurrence1 == 1 && occurrence2 == 1) {
        removedChar = removeChar(Math.max(charCnt1, charCnt2));
    } else if (occurrence1 == 1 && occurrence2 == 2) {
        removedChar = removeChar(charCnt1);
    } else if (occurrence2 == 1 && occurrence1 == 2) {
        removedChar = removeChar(charCnt2);
    } else if (occurrence1 == 1) {
        removedChar = removeChar(charCnt1);
    } else if (occurrence2 == 1) {
        removedChar = removeChar(charCnt2);
    }
    return removedChar;
}

/**
 * see {@link #isValid()}
 * see {@link #getRemovalChar()}
 * Decode the letters and give removal character.
 * @see #isValid()
 * @see #getRemovalChar()
 * @return The character that will be removed.
 */
public char decode() {
        return (!isValid()) ? 0 : getRemovalChar();
    }

}

class Q2 {

    public static void main(String[] args) {

        String str;
        char extraLetter;
        Scanner sc = new Scanner(System.in);

        for (int i = 0; i < 5; i++) {

            System.out.print("\nEnter Letter: ");
            str = sc.nextLine();

            Detective rahul = new Detective(str);
            extraLetter = rahul.decode();

            System.out.println((extraLetter != 0) ? ("Yes " + extraLetter) : "No");
        }
    }
}
```

Output #2:



```java
    }

}

class Q2 {

    public static void main(String[] args) {

        String str;
        char extraLetter;
        Scanner sc = new Scanner(System.in);

        for (int i = 0; i < 5; i++) {

            System.out.print("\nEnter Letter: ");
            str = sc.nextLine();

            Detective rahul = new Detective(str);
            extraLetter = rahul.decode();

            System.out.println((extraLetter != 0) ? ("Yes " + extraLetter) : "No");
        }
    }
}
```

```
"C:\Program Files\Java\jdk1.8.0_241\bin\java.exe" ...

Enter Letter: madam
Yes d

Enter Letter: app
Yes p

Enter Letter: maam
Yes

Enter Letter: map
Yes

Enter Letter: sagarvariya
No

Process finished with exit code 0
```

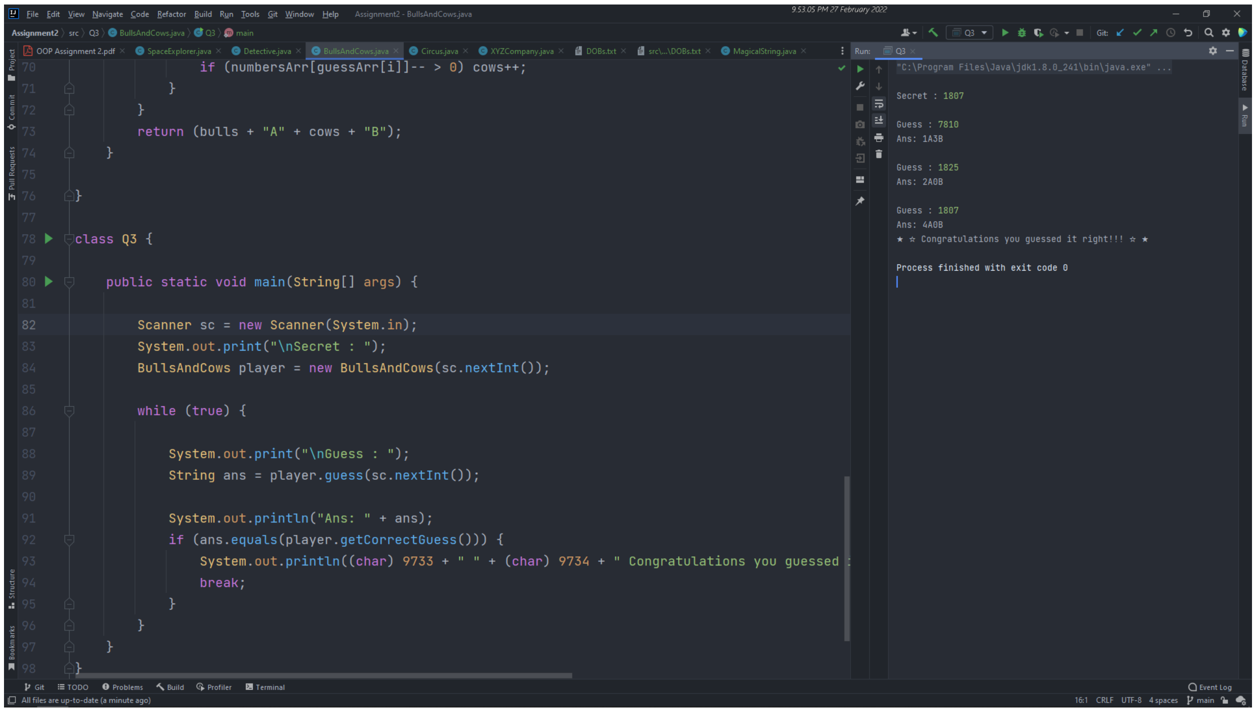
Question #3:

You are playing a very famous game, the “Bulls and cows” with your friend. You write down a secret number and ask your friend to guess what the number is. When your friend makes a guess, you provide a hint with the following info. The number of "bulls", are digits in the guess that are in the correct position. The number of "cows", which are digits in the guess that are in your secret number but are located in the wrong position. Specifically, the non-bull digits in the guess could be rearranged such that they become bulls. Given the secret number secret and your friend's guess guess, return the hint for your friend's guess. The hint should be formatted as “xAyB”, where x is the number of bulls and y is the number of cows. Note that both secret and guess may contain duplicate digits.
Input: secret = "1807", guess = "7810"
Output: "1A3B"

Code #3:

```
package Q3;

import java.util.Scanner;

/**
 * Famous game bulls and cows.
 * On a sheet of paper, the players each write a 4-digit secret number.
 * The digits must be all different. Then, in turn,
 * the players try to guess their opponent's number who gives the number of
matches.
 * If the matching digits are in their right positions, they are "bulls",
 * if in different positions, they are "cows"
 * @version 22.2.23
 * @author SaGaR VaRiyA
 */
public class BullsAndCows {

    /**
     * using secret to store the secret number.
     * using answer to store the correct answer.
     */
    private final int secret;
    private final String ans;

    /**
     * Construct a game with secret number to guess.
     * Initialize the correct answer.
     * @param secret Secret number to guess.
     */
    public BullsAndCows(int secret) {
        this.secret = secret;
        ans = guess(this.secret);
    }

    /**
     * Get the correct answer of secret number.
     * @return correct answer.
     */
    public String getCorrectGuess() {
        return ans;
    }

    /**
     * Guess the number.
     * @param guess guessed number.
     * @return answer of guessed number.
     */
    public String guess(int guess) {
        int s = secret, g = guess;
        int Length = Integer.toString(this.secret).length();
        int[] numbersArr = new int[10];
        int[] guessArr = new int[Length];
        int[] secretArr = new int[Length];
        int bulls = 0, cows = 0, index = 0;

        do {
            secretArr[index++] = (s % 10);
        } while ((s /= 10) > 0);

        index = 0;
        do {
            guessArr[index++] = (g % 10);
        } while ((g /= 10) > 0);


        for (int i = 0; i < Length; i++) {
            if (secretArr[i] == guessArr[i]) bulls++;
            else {
                if (numbersArr[secretArr[i]]++ < 0) cows++;
                if (numbersArr[guessArr[i]]-- > 0) cows++;
            }
        }
        return (bulls + "A" + cows + "B");
    }

}

class Q3 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        System.out.print("\nSecret : ");
        BullsAndCows player = new BullsAndCows(sc.nextInt());

        while (true) {

            System.out.print("\nGuess : ");
            String ans = player.guess(sc.nextInt());

            System.out.println("Ans: " + ans);
            if (ans.equals(player.getCorrectGuess())) {
                System.out.println((char) 9733 + " " + (char) 9734 + " Congratulations you
guessed it right!!! " + (char) 9734 + " " + (char) 9733);
                break;
            }
        }
    }
}
```

Output #3:



```java
                if (numbersArr[guessArr[i]]-- > 0) cows++;
            }
        }
        return (bulls + "A" + cows + "B");
    }

}

class Q3 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        System.out.print("\nSecret : ");
        BullsAndCows player = new BullsAndCows(sc.nextInt());

        while (true) {

            System.out.print("\nGuess : ");
            String ans = player.guess(sc.nextInt());

            System.out.println("Ans: " + ans);
            if (ans.equals(player.getCorrectGuess())) {
                System.out.println((char) 9733 + " " + (char) 9734 + " Congratulations you guessed
                break;
            }
        }
    }
}
```

```
"C:\Program Files\Java\jdk1.8.0_241\bin\java.exe" ...

Secret : 1807

Guess : 7810
Ans: 1A3B

Guess : 1825
Ans: 2A0B

Guess : 1807
Ans: 4A0B
★ ☆ Congratulations you guessed it right!!! ☆ ★

Process finished with exit code 0
```

Question #4:

Shepherd is a ring master at the grand royal circus. He has n rings. Rings can be either red, green, or blue. There are a total of ten rods that are present on the stage. Shepherd is trying to entertain the audience by putting rings (i.e., 'n' is the number of rings) onto the rods. All the rods are numbered from 0 to 9. As an audience, your job is to find how many rods have all the colored rings. For that, you have one string containing the information telling which ring is onto which rod. For example, the string "R3G2B1" gives a description for 3 rings: a red ring placed onto the rod labeled 3, a green ring placed onto the rod labeled 2, and a blue ring placed onto the rod labeled 1. Here in the given case, the answer would be zero as no rods have all three rings. Your answer should take care of all the possible edge cases.
Input: "R3G2B1G3B3"
Output: 1 (only rod-3 has all three rings)

Code #4:

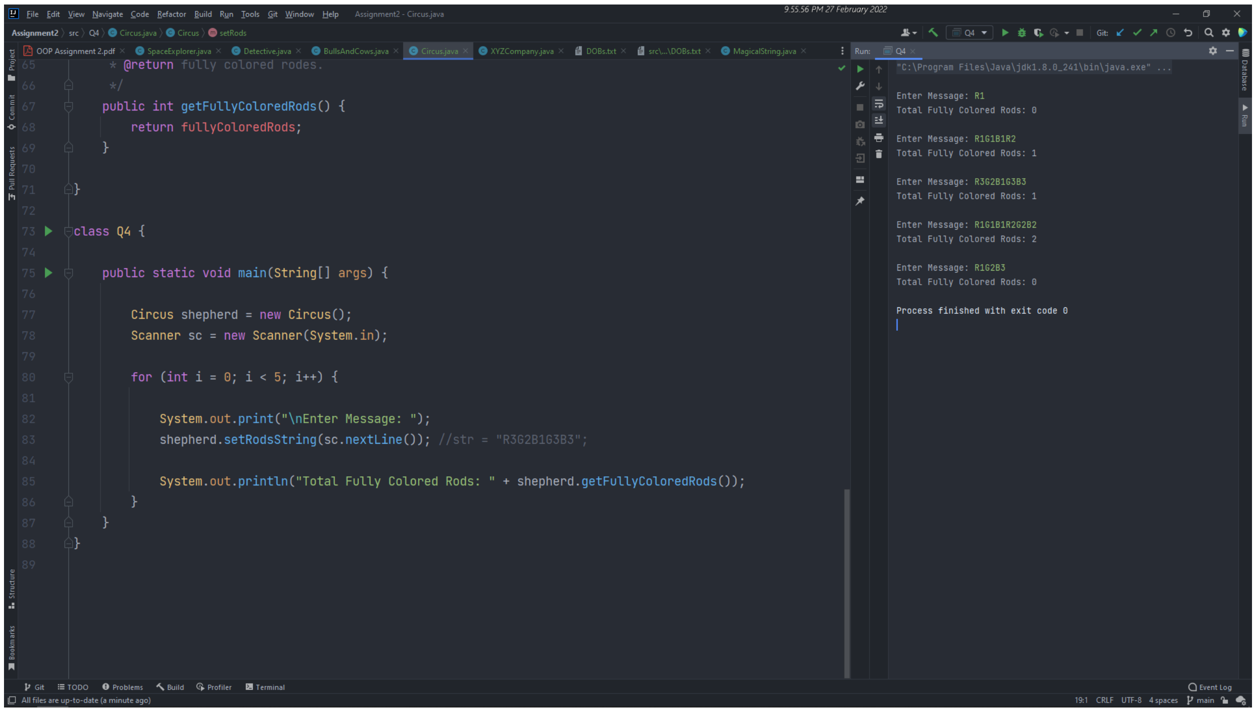
```
package Q4;

import java.util.Scanner;

/**
 * Circus game of ring and rods.
 * Rings can be either red, green, or blue.
 * There are a total of ten rods that are present on the stage.
 * The rod that has all three colored ring is called fully colored rode.
 * This class has default constructor.
 * @version 22.2.22
 * @author SaGaR VaRiyA
 */
```

```
public class Circus {

   /**
    * using rodes array to store rings position in rods.
    * using fullyColoredRods to count the rod that has all three colored rings.
    */
   boolean[][] rods;
   int fullyColoredRods;

   /**
    * using rods to store ring positions in all rods.
    * using fullyColoredRods to count Rods that has all three colored rings.
    * Set the rods with different colored rings and count fully colored rode.
    * @param s Thrown ring in rods.
    */
   private void setRods(String s) {

      fullyColoredRods = 0;
      rods = new boolean[10][3];

      for (int i = 0; i < s.length(); i += 2) {

         int colorNum = -1;
         char color = s.charAt(i);
         int rod = Integer.parseInt(String.valueOf(s.charAt(i + 1)));

         if (color == 'R') colorNum = 0;
         if (color == 'G') colorNum = 1;
         if (color == 'B') colorNum = 2;

         rods[rod][colorNum] = true;

         if (rods[rod][0] && rods[rod][1] && rods[rod][2]) fullyColoredRods++;
      }
   }
```

```
    /**
     * see {@link #setRods(String)}
     * Set rods if given rod string is valid.
     * It will set rods via setRods method.
     * @see #setRods(String)
     * @param str Thrown rings in rods string
     */
    public void setRodsString(String str) {

        str = str.toUpperCase();
        if (str.length() % 2 == 0) setRods(str);
    }

    /**
     * Get the fully colored rode count
     * @return fully colored rodes.
     */
    public int getFullyColoredRods() {
        return fullyColoredRods;
    }

}
```

```
class Q4 {

   public static void main(String[] args) {

      Circus shepherd = new Circus();
      Scanner sc = new Scanner(System.in);

      for (int i = 0; i < 5; i++) {

         System.out.print("\nEnter Message: ");
         shepherd.setRodsString(sc.nextLine()); //str = "R3G2B1G3B3";

         System.out.println("Total Fully Colored Rods: " +
shepherd.getFullyColoredRods());
      }
   }
}
```

Output #4:



```java
     * @return fully colored rodes.
     */
    public int getFullyColoredRods() {
        return fullyColoredRods;
    }

}

class Q4 {

    public static void main(String[] args) {

        Circus shepherd = new Circus();
        Scanner sc = new Scanner(System.in);

        for (int i = 0; i < 5; i++) {

            System.out.print("\nEnter Message: ");
            shepherd.setRodsString(sc.nextLine()); //str = "R3G2B1G3B3";

            System.out.println("Total Fully Colored Rods: " + shepherd.getFullyColoredRods());
        }
    }
}
```

```
"C:\Program Files\Java\jdk1.8.0_241\bin\java.exe" ...

Enter Message: R1
Total Fully Colored Rods: 0

Enter Message: R1G1B1R2
Total Fully Colored Rods: 1

Enter Message: R3G2B1G3B3
Total Fully Colored Rods: 1

Enter Message: R1G1B1R2G2B2
Total Fully Colored Rods: 2

Enter Message: R1G2B3
Total Fully Colored Rods: 0

Process finished with exit code 0
```

Question #5:

You are appointed as a Data Engineer in XYZ company. Your task is to count the people whose age is below 50. For performing the task, you can assume that you're given a text file in which they have entered their birthdates manually. While collecting data, they have forgotten to tell the people about the prescribed format. Some of the dates contain the alphabet. Your job is to identify the number of faulty dates and correct them into the prescribed format along with calculating the number of people who are below 50. Output the count and the revised list of dates.

Notes:

- The date must be in the prescribed format (i.e., DD-MM-YYYY)
- Your code should not terminate if the file name is not as desired. You should give appropriate warnings/comments to the user.
- Clearly mention which types of exceptions you have handled in the Remarks section
- Write a custom message for each Exception

Example input text file

```
10-10-1985
05-12-1997
01-jan-1998
30-may-1972
5-10-1997
```

Code #5:

```
package Q5;

import java.io.*;
import java.util.*;
import java.nio.file.*;
import java.nio.charset.StandardCharsets;

/**
 * Company contains their employees' data in txt file.
 * you can calculate the number of employees who are below 50 year old.
 * @version 22.2.26
 * @author SaGaR VaRiyA
 */
public class XYZCompany {

   /**
    * using sc to open file.
    * using below50 to count dates below 50.
    * using file to store the file.
    * using currYear to store current year.
    * using currDay to store current day.
    * using currMonth to store current month.
    * using invalid to return invalid statement.
    */
   private Scanner sc;
   private int below50;
   private final File file;
   private final int currYear;
   private final byte currDay;
   private final byte currMonth;
   private final String invalid;
```

```
/**
 * Construct company with storing the file.
 * Initialization of many variables like
 * currDay, currMonth, currYear, below50, invalid, file
 * with respect to their use.
 * @param fileName file containing dates
 */
public XYZCompany(String fileName) {

    Calendar cal = Calendar.getInstance();
    currDay = (byte) cal.get(Calendar.DATE);
    currMonth = (byte) (cal.get(Calendar.MONTH) + 1);
    currYear = cal.get(Calendar.YEAR);
    below50 = 0;
    invalid = "INVALID DATE";
    file = new File(fileName);
}

/**
 * Get below 50 year count
 * @return count of less than 50 years
 */
public int getBelow50() {
    return below50;
}

/**
 * Display file data.
 */
public void displayFileData() {

    try {
        sc = new Scanner(file);
        sc.useDelimiter("\\Z");
        System.out.println(sc.next());
    } catch (IOException e) {
        System.out.println("Error: " + e.getMessage());
    }
}

/**
 * Count the year if it is below 50
 * @param day day of DOB
 * @param month month of DOB
 * @param year year of DOB
 */
private void countIfBelow50(int day, int month, int year) {

    if (currYear - year > 0 && currYear - year < 50) {
        below50++;
    } else if (currYear - year == 50) {
        if (currMonth - month > 0 && currMonth - month < 12) {
            below50++;
        } else if (currMonth - month == 0) {
            if (currDay - day >= 0 && currDay - day < currDay) {
                below50++;
            }
        }
    }
}

/**
 * give the year is leap or not
 * @param year year of DOB
 * @return true if year is leap else false
 */
private boolean isLeapYear(int year) {
    return (year % 4 == 0 && (year % 100 != 0 || year % 400 == 0));
}
```

```
    /**
     * A function to correct the date in int format
     * if its wrong then return invalid
     * @param day day of DOB
     * @return corrected day
     */
    private String dayCorrection(String day) {

        if (!day.matches("[0-9]+")) return invalid;
        byte d = Byte.parseByte(day);
        return (d < 10) ? "0" + d : String.valueOf(d);
    }

    /**
     * A function to correct the month in int format
     * if its wrong then return invalid
     * @param month month of DOB
     * @return corrected month
     */
    private String monthCorrection(String month) {

        byte m;
        if (month.matches("[0-9]+") && (month.length() == 1 || month.length() == 2) &&
Byte.parseByte(month) > 0 && Byte.parseByte(month) <= 12) {
            m = Byte.parseByte(month);
        } else {
            month = month.toLowerCase();
            switch (month) {
                case "jan":
                case "january":
                    m = 1;
                    break;
                case "feb":
                case "february":
                    m = 2;
    break;
case "mar":
case "march":
    m = 3;
    break;
case "apr":
case "april":
    m = 4;
    break;
case "may":
    m = 5;
    break;
case "jun":
case "june":
    m = 6;
    break;
case "jul":
case "july":
    m = 7;
    break;
case "aug":
case "august":
    m = 8;
    break;
case "sep":
case "sept":
case "september":
    m = 9;
    break;
case "oct":
case "october":
    m = 10;
    break;
case "nov":
case "november":
```

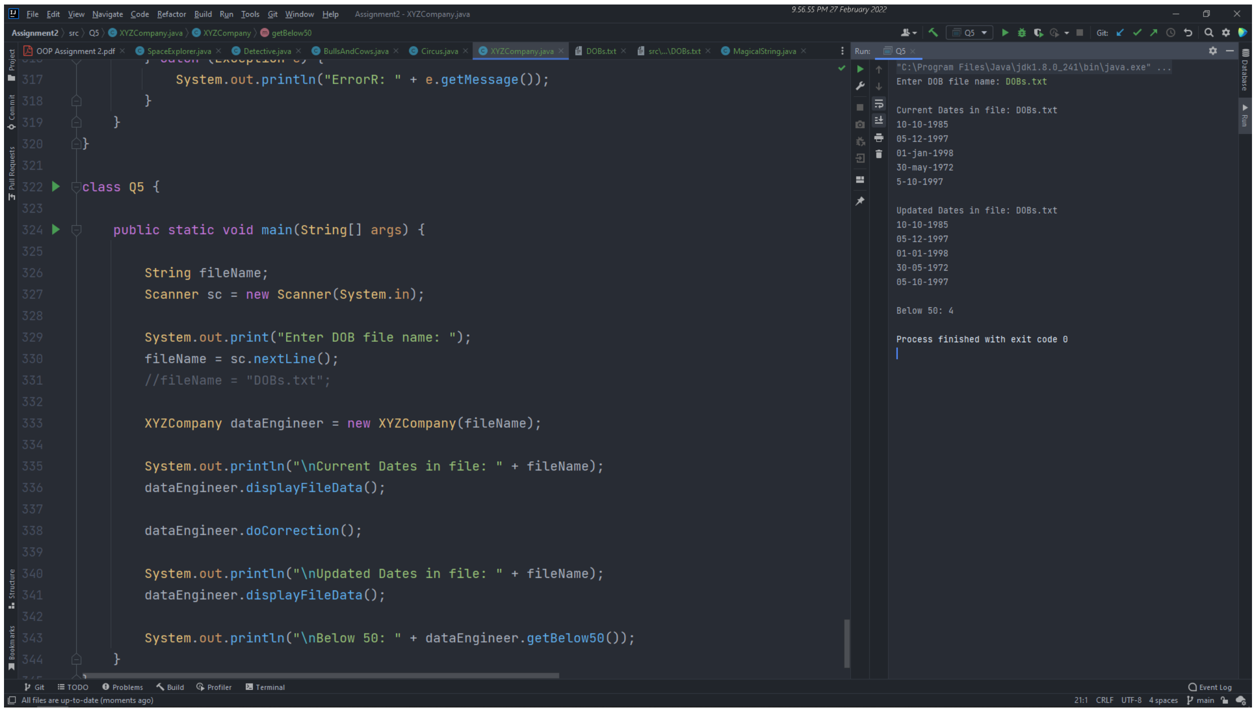
```
            m = 11;
            break;
        case "dec":
        case "december":
            m = 12;
            break;
        default:
            return invalid;
        }
    }

    return (m < 10) ? "0" + m : String.valueOf(m);
}

/**
 * A function to correct the year in int format
 * if its wrong then return invalid
 * @param year year of DOB
 * @return corrected year
 */
private String yearCorrection(String year) {

    if (!year.matches("[0-9]+")) return invalid;
    return (Integer.parseInt(year) >= 0) ? year : invalid;
}

/**
 * see {@link #isLeapYear(int)}
 * see {@link #countIfBelow50(int, int, int)}
 * A function that will validate the date string into valid formatted date
 * @param day day of DOB
 * @param month month of DOB
 * @param year year of DOB
 * @return valid DOB in string format
 */
```

```
    private String dateValidation(String day, String month, String year) {

        if (month.equals("99")) return invalid;
        boolean leapYear = isLeapYear(Integer.parseInt(year));

        byte m = Byte.parseByte(month);
        byte d = Byte.parseByte(day);
        byte[] days = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

        if (d >= 1) {
            if (leapYear && m == 2) {
                if (29 < d) return invalid;
            } else if (days[m - 1] < d) return invalid;
        } else return invalid;

        countIfBelow50(Integer.parseInt(day), Integer.parseInt(month),
Integer.parseInt(year));

        return day + "-" + month + "-" + year;
    }

    /**
     * see {@link #dayCorrection(String)}
     * see {@link #monthCorrection(String)}
     * see {@link #yearCorrection(String)}
     * see {@link #dateValidation(String, String, String)}
     * A function to convert the given date in string format to valid format.
     * @see #dateValidation(String, String, String)
     * @see #monthCorrection(String)
     * @see #yearCorrection(String)
     * @see #dateValidation(String, String, String)
     * @param str date in string format
     * @return converted in valid string date
     */
    private String convertToValidDate(String str) {
```

```
    if (str.contains(invalid)) return invalid;

    String day, month, year;
    int dashIndex = 0, dm = -1, my = -1;

    while ((dashIndex = str.indexOf("-", dashIndex)) != -1) {
        if (my != -1) return invalid;
        if (dm == -1) {
            dm = dashIndex;
        } else {
            my = dashIndex;
        }
        dashIndex++;
    }

    if (dm == -1 || my == -1) return invalid;

    day = dayCorrection(str.substring(0, dm));
    month = monthCorrection(str.substring(dm + 1, my));
    year = yearCorrection(str.substring(my + 1));

    if (month.equals(invalid) || year.equals(invalid)) return invalid;

    return dateValidation(day, month, year);
}

/**
 * Replace the line with new data in file.
 * @param lineNumber line number to replace
 * @param data new data to be place in given line number
 */
private void replaceLine(int lineNumber, String data) {

    try {
            Path path = Paths.get(String.valueOf(file));
            List<String> lines = Files.readAllLines(path, StandardCharsets.UTF_8);
            lines.set(lineNumber - 1, data);
            Files.write(path, lines, StandardCharsets.UTF_8);

        } catch (IOException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    /**
     * see {@link #replaceLine(int, String)}
     * see {@link #convertToValidDate(String)}
     * A function that read the file data.
     * replace the date with the help convertToValidDate() and replaceLine methods
     * @see #replaceLine(int, String)
     * @see #convertToValidDate(String)
     */
    public void doCorrection() {

        try {

            int lineNumber = 0;
            sc = new Scanner(file);
            while (sc.hasNextLine()) {

                lineNumber++;
                String readDate = sc.nextLine();
                if (readDate.contains(invalid)) continue;
                String finalDate = convertToValidDate(readDate);

                if (!readDate.equals(finalDate)) {

                    if (finalDate.equals(invalid)) replaceLine(lineNumber, readDate + " | " +
finalDate);
```

```
                else replaceLine(lineNumber, finalDate);
            }
        }

        sc.close();
    } catch (Exception e) {
        System.out.println("ErrorR: " + e.getMessage());
    }
  }
}

class Q5 {

  public static void main(String[] args) {

    String fileName;
    Scanner sc = new Scanner(System.in);

    System.out.print("Enter DOB file name: ");
    fileName = sc.nextLine();
    //fileName = "DOBs.txt";

    XYZCompany dataEngineer = new XYZCompany(fileName);

    System.out.println("\nCurrent Dates in file: " + fileName);
    dataEngineer.displayFileData();

    dataEngineer.doCorrection();

    System.out.println("\nUpdated Dates in file: " + fileName);
    dataEngineer.displayFileData();

    System.out.println("\nBelow 50: " + dataEngineer.getBelow50());
  }
}
```

Output #5:



```java
        } catch (Exception e) {
            System.out.println("ErrorR: " + e.getMessage());
        }
    }
}

class Q5 {

    public static void main(String[] args) {

        String fileName;
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter DOB file name: ");
        fileName = sc.nextLine();
        //fileName = "DOBs.txt";

        XYZCompany dataEngineer = new XYZCompany(fileName);

        System.out.println("\nCurrent Dates in file: " + fileName);
        dataEngineer.displayFileData();

        dataEngineer.doCorrection();

        System.out.println("\nUpdated Dates in file: " + fileName);
        dataEngineer.displayFileData();

        System.out.println("\nBelow 50: " + dataEngineer.getBelow50());
    }
```

```
"C:\Program Files\Java\jdk1.8.0_241\bin\java.exe" ...
Enter DOB file name: DOBs.txt

Current Dates in file: DOBs.txt
10-10-1985
05-12-1997
01-jan-1998
30-may-1972
5-10-1997

Updated Dates in file: DOBs.txt
10-10-1985
05-12-1997
01-01-1998
30-05-1972
05-10-1997

Below 50: 4

Process finished with exit code 0
```

Question #6:

Yashwant has string str that is used to build magical string magic by repeating the string str infinitely many times. For example, if str='xxyyy', then magic='xxyyyxxyyyxxyyy...' . One of Yashwant's friends, Henry always boasts about his sharp memory and claims that he is very good at counting. To test Henry's ability, Yashwant decided to hold a test for Henry, to check the reality of his claims.

Yashwant will give Henry q queries, such that each query consists of two integers l and r, and a lowercase English letter ch. The answer to each query is how many times the letter c appears between the lth and rth letters in string magic.

Henry must answer all the queries correctly, in order to prove his claims. Can you help Henry by answering all queries?

Example Input

str: abcabdca

l: 1

r: 8

ch: c

Example Output

2

You have to show the output for the given inputs

str: abcabdca

Query #1:

l: 1 ,

r: 8 ,

ch: c

Query #2:
l: 1 ,
r: 15 ,
ch: b

Query #3:
l: 4 ,
r: 9 ,
ch: a

Query #4:
l: 5 ,
r: 25 ,
ch: d

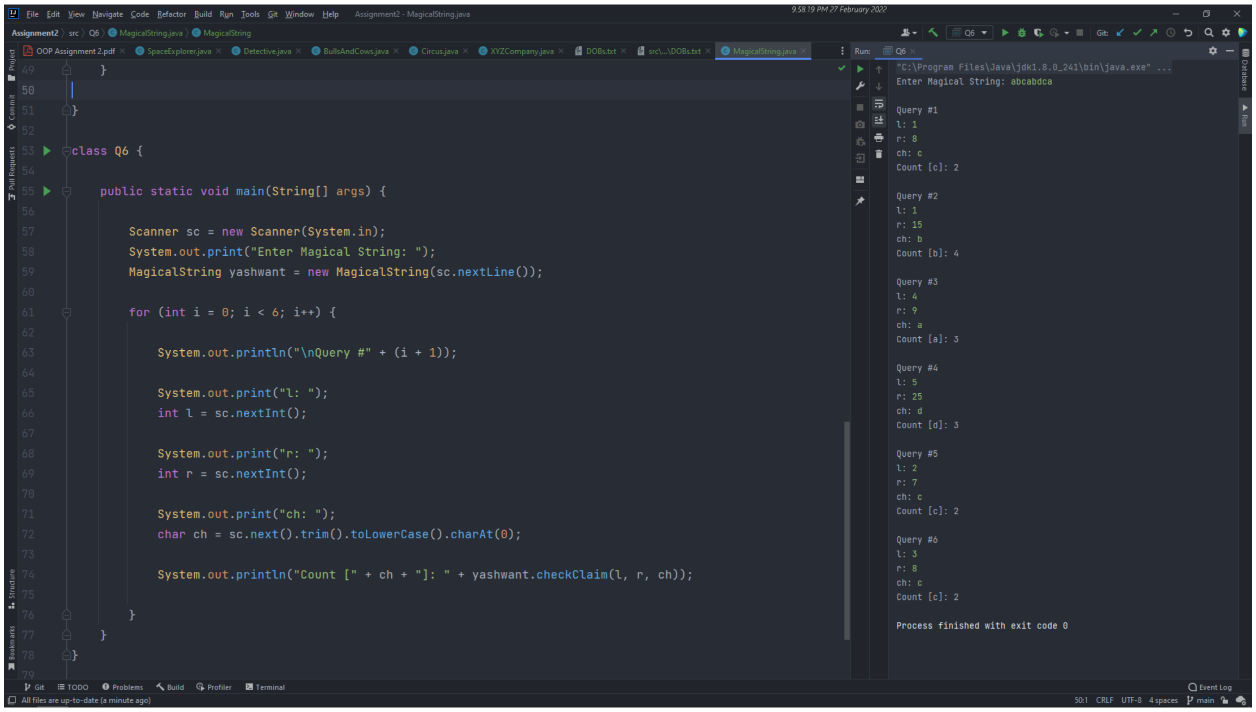
Query #5:
l: 2 ,
r: 7 ,
ch: c

Query #6:
l: 3 ,
r: 8 ,
ch: c

Code #6:

```
package Q6;
import java.util.Scanner;

/**
 * Build magical string magic by repeating the string str infinitely many times.
 * For example, if str = ’xxyyy’,then magic = ’xxyyyxxyyyxxyyy...’.
 * @version 22.2.26
 * @author SaGaR VaRiyA
 */
public class MagicalString {

    /**
     * using magicalStr to store original string.
     */
    private final String magicalStr;

    /**
     * Construct MagicalString class with the initialization of original string.
     * @param magicalStr original string.
     */
    public MagicalString(String magicalStr) {
        this.magicalStr = magicalStr;
    }

    /**
     * A function to get the character count in magical string
     * in specific sub string of magical string.
     * @param l Starting index
     * @param r Ending index
     * @param ch Character to be found
     * @return Appearance count of given character in magical string
     */
    public int checkClaim(int l, int r, char ch) {

        int left = l - 1, right = r - 1, countChar = 0;
        int iterations = right / magicalStr.length();
        StringBuilder tempMagicalStr = new StringBuilder(magicalStr);

        while (iterations-- > 0) {
            tempMagicalStr.append(magicalStr);
        }

        while ((left = tempMagicalStr.toString().indexOf(ch, left)) != -1) {
            if (left++ > right) break;
            countChar++;
        }
        return countChar;
    }
}

class Q6 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        System.out.print("Enter Magical String: ");
        MagicalString yashwant = new MagicalString(sc.nextLine());

        for (int i = 0; i < 6; i++) {

            System.out.println("\nQuery #" + (i + 1));

            System.out.print("l: ");
            int l = sc.nextInt();

            System.out.print("r: ");
            int r = sc.nextInt();
            System.out.print("ch: ");
            char ch = sc.next().trim().toLowerCase().charAt(0);

            System.out.println("Count [" + ch + "]: " + yashwant.checkClaim(l, r, ch));

        }
    }
}
```

Output #6:



```java
    }

}

class Q6 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        System.out.print("Enter Magical String: ");
        MagicalString yashwant = new MagicalString(sc.nextLine());

        for (int i = 0; i < 6; i++) {

            System.out.println("\nQuery #" + (i + 1));

            System.out.print("l: ");
            int l = sc.nextInt();

            System.out.print("r: ");
            int r = sc.nextInt();

            System.out.print("ch: ");
            char ch = sc.next().trim().toLowerCase().charAt(0);

            System.out.println("Count [" + ch + "]: " + yashwant.checkClaim(l, r, ch));

        }
    }
}
```

```
"C:\Program Files\Java\jdk1.8.0_241\bin\java.exe" ...
Enter Magical String: abcabdca

Query #1
l: 1
r: 8
ch: c
Count [c]: 2

Query #2
l: 1
r: 15
ch: b
Count [b]: 4

Query #3
l: 4
r: 9
ch: a
Count [a]: 3

Query #4
l: 5
r: 25
ch: d
Count [d]: 3

Query #5
l: 2
r: 7
ch: c
Count [c]: 2

Query #6
l: 3
r: 8
ch: c
Count [c]: 2

Process finished with exit code 0
```