implement of a java program that generates a random string of a given length and finds different special patterns in the string using the concept of “Exception Handling” in Java.

1 Flow of the Program

1.1 Input from System.in

At the beginning, the program asks user to enter the following two quantities via keyboard:

- Length of the random string that will be used in the rest of the program (it is an integer in the legal range of 100 thousand (inclusive) to 1 billion (inclusive)),
- Maximum length of the special patterns that the program looks for (it is an integer in the legal range of 3 (inclusive) to 15 (inclusive))

If the user does not enter each of the above values within the legal range, the main method needs to throw a NumberFormatException in a try block. The exception must be caught in the main method by requesting the the user to try again (see the starter code posted on Canvas for more details).

1.2 Random String Generation

In the second step, the program uses *java.util.Random.nextInt(int bound)* method to generate a random string made of a sequence of randomly generated lowercase letters. The above

```java
private static String randomStringGenerator(int length) {
    Random random = new Random(System.nanoTime());
    char[] array = new char[length];
    for (int i = 0; i < length; i++)
        array[i] = (char) ('a' + random.nextInt(26));
    return new String(array);
}
```

method, which is called by the main method, generates a random string with a given length. This string is only made of lowercase letters in the alphabet.

1.3 Finding Special Patterns

In the third step, the program uses "Exception Handling" to find the longest most-interesting/special pattern in the generated random string. The program is interested in finding one of the following special patterns:

I. Singleton string: A singleton string is made of only one letter. Examples: mmmmm, qqqqqqq, rr, s, yyy

II. Arithmetic String of Order 1: A string made of subsequent alphabetical letters that appear in the alphabetical order. Examples: bcdef, pqrstuvwx, jk, y

III. Arithmetic String of Order -1: A string made of subsequent alphabetical letters that appear in the reverse alphabetical order. Example: fedcb, xwvutsrqp, kj, y

IV. Balanced Tripartite String: A string made of three identical parts. Example: busbusbus, laptoplaptoplaptop, zzz

V. Balanced Bipartite String: A string made of two identical halves. Examples: ticktick, hophop, tantan, nocknock, nn

VI. Palindrome: A palindrome reads the same backward as forward. Examples: abcba, bob, g

Please note that the above list is ranked in the decreasing order of their rarity. If there are two interesting patterns of the same length in the random string, the program needs to report the more infrequent one!

For example, consider the following random string:

$$thisisarandomstringbobbobbobendofthestring$$

In this string, there is a palindrome of length 9 (bobbobbob) and a balanced tripartitie of the same length (bobbobbob). The program reports the latter one as it is higher in the rank of special patterns.

As another example, consider the following random string:

$$yyyyyyyyycdefghijkl$$

In this string, there is a sinleton string of length 9 (yyyyyyyyy) and an arithmetic string of order 1 with length 10 (cdefghijkl). The program reports the latter one as it is longer. When reporting the special pattern, the program needs to specify the actual pattern (cdefghijkl) and the index of its occurrence (9).

2 Required Exception Classes

To implement the third step of the program, you need to first define one exception class for each special pattern and then throw an instance of it once you find the pattern in the random string. For example, the following exception is defined to handle singleton strings: An instance of this exception is constructed and thrown once it is found in the random

```java
public class SingletonException extends Exception {
    private String singletonString;
    private int occurrenceIndex;
    @Override
    public String getMessage() {
        return singletonString + " is a singleton string that is found at index " + occurrenceIndex + "!";
    }
    public SingletonException(String singletonString, int index) {
        this.singletonString = singletonString;
        occurrenceIndex = index;
    }
}
```

string. The following static method is responsible for finding the singleton patterns: The

```java
private static void singletonMiner(String mine, int length) throws SingletonException{
    for (int start = 0; start < mine.length() - length; start++) {
        int i;
        for (i = start + 1; i < start + length; i++)
            if (mine.charAt(i) != mine.charAt(i - 1))
                break;
        if (i == start + length)
            throw new SingletonException(mine.substring(start, start + length), start);
    }
}
```

exception thrown by the method *singletonMiner* is caught by the following try-catch block that appears in the main method:

```java
patternMaxLength = 5;//the max length of pattern is given by user via keyboard!
String randomString = randomStringGenerator(randomStringLength);
try {
    for (int length = patternMaxLength; length > 0; length--)
        singletonMiner(randomString, length);
} catch (Exception exp) {
    System.out.println(exp.getMessage());
}
```

3 Submissions

You need to submit a *.zip* file compressing the packages containing all the java source files related to the assignment (.java files).