COL 106 : Data Structures and Algorithms
Semester II, 2022-23, IIT Delhi

Assignment - 1 (due on 15th January, 11:00 PM)

Important Guidelines:

- You must ensure that your submitted code runs in the JDK 11.0.17 environment.
- You are not allowed to share your code with anyone. Cheating of any form will lead to strict disciplinary action. Typical penalty would include Fail grade in the course.
- The assignment must be done individually, and no group submissions are allowed.
- All four starter code files must be included in the submission, which must be uploaded on the gradescope without compressing the files.
- The names of files and method signatures must not be changed in the starter code.
- You should write the code in the portion of the starter code labelled **"To be filled in by the student"**.

1 Frequency of letters

Write a program that given a text file returns an array of size 26 which stores the number of occurrences of each letter (i.e. a-z) in the file. If there is no file with the given name, then the program should return the appropriate exception(FileNotFoundException).

Starter code is present in file **FreqOfLetters.java**.

```
Input Format : You will be provided with the file name.
File will only contain lower case English alphabets.
Ignore space and end of line characters.
```

```
Input 1 :
"dummy.txt"
```

```
Return 1 :
{ 29, 3, 16, 19, 38, 3, 3, 1, 42, 0, 0, 22, 17, 24, 29, 11, 5,
   22, 18, 31, 29, 3, 0, 3, 0, 0 }
```

```
Input 2 :
"dummy1.txt" (File not present)
```

```
Return1 :
FileNotFoundException
```

2 Method Overloading

Write a java class named ”MethodOverloading” with the following three methods with same name but different number of parameters:

- public double calculate(int a)
  Returns area of a square with side a.
- public double calculate(int a, int b)
  Returns area of a rectangle with sides a, b.
- public double calculate(int a, int b, int c)
  Returns area of a triangle with sides a, b ,c.

Starter code is present in **MethodOverloading.java**.

```
Input Format : We will call the functions with 1,2 or 3 arguments.
```

```
Input 1:
calculate(2)
```

```
Return 1:
4
```

```
Input 2:
calculate(2,3)
```

```
Return 2:
6
```

```
Input 3:
calculate(3,4,5)
```

```
Return 3:
6
```

3 Stack implementation

Implement the following methods for a stack using dynamic arrays (covered in lecture 3).

- public void push(Character i);
- public Character pop();
- public Boolean is_empty();
- public Integer size();
- public Character top();
- public Character[] return_base_array();

Starter code is present in **DemoStack.java**.

```
Input Format : We will call the functions which you implement.
If the stack is empty, then call to pop and top should throw
   EmptyStackException.
```

```
Input 1:
push('[')
push('{')
push('(')
pop()
push(']')
pop()
size()
top()
pop()
pop()
isEmpty()
```

```
Returns 1:
null
null
null
'('
null
']'
2
'{'
'{'
'['
True
```

```
Input 2:
push('[')
push('{')
size()
top()
pop()
pop()
pop()
```

```
Returns 2:
null
null
2
'{'
'{'
'['
EmptyStackException
```

4 Parenthesis Matching

Write a java program that takes from user a string comprising of only the characters '(', '{', '[', ')', '}', and ']', and checks if the parentheses in the string are balanced. Your program must use the stack data-structure from Question 3.

Starter code is present in **ParenthesisMatching.java**.

```
Input Format : String will be given as an input to the function.
String comprises only the characters '(', '{', '[', ')', '}', and
   ']'
```

```
Input 1:
"{[()][]()}"
```

```
Return 1:
True
```

```
Input 2:
"{[(])}"
```

```
Returns 2:
False
```