CSE3040: Java Programming (Fall 2020)
Midterm Project 1
Due: November 17, 11:59PM (KST)

※ Write a Java code for each problem. Make sure your program satisfies all specified requirements. When an example result is given, output of your program should match the result.

Problem 11 (20 points)

• A palindrome is a word, number, phrase, or other sequence of characters which reads the same backward as forward, such as madam and racecar (*from wikipedia*). In this problem, you will write a program that checks whether a given string or integer is a palindrome or not.

• You are given the code for class Problem11, which must NOT be modified. Write other parts of the code to complete the program.

```
public class Problem11 {
    public static void main(String[] args) {
        PalindromeChecker.check("abcde");
        PalindromeChecker.check("abcba");
        PalindromeChecker.check(1234);
        PalindromeChecker.check(12321);
    }
}
```

• When we run the program, the result printed on the screen should be:

```
abcde is not a palindrome.
abcba is a palindrome.
1234 is not a palindrome.
12321 is a palindrome.
```

• If the given string is a palindrome (for example, "abcba"), the program should print "abcba is a palindrome". If not (for example, "abcde"), it should print "abcde is not a palindrome."

• We will change the argument to the method `PalindromeChecker.check` when we evaluate your code. The argument will be either a String type or an int type.

• If the argument is an integer, assume that the given integer is a non-negative integer. If the argument is a string, assume that all characters are in lowercase alphabets.

• If you define a new class, all instance variables must be declared as private, if there is any.

• You must not use class `StringBuffer` in your code for this problem. This class name must not appear in your code.

Problem 12 (20 points)

• A subsequence of a string is a string that is generated by deleting some character of a given string without changing its order. For example, if the given string is "abcd", the subsequences of this string are the following strings.

a, b, c, d, ab, ac, ad, bc, bd, cd, abc, abd, acd, bcd, abcd

• In this problem, you will write a program that checks whether a given string is a subsequence of another given string.

• You are given the code for class Problem12, which must NOT be modified. Write other parts of the code to complete the program.

```
public class Problem12 {
  public static void main(String[] args) {
    SubsequenceChecker.check("supercalifragilisticexpialidocious", "pads");
    SubsequenceChecker.check("supercalifragilisticexpialidocious", "padsx");
  }
}
```

• When we run the program, the result printed on the screen should be:

```
pads is a subsequence of supercalifragilisticexpialidocious
2 6 27 33
padsx is not a subsequence of supercalifragilisticexpialidocious
```

• If the second string (str2) is a subsequence of the first string (str1), then the program should first print a line that says "str2 is a subsequence of str1", where str1 is the first string and str2 is the second string.

• Then, on the next line, the program should print the index numbers of characters in str1 that are selected in the subsequence. In the example above, characters at index 2, 6, 27, 33 are "p", "a", "d", "s", which makes the string str2. If multiple instances of subsequence can be found, the program should print the first instance which has the smallest index number.

• If str2 is not a subsequence of str1, the program just prints one line saying "str2 is not a subsequence of str1".

• When we evaluate your code, we will change the strings given to the method `SubsequenceChecker.check`. Other parts of class Problem12 will not be modified.

• If you define a new class, all instance variables must be declared as private, if there is any.

Problem 13 (20 points)

• In this problem, you will write a program that reads text from a file and counts number of appearances for each alphabet character (a-z).

• You are given the code for class Problem13, which must NOT be modified. Write other parts of the code to complete the program.

```
public class Problem13 {
    public static void main(String[] args) {
        Text t = new Text();
        if(t.readTextFromFile("input_prob13.txt")) {
            for(char c = 'a'; c <= 'z'; c++) {
                System.out.println(c + ": " + t.countChar(c));
            }
        }
    }
}
```

• For example, suppose the content of the input file was as follows.

Java is a general-purpose programming language that is class-based, object-oriented (although not a pure object-oriented language, as it contains primitive types), and designed to have as few implementation dependencies as possible. It is intended to let application developers write once, run anywhere (WORA) meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of the underlying computer architecture. The syntax of Java is similar to C and C++, but it has fewer low-level facilities than either of them. As of 2019, Java was one of the most popular programming languages in use according to GitHub, particularly for client-server web applications, with a reported 9 million developers.

• When we run the program with this input file, the output should be as follows.

```
a: 73
b: 8
c: 28
d: 22
e: 77
f: 11
g: 18
h: 21
i: 53
j: 10
k: 0
l: 36
m: 19
n: 47
o: 50
p: 30
q: 0
```

```
r: 42
s: 35
t: 60
u: 20
v: 15
w: 10
x: 1
y: 9
z: 0
```

• When counting the characters, the program should ignore the letter case. In other words, 'A' and 'a' are all counted as 'a's. In the text, 'A' and 'a' appear 73 times in total.

• You do not need to count characters other than [a-z], such as numbers or special characters.

• In the code you write, a method must never throw exceptions. For checked exceptions, you should handle all of them using try-catch statements.

• If the input file is not found, your program should print "`Input file not found.`" and terminate. You should NOT make your program crash due to run-time errors.

• In the evaluation process, we will use a different input file to check the correctness of your program.

• If you define a new class, all instance variables must be declared as private, if there is any.

Problem 14 (20 points)

• In this problem, you will write a program that reads data from a file and print out statistics on the data.

• The input file is a list of items and their prices. Each line of the file consists of a string and a floating-point number. The string corresponds to the item name, and the floating-point number is its price. An example input file looks like this:

```
apple 3.99
watermelon 11.99
strawberries 4.99
pear 6.99
mango 8.99
bananas 4.99
pineapples 7.99
kiwi 5.99
raspberries 2.99
peaches 10.99
```

• You can assume that no two items have the same name, and there are no two items that have the same price.

• After reading the file, more items could be added from the main method.

• You are given the code for class Problem14, which must NOT be modified. Write other parts of the code to complete the program.

```
public class Problem14 {
  public static void main(String[] args) {
    FruitBox<Fruit> box = new FruitBox<>();
    boolean rv = ItemReader.fileToBox("input_prob14.txt", box);
    if(rv == false) return;
    box.add(new Fruit("orange", 9.99));
    System.out.println("----------------");
    System.out.println("    Summary");
    System.out.println("----------------");
    System.out.println("number of items: " + box.getNumItems());
    System.out.println("most expensive item: " + box.getMaxItem() + " (" +
                       box.getMaxPrice() + ")");
    System.out.println("cheapest item: " + box.getMinItem() + " (" +
                       box.getMinPrice() + ")");
    System.out.printf("average price of items: %.2f", box.getAvgPrice());}
}
```

• As you can see in the example main method, items could be added to the box by reading data from file (`ItemReader.fileToBox`), or added to the box by calling method `add`.

• For the input file shown above, the output of your program should be as follows. The program first prints all items and their prices on the screen, and then show statistics of data such as number of items, most expensive item, cheapest item, and average price of items. Your program should match the result shown here.

```
apple 3.99
watermelon 11.99
strawberries 4.99
pear 6.99
mango 8.99
bananas 4.99
pineapples 7.99
kiwi 5.99
raspberries 2.99
peaches 10.99
orange 9.99
----------------
    Summary
----------------
number of items: 11
most expensive item: watermelon (11.99)
cheapest item: raspberries (2.99)
average price of items: 7.26
```

• Assume that if the input file exists, the format will be always correct: item name and price on each line. You do not need to worry about handling wrong input format.

• In the code you write, a method must never throw exceptions. For checked exceptions, you should handle all of them using try-catch statements.

• If the input file is not found, your program should print "`Input file not found.`" and terminate. You should NOT make your program crash due to run-time errors.

• In the evaluation process, we will use a different input file to check the correctness of your program.

• If you define a new class, all instance variables must be declared as private, if there is any.

Problem 15 (20 points)

• In this problem, you will write a program that reads text from a file and print out appearances of each word in the text.

• You are given the code for class Problem15, which must NOT be modified. Write other parts of the code to complete the program.

```
public class Problem15 {
  public static void main(String[] args) {
    ArrayList<Item> list = new ArrayList<>();
    boolean rv = MyFileReader.readDataFromFile("input_prob15.txt", list);
    if(rv == false) {
      System.out.println("Input file not found.");
      return;
    }
    for(Item it: list) System.out.println(it);
  }
}
```

• Suppose the text in the input file is as follows.

Java is an island of Indonesia, bordered by the Indian Ocean on the south and the Java Sea on the north. With a population of over 141 million (Java only) or 145 million (including the inhabitants of its surrounding islands), Java constitutes 56.7 percent of the Indonesian population and is the world's most-populous island. The Indonesian capital city, Jakarta, is on its northwestern coast. Much of the well-known part of Indonesian history took place on Java. It was the centre of powerful Hindu-Buddhist empires, the Islamic sultanates, and the core of the colonial Dutch East Indies. Java was also the center of the Indonesian struggle for independence during the 1930s and 1940s. Java dominates Indonesia politically, economically and culturally. Four of Indonesia's eight UNESCO world heritage sites are located in Java: Ujung Kulon National Park, Borobudur Temple, Prambanan Temple, and Sangiran Early Man Site.

• Then, your program should output the following result.

```
java 5
is 3
an 1
island 1
of 10
indonesia, 1
bordered 1
by 1
the 16
indian 1
ocean 1
on 4
south 1
```

```
and 6
sea 1
north. 1
with 1
a 1
population 2
over 1
141 1
million 2
(java 1
only) 1
or 1
145 1
(including 1
inhabitants 1
its 2
surrounding 1
islands), 1
constitutes 1
56.7 1
percent 1
indonesian 4
world's 1
most-populous 1
island. 1
capital 1
city, 1
jakarta, 1
northwestern 1
coast. 1
much 1
well-known 1
part 1
history 1
took 1
place 1
java. 1
it 1
was 2
centre 1
powerful 1
hindu-buddhist 1
empires, 1
islamic 1
sultanates, 1
core 1
colonial 1
dutch 1
east 1
indies. 1
also 1
center 1
struggle 1
for 1
independence 1
during 1
1930s 1
1940s. 1
```

```
dominates 1
indonesia 1
politically, 1
economically 1
culturally. 1
four 1
indonesia's 1
eight 1
unesco 1
world 1
heritage 1
sites 1
are 1
located 1
in 1
java: 1
ujung 1
kulon 1
national 1
park, 1
borobudur 1
temple, 2
prambanan 1
sangiran 1
early 1
man 1
site. 1
```

• Here, we will just define a "word" as a token that is separated by a space. For example, if a part of the text reads:

`(Java only)`

Then we have two words: `(Java` and `only)`

The word "only" and "only)" are treated as two different words. Similarly, "java", "java,", "java.", "(java" are all different words.

• When processing the text, all uppercase letters should be converted to lowercase letters, in order to ignore letter case.

• Each word in the output should have at least one letter. A word with no letter should not appear in the output.

• The words in the output are sorted by their first appearance in the text. A word found earlier in the text will appear earlier in the output.

• In the code you write, a method must never throw exceptions. For checked exceptions, you should handle all of them using try-catch statements.

• If the input file is not found, your program should print "`Input file not found.`" and

terminate. You should NOT make your program crash due to run-time errors.

• In the evaluation process, we will use a different input file to check the correctness of your program.

• If you define a new class, all instance variables must be declared as private, if there is any.

※ Submission Instructions

※ Carefully read this part and follow the instructions. Not properly following the instructions here may result in point deduction.

(1) For this homework, you are going to submit only the .java files. You are going to submit one .java file for each problem.

(2) For problem 11, your file name should be Problem11.java. This means that your public class name should also be Problem11. For other problems, you should name your .java files this way.

(3) Once you are ready to submit, you should make the .java files into a single zip file named cse3040_mp1_20180001.zip. The numbers in the file name should be your student ID.

When you make the zip file, make sure that you are not using subfolders inside the zip file. When the zip file is extracted, the .java files should appear without having any subdirectory structures.

(4) Submit your zip file on the cyber campus.

※ Evaluation Criteria

Your solution to each problem will be tested with various test cases. You will get full points if your program passes all tests. If not, points will be given based on percentage of test cases passed.

For this homework, each problem is worth 20 points. The perfect score is 100.

※ Academic Integrity

- You should write your own code. You can discuss ideas with other students but must not copy their work. You can also get help from the Internet, but you must not copy the source code from the Internet either. We have a duplicate check program which tests whether your source code is similar to other students' code as well as codes that are on the Internet.

- Duplicate work will receive zero grade.

※ Late Policy

- 10% of the score is deducted for each day, up to three days. Submissions are accepted up to three days after the deadline.