Concordia University
Comp 249 – Summer 2020
Object-Oriented Programming II - Assignment 3
Due 11:59 PM- Sunday, August 9, 2020

Purpose: The purpose of this assignment is to allow you practice File I/O, ArrayList, and Linked Lists.

Part 1: ArrayList & File I/O

Sub-Dictionary Creator

In this part, you are required to write a program that will accept any text file, as input, and creates a sub-dictionary that includes all the words found in that input file based on some rules. The input file may have zero or more words, as well as specific limited set of other characters /punctuation that are used in a specific and predetermined manner. In particular, the text on the input file is only assumed to have the following (besides the words of course!):

- "**?**": only appears at the end of a word; for instance: why not**?**
- "**:**": only appears at the end of a word; for instance: the rules are**:** ….
- "**'**": which can only appear in front of m or s; for instance: I'**m** … or It'**s** …
- "**,**": only appears at the end of a word; for instance: However**,** ….
- "**=**": which can appear by itself in the middle of an equation; for instance: x **=** y
- "**;**": only appears at the end of a word; for instance: violin**;** what else …
- "**!**": only appears at the end of a word; for instance: That is fantastic**!**
- "**.**": only appears at the end of a word; for instance: These times were good**.**
- Digits: 0 to 9, which may appear as a number (i.e. ***1927***); or as a part of a word (i.e. ***hi5***)
- Single characters: ***A***, ***B***, ***T***, etc.

You are required to implement a program that will take any such text file as input, and create a sub-dictionary of the words on that file, based on the following rules:

- For each word, only one entry can be recorded in the sub-dictionary. For instance, if the word "Hello" appeared in the text 15 times, it is still recorded once in the sub-dictionary.
- All words must be recorded only in UPPERCASE. For instance, Hello must be recorded as HELLO.
- Words cannot be recorded with any of the punctuation; for instance, "fantastic!" must be recorded only as "fantastic".
- No numbers or words that have digits anywhere (i.e. 1927, hi5 or b4that) can be recorded in the sub-dictionary.
- No single characters (i.e. k, M, t, etc.), with the exception of A and I, can be recorded in the dictionary. (Notice that "*a*" and "*i*" are allowed but need to be recorded as *A* and *I* respectively).
- All words with "'s" or "'*m*" (or their upper case versions) must be recorded without the 's or '*m*. For instance, *It's*, will need to be recorded as *IT*.

- All words are recorded in the sub-dictionary in the usual sorted alphabetic order. Additionally, each group must be preceded with an indication of the character that starts this group (similar to real-life dictionaries). For instance, all words starting with "K", will be preceded with something like:

  K

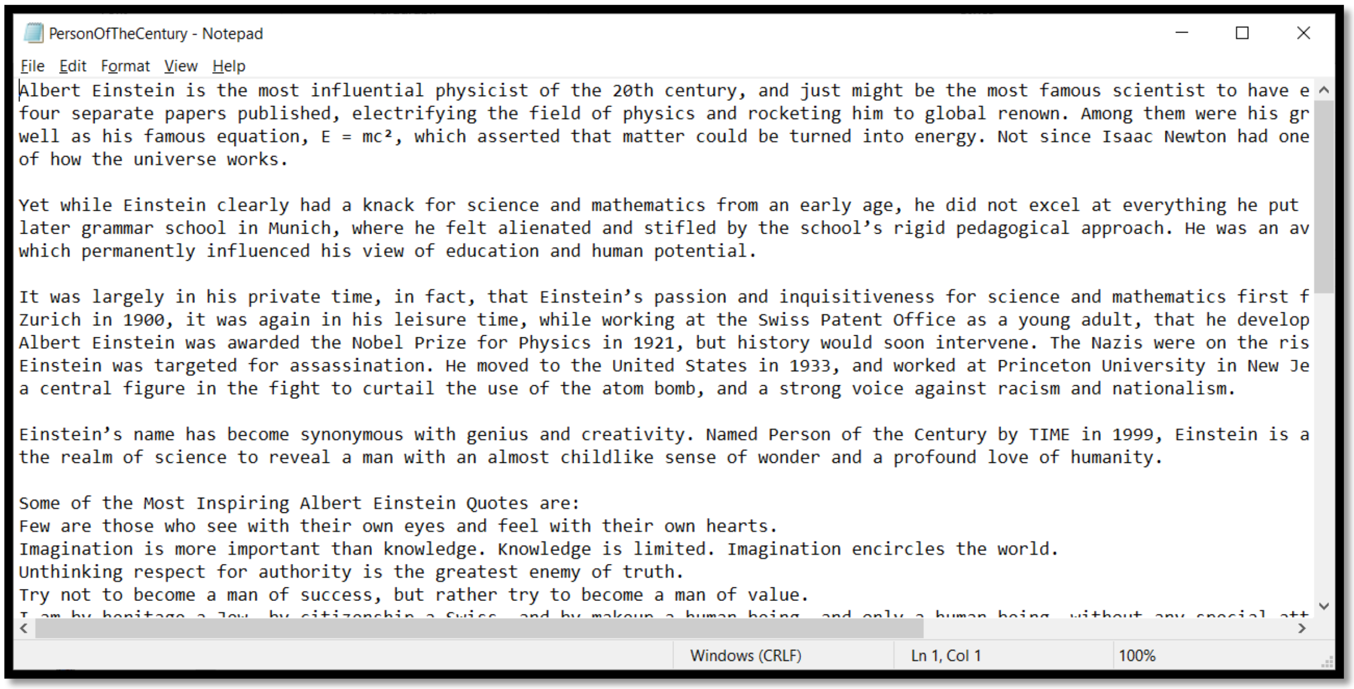
  = =

- Finally, the sub-dictionary must have an initial line indicating its size based on the given input file. For instance:

  The document produced this sub-dictionary, which includes 447 entries.

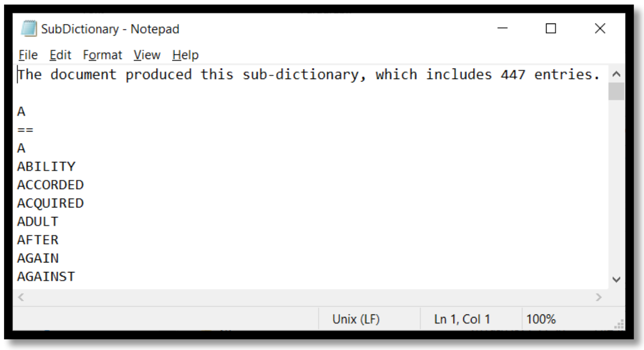
For instance, given the following file PersonOfTheCentury.txt[1] (which is provided with the assignment), your program will create the output file SubDictionary.txt (which is also provided with the assignment). You should notice that this input file is just one example and your program must be able to work with any input file. In fact, the marker will actually test your program with other input files. For immediate illustration, partial images of the input and output files are shown below.

[1] File is extracted from text obtained from: www.goalcast.com



```
Albert Einstein is the most influential physicist of the 20th century, and just might be the most famous scientist to have e
four separate papers published, electrifying the field of physics and rocketing him to global renown. Among them were his gr
well as his famous equation, E = mc², which asserted that matter could be turned into energy. Not since Isaac Newton had one
of how the universe works.

Yet while Einstein clearly had a knack for science and mathematics from an early age, he did not excel at everything he put
later grammar school in Munich, where he felt alienated and stifled by the school's rigid pedagogical approach. He was an av
which permanently influenced his view of education and human potential.

It was largely in his private time, in fact, that Einstein's passion and inquisitiveness for science and mathematics first f
Zurich in 1900, it was again in his leisure time, while working at the Swiss Patent Office as a young adult, that he develop
Albert Einstein was awarded the Nobel Prize for Physics in 1921, but history would soon intervene. The Nazis were on the ris
Einstein was targeted for assassination. He moved to the United States in 1933, and worked at Princeton University in New Je
a central figure in the fight to curtail the use of the atom bomb, and a strong voice against racism and nationalism.

Einstein's name has become synonymous with genius and creativity. Named Person of the Century by TIME in 1999, Einstein is a
the realm of science to reveal a man with an almost childlike sense of wonder and a profound love of humanity.

Some of the Most Inspiring Albert Einstein Quotes are:
Few are those who see with their own eyes and feel with their own hearts.
Imagination is more important than knowledge. Knowledge is limited. Imagination encircles the world.
Unthinking respect for authority is the greatest enemy of truth.
Try not to become a man of success, but rather try to become a man of value.
```

Figure 1: Partial view of the input file PersonOfTheCentury.txt

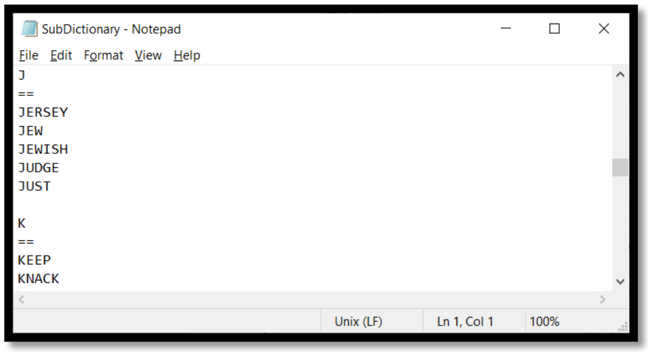


SubDictionary - Notepad

File Edit Format View Help

```
The document produced this sub-dictionary, which includes 447 entries.

A
==
A
ABILITY
ACCORDED
ACQUIRED
ADULT
AFTER
AGAIN
AGAINST
```

Unix (LF) Ln 1, Col 1 100%



SubDictionary - Notepad

File Edit Format View Help

```
J
==
JERSEY
JEW
JEWISH
JUDGE
JUST

K
==
KEEP
KNACK
```

Unix (LF) Ln 1, Col 1 100%

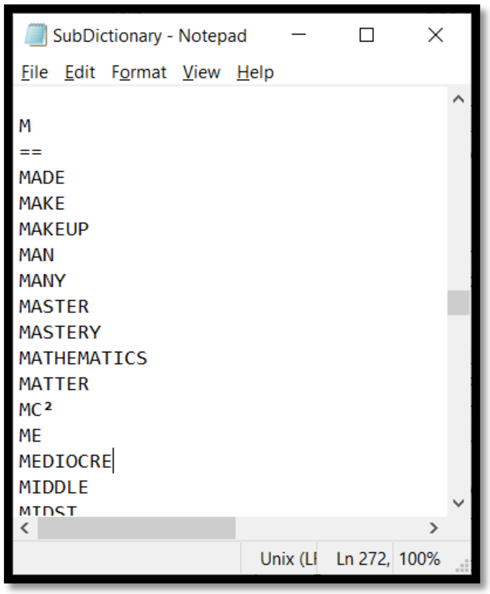
Figure 2: Partial views of the output file Sub-Dictionary.txt

Here are the most important technical rules that you need to follow:

- You **MUST** use the **ArrayList** class to implement what is needed. In specific, when you read the input file, all data must be stored in one, or more, ArrayList objects before finally being stored in the output file.
- Your program should allow the user to enter the name of the input file at run-time. The name of the output files is always assumed to be SubDictionary.txt.
- You **MUST NOT** use any other built-in Java classes/packages to do what is needed. For instance, you are **NOT** allowed to use other classes such as List, Map, HashMap, etc. In fact, here are all the needed imports for your program:

```
import java.util.ArrayList;
import java.util.Scanner;
import java.io.PrintWriter;
import java.io.FileOutputStream;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
```

⇨ Finally, as a side note, and in honor/respect of this great scientist, you may notice the word **MC²** in the output file, which was kept in the text and the output in purpose, in spite of the rule stating that words with digits must not be recorded!



SubDictionary - Notepad

File Edit Format View Help

```
M
==
MADE
MAKE
MAKEUP
MAN
MANY
MASTER
MASTERY
MATHEMATICS
MATTER
MC²
ME
MEDIOCRE
MIDDLE
MIDST
```

Unix (LF Ln 272, 100%

Figure 3: In honor of Albert Einstein

Part 2: Linked Lists

Cell Phones Records

In this part, you are required to write a program, using linked lists, that manipulates a set of records of cell phones and performs some operations on these records.

I) The **`CellPhone`** class has the following attributes: a `serialNum` (long type), a `brand` (String type), a `year` (int type, which indicates manufacturing year) and a `price` (double type). It is assumed that the brand name is always recorded as a single word (i.e. Motorola, SonyEricsson, Panasonic, etc.). It is also assumed that all cellular phones follow one system of assigning serial numbers, regardless of their different brands, so no two cell phones may have the same serial number.

You are required to write the implementation of the `CellPhone` class. Beside the usual mutator and accessor methods (i.e. `getPrice(), setYear()`) the class must have the following:

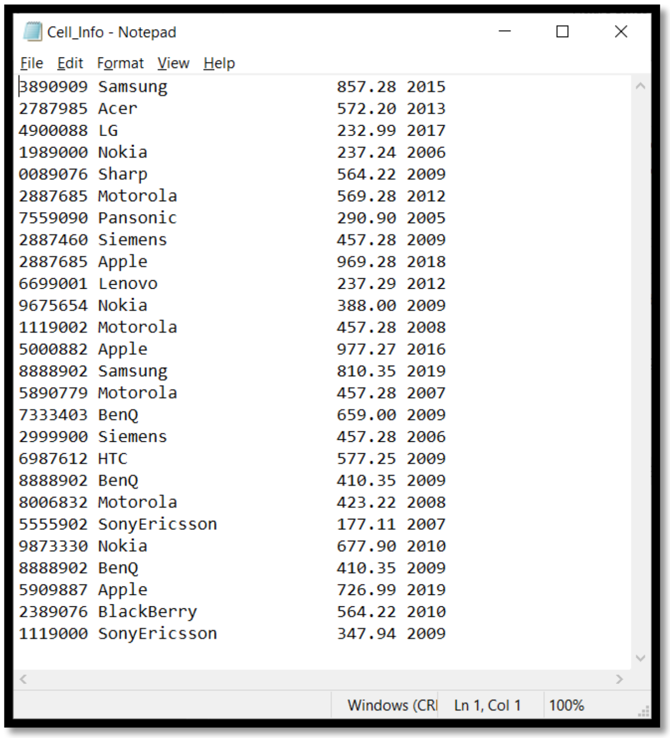
(a) Parameterized constructor that accepts four values and initializes *serialNum*, *brand*, *year* and *price* to these passed values;

(b) Copy constructor, which takes two parameters, a `CellPhone` object and a long value. The newly created object will be assigned all the attributes of the passed object, with the exception of the serial number. *serialNum* is assigned the value passed as the second parameter to the constructor. It is always assumed that this value will correspond to the unique serial number rule;

(c) `clone()` method. This method will prompt the user to enter a new serial number, then creates and returns a clone of the calling object with the exception of the serial number, which is assigned the value entered by the user;

(d) Additionally, the class should have a `toString()` and an `equals()` methods. Two cell phones are equal if they have the same attributes, with the exception of the serial number, which could be different.

II) The file **`Cell_Info.txt`**, which one of its versions is provided with this assignment, has the information on various cell phone objects. The file may have zero or more records. The information stored in this file is always assumed to be correct and following the unique serial number rule. A snapshot of the contents of the `Cell_info.txt` file is shown in Figure 1 below.



```
3890909 Samsung                 857.28 2015
2787985 Acer                    572.20 2013
4900088 LG                      232.99 2017
1989000 Nokia                   237.24 2006
0089076 Sharp                   564.22 2009
2887685 Motorola                569.28 2012
7559090 Pansonic                290.90 2005
2887460 Siemens                 457.28 2009
2887685 Apple                   969.28 2018
6699001 Lenovo                  237.29 2012
9675654 Nokia                   388.00 2009
1119002 Motorola                457.28 2008
5000882 Apple                   977.27 2016
8888902 Samsung                 810.35 2019
5890779 Motorola                457.28 2007
7333403 BenQ                    659.00 2009
2999900 Siemens                 457.28 2006
6987612 HTC                     577.25 2009
8888902 BenQ                    410.35 2009
8006832 Motorola                423.22 2008
5555902 SonyEricsson            177.11 2007
9873330 Nokia                   677.90 2010
8888902 BenQ                    410.35 2009
5909887 Apple                   726.99 2019
2389076 BlackBerry              564.22 2010
1119000 SonyEricsson            347.94 2009
```

Figure 1: Cell_info.txt

III) The **`CellList`** class has the following:

(a) An inner class called `CellNode`. This class has the following:

  i. Two private attributes: an object of `CellPhone` and a pointer to a `CellNode` object;

  ii. A default constructor, which assigns both attributes to `null`;

  iii. A parameterized constructor that accepts two parameters, a `CellPhone` object and a `CellNode` object, then initializes the attributes accordingly;

  iv. A copy constructor;

  v. A `clone()` method;

  vi. Other mutator and accessor methods.

(b) A private attribute called `head`, which should point to the first node in this list object;

(c) A private attribute called `size`, which always indicates the current size of the list (how many nodes are in the list);

(d) A default constructor, which creates an empty list;

(e) A copy constructor, which accepts a `CellList` object and creates a copy of it;

(f) A method called `addToStart()`, which accepts one parameter, an object from `CellPhone` class. The method then creates a node with that passed object and inserts this node at the head of the list;

(g) A method called `insertAtIndex()`, which accepts two parameters, an object from `CellPhone` class, and an integer representing an index. If the index is not valid (a valid index must have a value between `0` and `size-1`), the method must throw a `NoSuchElementException` and terminate the program. If the index is valid, then the method creates a node with the passed `CellPhone` object and inserts this node at the given index. The method must properly handle all special cases;

(h) A method called `deleteFromIndex()`, which accepts one integer parameter representing an index. Again, if the index is not valid, the method must throw a `NoSuchElementException` and terminate the program. Otherwise; the node pointed by that index is deleted from the list. The method must properly handle all special cases;

(i) A method called `deleteFromStart()`, which deletes the first node in the list (the one pointed by `head`). All special cases must be properly handled.

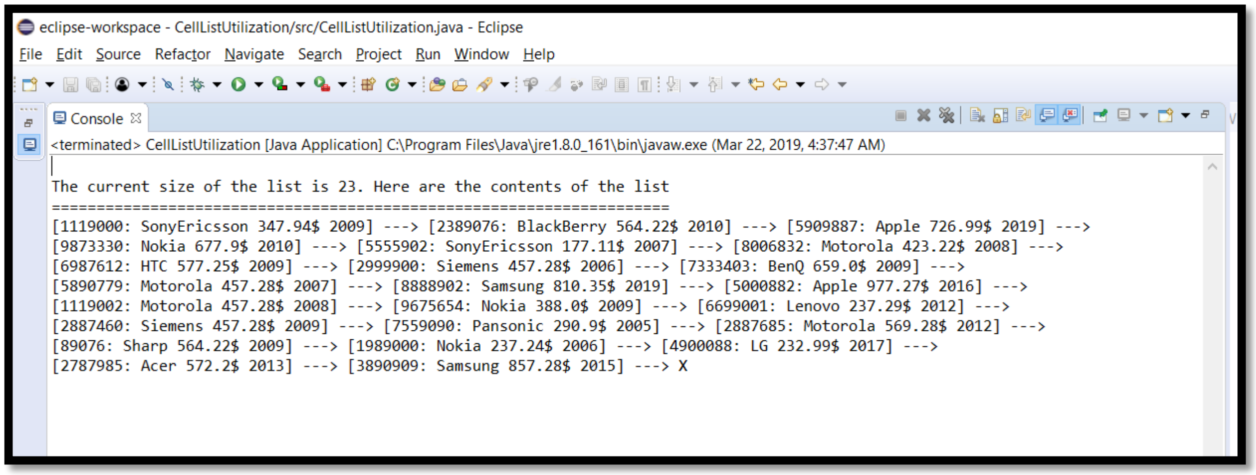
(j) A method called `replaceAtIndex()`, which accepts two parameters, an object from `CellPhone` class, and an integer representing an index. If the index is not valid, the method simply returns; otherwise, the object in the node at the passed index is to be replaced by the passed object;

(k) A method called `find()`, which accepts one parameter of type long representing a serial number. The method then searches the list for a node with a cell phone with that serial number. If such an object is found, then the method returns a pointer to that node where the object is found; otherwise, the method returns `null`. The method must keep track of how many iterations were made before the search finally finds the phone or concludes that it is not in the list;

(l) A method called `contains()`, which accepts one parameter of type long representing a serial number. The method returns `true` if an object with that serial number is in the list; otherwise, the method returns `false`;

(m) A method called `showContents()`, which displays the contents of the list, in a similar fashion to what is shown in Figure 2 below.

(n) A method called `equals()`, which accepts one parameter of type `CellList`. The method returns `true` if the two lists contain similar objects; otherwise, the method returns `false`. Recall that two `CellPhone` objects are equal if they have the same values with the exception of the serial number, which can, and actually is expected to be, different.


eclipse-workspace - CellListUtilization/src/CellListUtilization.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Console

```
<terminated> CellListUtilization [Java Application] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (Mar 22, 2019, 4:37:47 AM)

The current size of the list is 23. Here are the contents of the list
=====================================================================
[1119000: SonyEricsson 347.94$ 2009] ---> [2389076: BlackBerry 564.22$ 2010] ---> [5909887: Apple 726.99$ 2019] --->
[9873330: Nokia 677.9$ 2010] ---> [5555902: SonyEricsson 177.11$ 2007] ---> [8006832: Motorola 423.22$ 2008] --->
[6987612: HTC 577.25$ 2009] ---> [2999900: Siemens 457.28$ 2006] ---> [7333403: BenQ 659.0$ 2009] --->
[5890779: Motorola 457.28$ 2007] ---> [8888902: Samsung 810.35$ 2019] ---> [5000882: Apple 977.27$ 2016] --->
[1119002: Motorola 457.28$ 2008] ---> [9675654: Nokia 388.0$ 2009] ---> [6699001: Lenovo 237.29$ 2012] --->
[2887460: Siemens 457.28$ 2009] ---> [7559090: Pansonic 290.9$ 2005] ---> [2887685: Motorola 569.28$ 2012] --->
[89076: Sharp 564.22$ 2009] ---> [1989000: Nokia 237.24$ 2006] ---> [4900088: LG 232.99$ 2017] --->
[2787985: Acer 572.2$ 2013] ---> [3890909: Samsung 857.28$ 2015] ---> X
```

Figure 2: Sample of Displaying the Contents of a CellList

➔ Finally, here are some general rules that you must consider when implementing the above methods:

- Whenever a node is added or deleted, the list size must be adjusted accordingly;
- All special cases must be handled, whether or not the method description explicitly states that;
- All `clone()` and copy constructors must perform a deep copy; no shallow copies are allowed;
- <u>If any of your methods allow a privacy leak, you must clearly place a comment at the beginning of method 1) indicating that this method may result in a privacy leak 2) explaining the reason behind the privacy leak. Please keep in mind that you are not required to implement these proposals;</u>

IV) Now, you are required to write a public class called **`CellListUtilization`**. In the `main()` method, you must do the following:

(a) Create at least two empty lists from the `CellList` class;

(b) Open the `Cell_Info.txt` file, and read its contents line by line. Use these records to initialize one of the `CellList` objects you created above. You can simply use the `addToStart()` method to insert the read objects into the list. However, the list should not have any duplicate records, so if the input file has duplicate entries, which is the case in the file provided with the assignment, for instance, your code must handle this case so that each record is inserted in the list only once;

(c) Show the contents of the list you just initialized;

(d) Prompt the user to enter a few serial numbers and search the list that you created from the input file for these values. Make sure to display the number of iterations performed;

(e) Following that, you must create enough objects to test each of the constructors/methods of your classes. The details of this part are left as open to you. You can do whatever you wish as long as your methods are being tested including some of the special cases.

<u>General Guidelines When Writing Programs:</u>

- Include the following comments at the top of your source codes
  // ---
  // Assignment (include number)
  // Question: (include question/part number, if applicable)
  // Written by: (include your name and student id)
  // ---
- In a comment, give a general explanation of what your program does. As the programming questions get more complex, the explanations will get lengthier.
- Include comments in your program describing the main steps in your program.

- Display a welcome message which includes your name(s).
- Display clear prompts for users when you are expecting the user to enter data from the keyboard.
- All output should be displayed with clear messages and in an easy to read format.
- End your program with a closing message so that the user knows that the program has terminated.

JavaDoc Documentation:

Documentation for your program must be written in **JavaDoc**.
In addition, the following information must appear at the top of each file:

```
Name(s) and ID(s)     (include full names and IDs)
COMP249
Assignment #          (include the assignment number)
Due Date              (include the due date for this assignment)
```

Submitting Assignment 3

- For this assignment, you are allowed to work individually, or in a group of a maximum of 2 students (i.e. you and one other student). You and your teammate must however be in the same section of the course.
- Only electronic submissions will be accepted. Zip together with the source codes.
- Students will have to submit their assignments (one copy per group) using the Moodle system (please check for your section submission).
- Naming convention for zip file: Create one zip file, containing all source files and produced documentation for your assignment using the following naming convention:

  The zip file should be called *a#_StudentName_StudentID*, where # is the number of the assignment and *StudentName/StudentID* is your name and ID number respectively. Use your "official" name only - no abbreviations or nicknames; capitalize the usual "last" name. Inappropriate submissions will be heavily penalized. For example, for the first assignment, student 12345678 would submit a zip file named like: *a1_Mike-Simon_123456.zip.* if working in a group, the name should look like: *a1_Mike-Simon_12345678-AND-Linda-Jackson_98765432.zip.*

- If working in a team, only one of the members can upload the assignment. Do **NOT** upload the file for each of the members!

⇨ Important: Following your submission, a demo is required (please refer to the course outline for full details). The marker will inform you about the demo times. Please notice that failing to demo your assignment will result in zero mark regardless of your submission.

Evaluation Criteria for Assignment 3 (10 points)

Part 1 (4 points)	
Correct Implementation	2 pts
Proper Use & Testing of the Code	2 pts
Part 2 (6 points)	
Design and Corretness of Classes	3.5 pts
Proper and Sufficient Testing of Your Methods	2 pts
Privay Leak Comments and Proposals to Avoid Them	0.5 pts