The University of Melbourne
School of Computing and Information Systems
COMP90041 Programming and Software Development
Lecturers: Prof Udaya Parampalli & Dr Thuan Pham
Semester 1, 2021

Assignment 2
Due: 10am (AEST), May 14, 2021

1 Overview

In this assignment, you will write a program named **SimpleCompetitions** that helps companies (e.g., retailers) create competitions to boost their sales. For instance, during Easter Sale, a retailer can use your software to create a lucky draw game. In the game, a customer is given entries equivalent to the size of its purchase after their single-purchase final balance, after discounts, reach a specific amount, say $50. Once the competition time ends, the retailer draws winners and the winning customers will get points added to their membership account; customers can use these points for future purchases if they wish. Please read the competition policy and application walk-through sections for more information. This assignment is designed in such a way that we can evaluate your knowledge on the following topics: 1) class design and implementation, 2) control flow structures (e.g., if-then-else & switch-case statements, loops), 3) basic I/O (e.g., Scanner class, formatted printing), 4) arrays, and 5) inheritance & polymorphism.

2 Your Task

Go to *https://classroom.github.com/a/vUhbB-Zy* and accept the assignment. For details on how to check out the repository, make sure to consult the Lab 3 Materials[1]. Once you have cloned the repository, you will find four classes in it:

```
SimpleCompetitions.java
Competition.java
Entry.java
AutoEntry.java
```

`SimpleCompetitions.java` will be the main application containing your `main()` method. The classes come with some initial implementations that follow the design shown in Figure 1. You are asked to add constructors, instance variables, and methods to complete the program. See the competition policy and application walk-through sections to understand the required features of the program. **Note that in the provided UML class diagram and the application skeleton on GitHub, we only suggest methods' names; you can make any modifications to the methods' return types and their argument lists if necessary to complete the task**. You may add your own class(es) if you think it

[1] https://canvas.lms.unimelb.edu.au/courses/105405/assignments/200112

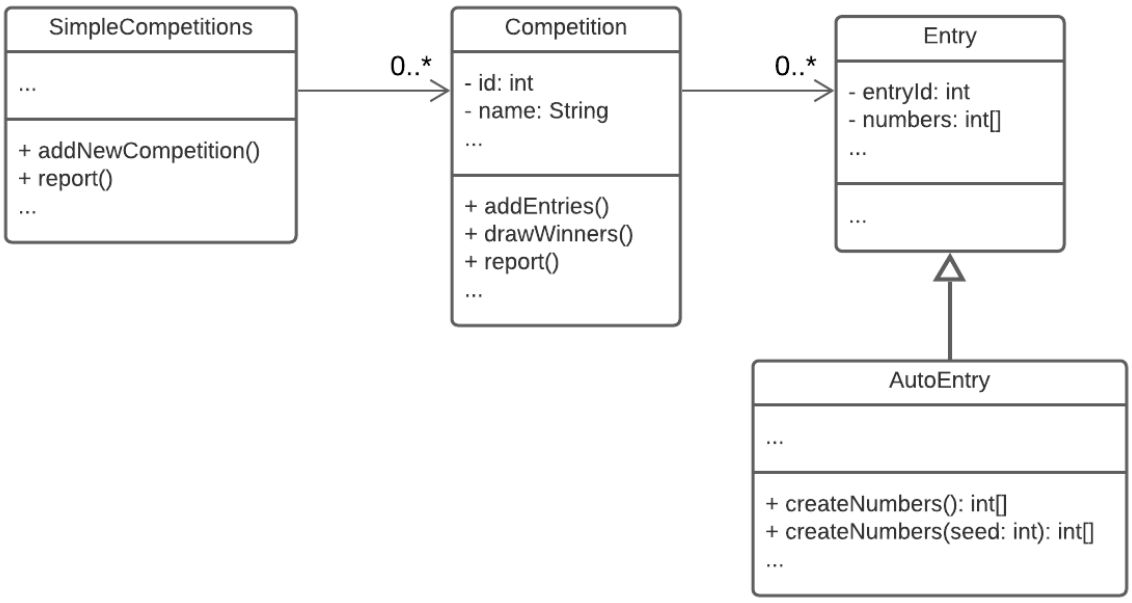

Figure 1: UML Class Diagram for the SimpleCompetitions Application.

is necessary, but your program must contain the classes mentioned above.

For this assignment, all user inputs will be assumed to be of correct data types; such that if a String is expected then it is assumed a string will be entered, and the same for an integer or a decimal number. No error control for incorrectly entered data types are required unless stated otherwise!.

3 Competition Policy

In this assignment, you are asked to implement a specific competition type with the following policy.

- Only customers who have valid membership accounts can enter a competition. Having said that, you do not need to check if a membership account exists in this assignment. As long as the membership identifier is in the correct format, it should be fine.
- There is no limit on the number of competitions that a customer can enter. However, for this assignment, there is only one active competition (whose result is not decided) at a specific point in time i.e. the system does not support concurrent competitions.
- In a specific competition, one customer can have an unlimited number of entries based on their paid bills.
- For each entry, the customer is asked to select 7 numbers in the range from 1 to 35. You can imagine that s/he is provided a card on which s/he can manually select numbers for some entries and/or ask the system to randomly generate entries. Completed cards must be returned to a staff member after purchases so that the information can be inputted into the system.
- Customers get one entry for each $50 in a single bill. For example, if a total bill is $245, it is eligible for getting four entries.
- Once the competition time ends, the organizer can draw winners by using the system to randomly generate a lucky entry. If an entry shares at least two numbers in common with the lucky entry,

the owner of that entry would get some prize. The detailed prizes are listed on Table 1. **Note that if a customer has several winning entries, only the first one with the highest prize will be awarded. By saying first winning entry, we mean the entry that has smallest entry identifier.**

To be matched

Division	Winning number required	Prize
1	7 numbers	50000 points
2	6 numbers	5000 points
3	5 numbers	1000 points
4	4 numbers	500 points
5	3 numbers	100 points
6	2 numbers	50 points

Table 1: Prize table

4 Application Walk-through

For this application, you are asked to support two different modes: 1) the normal mode (i.e. release mode), and 2) the testing mode. The testing mode is designed in such a way that you can test your program in a deterministic manner. The markers will also use that mode for running automated tests. The mode is required because this application is supposed to generate random numbers for both customers' entries and the lucky entries and it is hard to test its correctness without controlling the randomness. More details about how to implement the testing mode are explained below. T/t N/n ✓

1. Your program will start by displaying a welcome message. You will need to choose one mode (normal or testing mode) to start the program. The options are case insensitive and there is a simple check for invalid options. **Note that the program outputs (e.g., menus and output messages) look the same in both the normal and testing mode. The two modes are different only in the ways entry numbers are generated.**

```
----WELCOME TO SIMPLE COMPETITIONS APP----
Which mode would you like to run? (Type T for Testing, and N for Normal mode):
A
Invalid mode! Please choose again.
Which mode would you like to run? (Type T for Testing, and N for Normal mode):
N
Please select an option. Type 5 to exit.
1. Create a new competition
2. Add new entries
3. Draw winners
4. Get a summary report
5. Exit
```

2. The main menu has five options for 1) creating a new competition, 2) adding new entries, both manually and automatically, into the current active competition, 3) drawing to find winners, 4) viewing a summary report of all competitions, and 5) exiting the program. Note that the user **should not** be able to 1) create a new competition if there is already an active one, or 2) add new entries or draw winners if there is no active competition. Error messages should be displayed if users try to do so. Following are some sample error messages. See all error messages in the provided test cases.

```
Please select an option. Type 5 to exit.
1. Create a new competition
```

```
2. Add new entries
3. Draw winners
4. Get a summary report
5. Exit
2
There is no active competition. Please create one!
Please select an option. Type 5 to exit.
1. Create a new competition
2. Add new entries
3. Draw winners
4. Get a summary report
5. Exit
3
There is no active competition. Please create one!
Please select an option. Type 5 to exit.
1. Create a new competition
2. Add new entries
3. Draw winners
4. Get a summary report
5. Exit
```

3. If the first option ("Create a new competition") is selected, by typing "1" and then pressing Enter, the program should call the addNewCompetition method of the SimpleCompetitions class to add a new competition with a given name. After creating a new competition, the program should goes back to the main menu. The newly created competition becomes active so that the user cannot create another competition. If s/he tries to do so, an error message should be displayed.

```
Please select an option. Type 5 to exit.
1. Create a new competition
2. Add new entries
3. Draw winners
4. Get a summary report
5. Exit
1
Competition name:
Easter Holidays
A new competition has been created!
Competition ID: 1, Competition Name: Easter Holidays
Please select an option. Type 5 to exit.
1. Create a new competition
2. Add new entries
3. Draw winners
4. Get a summary report
5. Exit
1
There is an active competition. SimpleCompetitions does not support concurrent competitions!
```

4. Now the user can choose the second option to add entries. The program should first ask for information such as membership ID, bill ID (i.e. receipt number), and the total amount of the bill. Both membership ID and billID need to follow the same format – these must be 6-digit numbers and the numbers can start with zero(es). For example, both 001234 and 123456 are valid identifiers but 123abc is not. The total bill should be used to calculate how many entries the customer can get for this bill.

```
Please select an option. Type 5 to exit.
```

```
1. Create a new competition
2. Add new entries
3. Draw winners
4. Get a summary report
5. Exit
2
Member ID:
1234
Invalid member id! It must be a 6-digit number. Please try again.
Member ID:
123abc
Invalid member id! It must be a 6-digit number. Please try again.
Member ID:
123456
Bill ID:
111aaa
Invalid bill id! It must be a 6-digit number. Please try again.
Bill ID:
112233
Total amount:
102.5
This bill is eligible for 2 entries. How many manual entries did the customer fill up?:
```

5. Once correct information is provided and the number of given entries are decided, your program should ask for the number of manual entries for this bill. In the above example, the customer is eligible to get two entries for their bill. Suppose that s/he manually filled up one entry, by selecting 7 numbers in the range from 1 to 35, and asked for another one to be automatically generated. The program output should looks like the following. **You will observe that manual entries should be added first and entries' indices start from 1.**

```
This bill is eligible for 2 entries. How many manual entries did the customer fill up?:
1
Please enter 7 different numbers (from the range 1 to 35) separated by whitespace.
1 2 3 4 5 6 7
The following entries have been added:
Entry ID: 1      Numbers:  1  2  3  4  5  6  7
Entry ID: 2      Numbers:  2  5 13 17 25 30 35 [Auto]
Add more entries (Y/N)?
```

6. You can see that the numbers in the second entry in the above example were randomly generated by the program. You can do so using different ways. However, in this assignment we leverage the shuffle method of the Collections class[2]. We provide the implementation for the createNumbers method in the AutoEntry class. **The randomness is controlled by the seed argument. In the testing mode, you should use the competition identifier as seed for generating the lucky entry and the number of entries in the currently active competition to generate automated customers' entries.**

```
public int[] createNumbers (int seed) {
    ArrayList<Integer> validList = new ArrayList<Integer>();
    int[] tempNumbers = new int[7];
    for (int i = 1; i <= 35; i++) {
     validList.add(i);
    }
    Collections.shuffle(validList, new Random(seed));
```

[2]https://docs.oracle.com/javase/7/docs/api/java/util/Collections.html

```
    for (int i = 0; i < 7; i++) {
     tempNumbers[i] = validList.get(i);
    }
    Arrays.sort(tempNumbers);
    return tempNumbers;
}
```

Note that there are checks for a valid entry. Error messages should be reported if the given numbers do not adhere to the format.

```
This bill is eligible for 2 entries. How many manual entries did the customer fill up?:
1
Please enter 7 different numbers (from the range 1 to 35) separated by whitespace.
1 2 3 4
Invalid input! Fewer than 7 numbers are provided. Please try again!
Please enter 7 different numbers (from the range 1 to 35) separated by whitespace.
1 2 3 4 4 5 6
Invalid input! All numbers must be different!
Please enter 7 different numbers (from the range 1 to 35) separated by whitespace.
1 2 3 4 5 6 7 8
Invalid input! More than 7 numbers are provided. Please try again!
Please enter 7 different numbers (from the range 1 to 35) separated by whitespace.
```

The user can then add more entries into the current competition and use the 3rd option from the main menu to draw and find winners. As shown below, the program should display the information of the lucky numbers and all the winning entries with their prizes. The entries are sorted by their identifiers. As stated in the competition policy, if a customer has several winning entries, only the first one with the highest prize will be awarded. By saying first winning entry, we mean the entry that has smallest entry identifier.

```
Add more entries (Y/N)?
Y
Member ID:
111222
Bill ID:
125678
Total amount:
305.8
This bill is eligible for 6 entries. How many manual entries did the customer fill up?:
0
The following entries have been added:
Entry ID: 3      Numbers:  9 12 13 18 20 23 28 [Auto]
Entry ID: 4      Numbers:  1  2  9 10 11 17 35 [Auto]
Entry ID: 5      Numbers:  6  9 14 15 20 23 30 [Auto]
Entry ID: 6      Numbers:  5 13 14 17 23 27 33 [Auto]
Entry ID: 7      Numbers:  6  9 13 16 18 24 29 [Auto]
Entry ID: 8      Numbers:  1  2  5 16 17 21 30 [Auto]
Add more entries (Y/N)?
N
Please select an option. Type 5 to exit.
1. Create a new competition
2. Add new entries
3. Draw winners
4. Get a summary report
5. Exit
3
Lucky entry for Competition ID: 1, Competition Name: Easter Holidays
```

```
Numbers:  3  4 17 18 24 29 30 [Auto]
Winning entries:
Member ID: 123456, Entry ID: 1     , Prize: 50   , Numbers:  1  2  3  4  5  6  7
Member ID: 111222, Entry ID: 7     , Prize: 100  , Numbers:  6  9 13 16 18 24 29 [Auto]
Please select an option. Type 5 to exit.
1. Create a new competition
2. Add new entries
3. Draw winners
4. Get a summary report
5. Exit
```

Now the user can follow similar steps to create other competitions, or s/he can use the 4th option to display a summary report, or the 5th option to terminate the program.

```
Please select an option. Type 5 to exit.
1. Create a new competition
2. Add new entries
3. Draw winners
4. Get a summary report
5. Exit
1
Competition name:
Anzac Day
A new competition has been created!
Competition ID: 2, Competition Name: Anzac Day
Please select an option. Type 5 to exit.
1. Create a new competition
2. Add new entries
3. Draw winners
4. Get a summary report
5. Exit
4
----SUMMARY REPORT----
+Number of completed competitions: 1
+Number of active competitions: 1

Competition ID: 1, name: Easter Holidays, active: no
Number of entries: 8
Number of winning entries: 2
Total awarded prizes: 150

Competition ID: 2, name: Anzac Day, active: yes
Number of entries: 0
Please select an option. Type 5 to exit.
1. Create a new competition
2. Add new entries
3. Draw winners
4. Get a summary report
5. Exit
5
Goodbye!
```

5 Example Executions

In this section, we show two example executions for both the normal mode and the testing mode. You may observe that the program logic is the same and the outputs look similar. As stated above, the two

modes are only different in the ways random numbers are generated. Specifically, for the testing mode, since the randomness is controlled by using seeds, if you provide the same input, the program should produce the same output. **Note that in the example executions, we show both the input and output of the program. In the given test cases, the input and output are stored in different files. See Section 7 (Testing Before Submission) for more details.**

5.1 Example Execution in Normal Mode

```
----WELCOME TO SIMPLE COMPETITIONS APP----
Which mode would you like to run? (Type T for Testing, and N for Normal mode):
N
Please select an option. Type 5 to exit.
1. Create a new competition
2. Add new entries
3. Draw winners
4. Get a summary report
5. Exit
1
Competition name:
Easter Holidays
A new competition has been created!
Competition ID: 1, Competition Name: Easter Holidays
Please select an option. Type 5 to exit.
1. Create a new competition
2. Add new entries
3. Draw winners
4. Get a summary report
5. Exit
2
Member ID:
123456
Bill ID:
112233
Total amount:
102.5
This bill is eligible for 2 entries. How many manual entries did the customer fill up?:
1
Please enter 7 different numbers (from the range 1 to 35) separated by whitespace.
1 2 3 4 5 6 7
The following entries have been added:
Entry ID: 1      Numbers:  1  2  3  4  5  6  7
Entry ID: 2      Numbers:  2  4  6 11 12 23 34 [Auto]
Add more entries (Y/N)?
Y
Member ID:
111222
Bill ID:
125678
Total amount:
305.8
This bill is eligible for 6 entries. How many manual entries did the customer fill up?:
0
The following entries have been added:
Entry ID: 3      Numbers:  1  2 12 13 16 17 34 [Auto]
Entry ID: 4      Numbers:  1 10 21 30 31 32 35 [Auto]
Entry ID: 5      Numbers:  3  7  9 10 13 19 35 [Auto]
```

```
Entry ID: 6      Numbers:  5  9 15 19 28 31 32 [Auto]
Entry ID: 7      Numbers:  1 10 12 18 22 27 32 [Auto]
Entry ID: 8      Numbers:  3  4 11 13 17 24 30 [Auto]
Add more entries (Y/N)?
N
Please select an option. Type 5 to exit.
1. Create a new competition
2. Add new entries
3. Draw winners
4. Get a summary report
5. Exit
3
Lucky entry for Competition ID: 1, Competition Name: Easter Holidays
Numbers:  1  3  8 13 24 25 31 [Auto]
Winning entries:
Member ID: 123456, Entry ID: 1     , Prize: 50   , Numbers:  1  2  3  4  5  6  7
Member ID: 111222, Entry ID: 8     , Prize: 100  , Numbers:  3  4 11 13 17 24 30 [Auto]
Please select an option. Type 5 to exit.
1. Create a new competition
2. Add new entries
3. Draw winners
4. Get a summary report
5. Exit
1
Competition name:
Anzac Day
A new competition has been created!
Competition ID: 2, Competition Name: Anzac Day
Please select an option. Type 5 to exit.
1. Create a new competition
2. Add new entries
3. Draw winners
4. Get a summary report
5. Exit
4
----SUMMARY REPORT----
+Number of completed competitions: 1
+Number of active competitions: 1

Competition ID: 1, name: Easter Holidays, active: no
Number of entries: 8
Number of winning entries: 2
Total awarded prizes: 150

Competition ID: 2, name: Anzac Day, active: yes
Number of entries: 0
Please select an option. Type 5 to exit.
1. Create a new competition
2. Add new entries
3. Draw winners
4. Get a summary report
5. Exit
5
Goodbye!
```

5.2 Example Execution in Testing Mode

```
----WELCOME TO SIMPLE COMPETITIONS APP----
Which mode would you like to run? (Type T for Testing, and N for Normal mode):
T
Please select an option. Type 5 to exit.
1. Create a new competition
2. Add new entries
3. Draw winners
4. Get a summary report
5. Exit
1
Competition name:
Easter Holidays
A new competition has been created!
Competition ID: 1, Competition Name: Easter Holidays
Please select an option. Type 5 to exit.
1. Create a new competition
2. Add new entries
3. Draw winners
4. Get a summary report
5. Exit
2
Member ID:
123456
Bill ID:
112233
Total amount:
102.5
This bill is eligible for 2 entries. How many manual entries did the customer fill up?:
1
Please enter 7 different numbers (from the range 1 to 35) separated by whitespace.
1 2 3 4 5 6 7
The following entries have been added:
Entry ID: 1      Numbers:  1  2  3  4  5  6  7
Entry ID: 2      Numbers:  3  4 17 18 24 29 30 [Auto]
Add more entries (Y/N)?
Y
Member ID:
111222
Bill ID:
125678
Total amount:
305.8
This bill is eligible for 6 entries. How many manual entries did the customer fill up?:
0
The following entries have been added:
Entry ID: 3      Numbers:  2  7 14 18 22 25 35 [Auto]
Entry ID: 4      Numbers:  1  5  6  8 31 32 35 [Auto]
Entry ID: 5      Numbers:  2  5 11 12 23 24 34 [Auto]
Entry ID: 6      Numbers:  1  2 13 24 25 31 34 [Auto]
Entry ID: 7      Numbers:  2  3 15 18 26 27 31 [Auto]
Entry ID: 8      Numbers:  4  6 14 16 17 18 20 [Auto]
Add more entries (Y/N)?
N
Please select an option. Type 5 to exit.
```

```
1. Create a new competition
2. Add new entries
3. Draw winners
4. Get a summary report
5. Exit
3
Lucky entry for Competition ID: 1, Competition Name: Easter Holidays
Numbers:  3  4 17 18 24 29 30 [Auto]
Winning entries:
Member ID: 123456, Entry ID: 2     , Prize: 50000, Numbers:  3  4 17 18 24 29 30 [Auto]
Member ID: 111222, Entry ID: 8     , Prize: 100  , Numbers:  4  6 14 16 17 18 20 [Auto]
Please select an option. Type 5 to exit.
1. Create a new competition
2. Add new entries
3. Draw winners
4. Get a summary report
5. Exit
1
Competition name:
Anzac Day
A new competition has been created!
Competition ID: 2, Competition Name: Anzac Day
Please select an option. Type 5 to exit.
1. Create a new competition
2. Add new entries
3. Draw winners
4. Get a summary report
5. Exit
4
----SUMMARY REPORT----
+Number of completed competitions: 1
+Number of active competitions: 1

Competition ID: 1, name: Easter Holidays, active: no
Number of entries: 8
Number of winning entries: 2
Total awarded prizes: 50100

Competition ID: 2, name: Anzac Day, active: yes
Number of entries: 0
Please select an option. Type 5 to exit.
1. Create a new competition
2. Add new entries
3. Draw winners
4. Get a summary report
5. Exit
5
Goodbye!
```

6 Assessment & Important Notes

This project is worth 15% of the total marks for the subject.

Your Java program will be assessed based on correctness of the output as well as quality of code implementation.

Automatic tests will be conducted on your program by compiling, running, and comparing your outputs for several test cases with generated expected outputs. The automatic test will deem your output wrong if your output does not match the expected output, even if the difference is just having an **extra space or missing a colon**. Therefore, it is crucial that **your output follows exactly the same format shown in the examples above.**

Also, use **ONLY ONE** Scanner object throughout your program. Otherwise the automatic tests may cause your program to generate exceptions and terminate. The reason is that in the automatic test, multiple lines of test inputs are sent all together to the program. As the program receives the inputs, it will pass them all to the currently active Scanner object, leaving the rest of the Scanner objects nothing to read and hence cause a run-time exception. Therefore, it is important that **your program has only one Scanner object.** Arguments such as "It runs correctly when I do the manual test, but fails under the automatic test" will not be accepted.

7 Testing Before Submission

You will find all input files and the expected output files in the Projects page on Canvas for you to use on your own. **In this assignment, the markers will use the same set of test data for marking.** You should use them to test your code carefully and then submit your code on GitHub. Note that each commit you make is recorded in your GitHub repository. Details of how the submission works can be found in Section 8.

To test your code by yourself:

1. Check your source code files, and make sure you have the test input data files ready (e.g.,"`input1.txt`").
2. Open a console, navigate to your project directory (where your .java classes reside), and compile your program: `javac *.java` (this command will compile all your java files in the current folder).
3. Run command: `java SimpleCompetitions < input1.txt > my-output1.txt` (it runs the program using contents in "`input1.txt`" as input and write the output to `my-output1.txt`).
4. Inspect the file `my-output1.txt` as it contains any errors your program execution may have encountered.
5. Compare your result with the provided output file `output1.txt`. Fix your code if they are different.
6. Repeat steps 3-5 for all given input and output pairs. When you are satisfied with your project, commit your changes to GitHub.

   Please follow the instructions in Lab-3 documentation if you want to use IDEs like Eclipse.

8 Submission

Your submission should have at least four Java source code files. You must name them correctly: `SimpleCompetitions.java`, `Competition.java`, `Entry.java`, `AutoEntry.java` (if you successfully cloned the assignment repository, these files should already be in your working directory). You can add more classes if you want but **please ensure that you do not include irrelevant files (e.g., unused classes or test files) in your submission**. You should verify your submission locally as described above before submitting your code to GitHub.

You can edit and re-submit your code as many times you want as long as you submit before the submission deadline of the assignment. **Any updates to your code on GitHub made after the submission deadline will incur a late-submission penalty. This means that even if you make a minor edit and re-submit a file after the deadline your entire submission will be subject to a**

late-submission penalty!

The deadline for the project is **10am (AEST), May 14, 2021**. The allowed time is more than enough for completing the project.

What will be graded? The **last** version of your program committed to GitHub before the submission deadline. Submissions via email will not be accepted!

8.1 Late-Submission Penalties

There is a 20% penalty per day for late submissions.

For example, suppose your project gets a mark of 5 but changes are submitted within 1 day after the deadline, then you get **20% penalty** and the mark will be 4 after the penalty.
There will be **0 marks for your submission if you make changes after the 10am (AEST), May 18, 2021**.

9 Individual Work

Note well that this project is part of your final assessment, so copying, working together, sharing work (i.e. cheating) is not acceptable! Any form of material exchange, whether written, electronic or any other medium is considered cheating, as is copying from any online sources in case anyone shares it. Providing undue assistance is considered as serious as receiving it, and in the case of similarities that indicate exchange of more than basic ideas, **formal disciplinary action will be taken for all involved parties without exceptions! We have a sophisticated tool that undertakes deep structural analysis of Java code to identify regions of similarity**.