Department of Computing

Faculty of Science and Engineering

Macquarie University

COMP125 - Fundamentals of Computer Science

Assignment 1 (Worth 5%)

Due date: Friday 24th March, 23:45

INSTRUCTIONS

You are given two options for this assignment:

a. ReportCard project – easier
b. LargeInteger project – more challenging

Both options are worth the same number of marks. LargeInteger provides a greater challenge for students seeking one. You need to submit only one completed project on iLearn

…continued on next page

OPTION 1: ReportCard Project

Files to be completed:

1. ReportCard.java (80 marks)

This is the core class that you need to complete. It represents marks of a student in the units (s)he attends. The data is held in an array marks. Some methods in the class are completed for you and you should not modify them. These methods are:

1. getMark()
2. getMarks()

You need to complete all other methods based on **Javadoc method comments**. You may add other *helper* methods if required. The list of methods to complete (in the order we suggest you complete these methods) are:

1. constrain(double, int, int) (5 marks)
2. setMarks(double[]) (10 marks)
3. ReportCard(double[]) (5 marks)
4. ReportCard() (5 marks)
5. countUnits() (5 marks)
6. average() (5 marks)
7. allPasses(double) (5 marks)
8. scale(double) (10 marks)
9. stdDev() (10 marks)
10. compareTo(ReportCard) (10 marks)
11. getGrades() (10 marks)

There are JUnit tests provided for each method in ReportCardTest.java. Complete a method, run the JUnit test corresponding to that method (select the test method by double-clicking on it and then “Run”). If it gives a green bar, the test passes, and you can move on. If it gives a red bar, use the feedback it provides to identify the mistake and fix it. Use the debugger to isolate the problem.

Note: as long as the JUnit tests pass, you are free to add other methods in your class.

2. ReportCardClient.java (20 marks)

The client must make use of the ReportCard class by creating objects of the class and performing operations on these objects. The details of the operations are provided as a comment in Client.java. If you do everything correctly, the output should be:

```
0.0 (Fail)
0.0 (Fail)
10.0 (Fail)
20.0 (Fail)
30.0 (Fail)
40.0 (Fail)
50.0 (Pass)
60.0 (Pass)
70.0 (Credit)
80.0 (Distinction)
90.0 (High Distinction)
100.0 (High Distinction)
100.0 (High Distinction)
```

Help Program

Please note that a help program is provided with the assignment package in classWithArrayJUnit.zip

OPTION 2: LargeInteger Project

This option deals with storing large integers (for example,

598236094624906846948019765047610317607014672408761406740857641086748056742086742086728067428067368045760847608764082708671506870857460847680427608760147608176807468042674807680745086

as Strings, and performing arithmetic operations on the String itself.

You cannot use any method from the Math class or the String class (use of length(), charAt(), indexOf(char), substring(int), substring(int, int) is allowed) or BigInteger class. Our tool will detect use of any of the prohibited methods and also strip any imports from the files.

Files to be completed:

1. LargeInteger.java (85 marks)

This class stores the large integer as:

a. number: String - containing the numerical value. For both -15 and 15, number = "15"
b. negative: boolean – stores sign for the value. For -15 negative = true, for 15, negative = false

The methods to be completed are:

1. getWithoutLeadingZeroes(String): 5 marks
2. isInteger(String): 10 marks
3. setNumber(String): 10 marks
4. equals(Object): 10 marks
5. compareTo(LargeInteger): 10 marks
6. negate(): 5 marks
7. absolute(): 5 marks
8. compareAbsolutes(LargeInteger): 5 marks
9. add(LargeInteger): 10 marks
10. subtract(LargeInteger): 10 marks

There are JUnit tests provided for each method in LargeIntegerTest.java. Complete a method, run the JUnit test corresponding to that method (select the test method by double-clicking on it and then “Run”). If it gives a green bar, the test passes, and you can move on. If it gives a red bar, use the feedback it provides to identify the mistake and fix it. Use the debugger to isolate the problem.

Note: as long as the JUnit tests pass, you are free to add other methods in your class.

2. LargeIntegerClient.java (20 marks)

Requirements of the client are provided in the file. If your LargeInteger.java code is correct and you follow the client instructions correctly, the output should be:

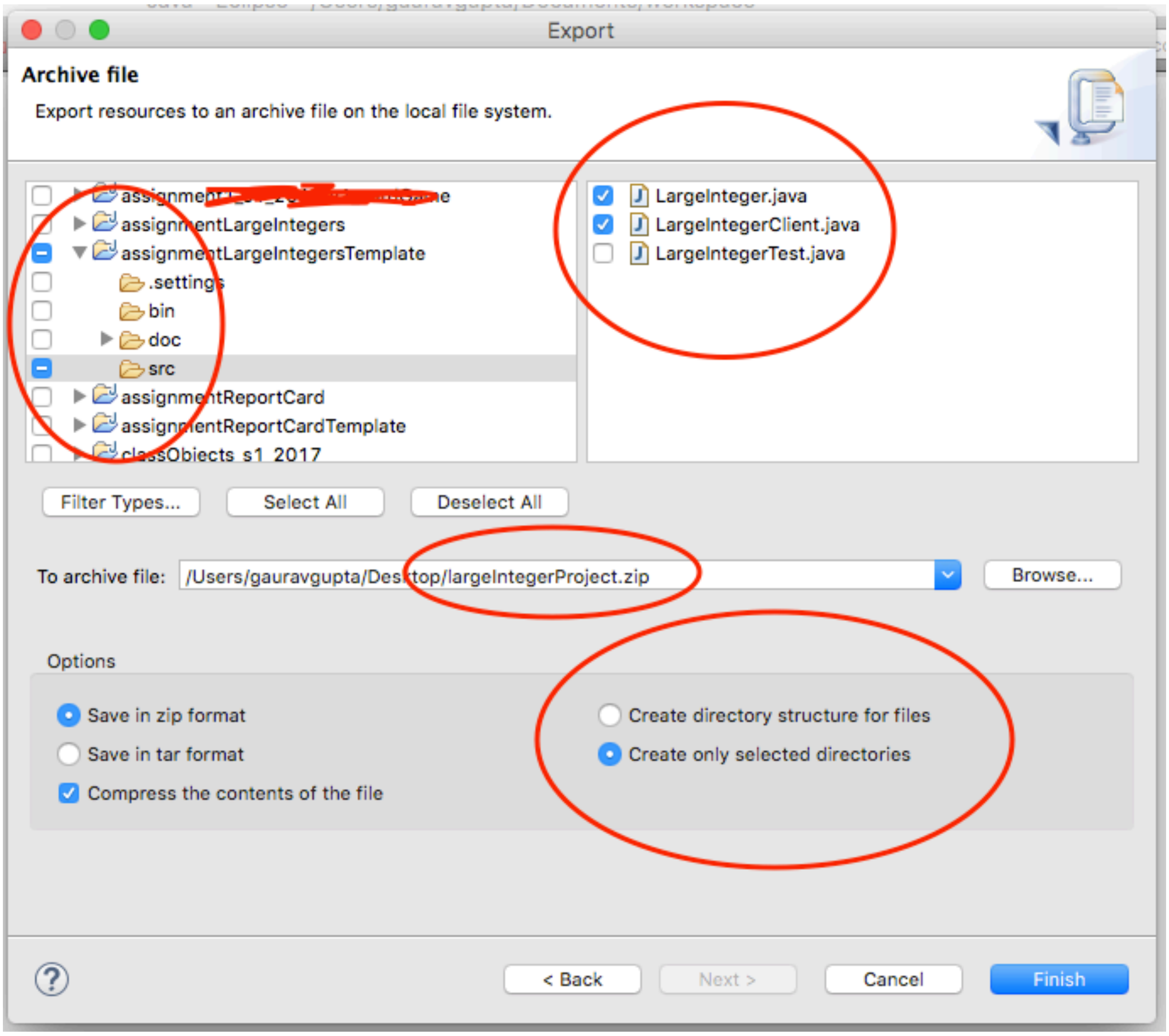
```
10360428604534086240673840928550543796238171974943838278910102479265299237861
```

SUBMISSION INSTRUCTIONS

Submit either,

a. ReportCard.java and ReportCardClient.java exported as archive file reportCardProject.zip, or,
b. LargeInteger.java and LargeIntegerClient.java and Client.java exported as archive file largeIntegerProject.zip

Export procedure is provided in exportingAssignment1.mp4 and a screenshot is given below -



Export

Archive file

Export resources to an archive file on the local file system.

- □ [redacted]
- □ assignmentLargeIntegers
- ⊟ assignmentLargeIntegersTemplate
  - □ .settings
  - □ bin
  - □ doc
  - ⊟ src
- □ assignmentReportCard
- □ assignmentReportCardTemplate
- □ classObjects_s1_2017

- ☑ LargeInteger.java
- ☑ LargeIntegerClient.java
- □ LargeIntegerTest.java

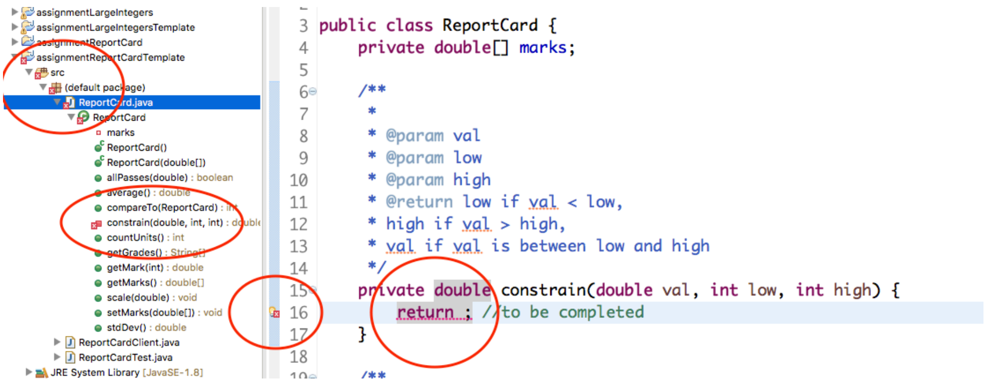
To archive file: /Users/gauravgupta/Desktop/largeIntegerProject.zip

Options

- ● Save in zip format
- ○ Save in tar format
- ☑ Compress the contents of the file
- ○ Create directory structure for files
- ● Create only selected directories

We will deploy plagiarism detection tools and ANY submission flagged for plagiarism will be reported to the university.

Ensure you have no compilation errors (red crosses in any of the files as shown below). 15% penalty will be applied if there are any compilation errors as it means that we need to manually mark your assignment.



```
public class ReportCard {
    private double[] marks;

    /**
     *
     * @param val
     * @param low
     * @param high
     * @return low if val < low,
     * high if val > high,
     * val if val is between low and high
     */
    private double constrain(double val, int low, int high) {
        return ; //to be completed
    }

    /**
```

LATE SUBMISSIONS

Late submissions up to 2 days (48 hours) from the due date will be accepted with a **penalty of 0.5% per hour**. If you cannot submit the assignment in time due to unavoidable circumstances, apply for Disruption to Studies via http://ask.mq.edu.au/

Note: take the Professional Authority Form to the doctor if requesting extension on medical grounds. The form is available at:

http://www.mq.edu.au/__data/assets/pdf_file/0009/183375/professional_authority_form_paf.pdf

RETENTION OF RECORDS

Student are required to keep a copy of all items submitted or completed for the purpose of assessment or evaluation until the end of the grade appeal period.

INFORMATION ABOUT HOW AND WHEN FEEDBACK WILL BE PROVIDED

Feedback will be provided within one week via iLearn. Please note that the JUnit tests provide instant feedback on your code.

ACADEMIC HONESTY

Using the work or ideas of another person, whether intentionally or not, and presenting them as your own without clear acknowledgement of the source is called Plagiarism.

Macquarie University promotes awareness of information ethics through its Academic Honesty Policy. This means that:

- all academic work claimed as original must be the work of the person making the claim
- all academic collaborations of any kind must be acknowledged
- academic work must not be falsified in any way
- when the ideas of others are used, these ideas must be acknowledged appropriately.

All breaches of the Academic Honesty Policy are serious and penalties apply. Students should be aware that they may fail an assessment task, a unit or even be excluded from the University for breaching the Academic Honesty Policy.

IMPORTANT: We will be running all submissions through Java plagiarism detection tools and if you are flagged for plagiarism, you will be reported to the faculty disciplinary committee and the teaching staff will have no further say in the matter.