CE203 ASSIGNMENT 2021

Credit:
30% of total module mark

Module Learning Outcomes Examined:
1) Demonstrate knowledge of core Java application libraries. 3) Write Java programs with interactive graphical user interfaces (GUIs). 5) Write Java programs that make efficient use of the Java collections package.

Submission Deadline:
Wednesday 24/11/2021 before 11:59:59 (Week 8)

Submission will be via the Faser online submission system

You should refer to the Undergraduate Students' Handbook for details of the departmental policy regarding late submission and university regulations regarding plagiarism; **the work handed in must be entirely your own**. It is expected that marking of the assignment will be completed within 3/4 working weeks of the submission deadline.

Introduction
This assignment involves writing a Java application using Swing and AWT GUI components and a Java collections data structure. This must be hand-coded by you. If you use IntelliJ Idea or any other IDE you must not use the GUI Designer or any similar tool. Additionally, you should not use any other external libraries not part of the Java SE API. The submitted program **must** be runnable without dependencies on any external libraries not part of the Java SE used for the module. The files should contain ***brief*** comments.

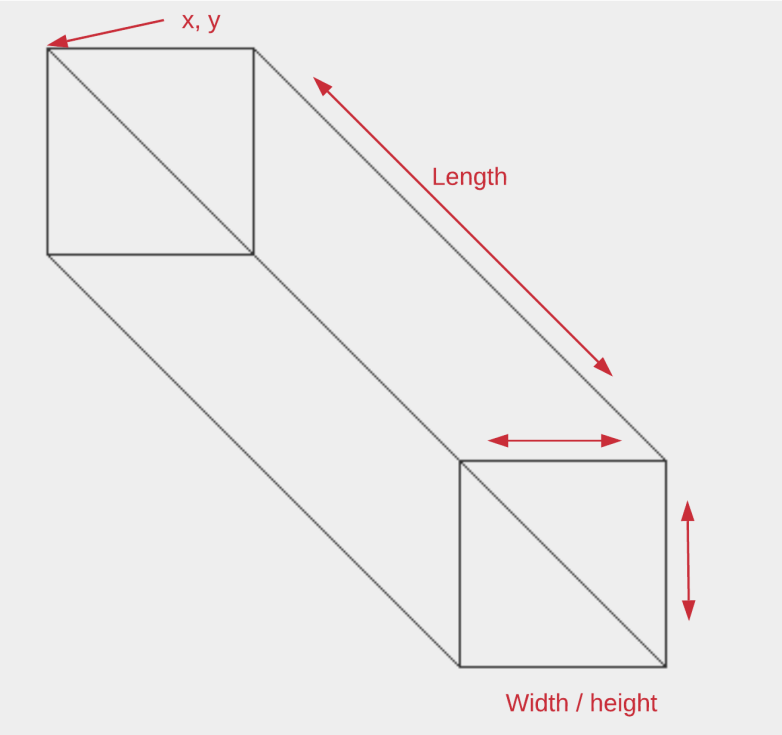
WARNING AND ADVICE ABOUT POSSIBLE ACADEMIC OFFENCES
Your solutions should be your own unaided work. You can make use of any of the programs from the CE203 lecture notes and the lab templates on Moodle. You may use other features from the Java Standard Edition Application Programming Interface (Java SE API) including those not covered in CE203.
You must **NOT** use any third-party classes (e.g. classes that are **not** provided as part of the Java SE used for this module). For more information, please see the University pages on plagiarism and the Academic Offences Procedures.
DO NOT COPY PROGRAM CODE FOR THIS ASSIGNMENT FROM ANOTHER STUDENT OR FROM THE INTERNET OR FROM ANY OTHER SOURCES. DO NOT LET OTHER STUDENTS COPY YOUR WORK

Write a frame-based application that allows the user to store and manipulate a list of objects of type CuboidContainer for which an incomplete java class is provided.

CuboidContainer is defined with the following attributes:

Int *id*: Cuboid id should be a six digit non-negative integer with no leading 0s
Int *x*: x pixel position of top left point of the first face of the cuboid (hardcoded)
Int *y*: y pixel position of top left point of the first face of the cuboid (hardcoded)
Int *Width / helight*: Width and height of the cuboid where width = height (each face is a square)
Int *Length*: Length of cuboid in pixels
Point[] *faceOnePoints*; // Points array used to store vertices (points) of first face of the cuboid
Point[] *faceTwoPoints*; // Points array used to store vertices (points) of second face of the cuboid

CuboidContainer class contains incomplete methods that you will use to draw a cuboid using the dimensions specified in the attributes. The drawn cuboid should look like the figure below where I have labelled how each of the attributes relate to it.



The incomplete ContainerPanel and ContainerFrame java classes provide a starting point for you where the functionality to draw an incomplete cuboid is given.

The CuboidContainer class also implements the Comparable interface and an incomplete compareTo() method is provided for you to also complete.

The layout of the GUI should be such that any buttons should be displayed at the bottom of the frame and user input should be provided using appropriate text fields that should be displayed at top of the frame. Here you may choose to display buttons and text fields in their own panels. The centre of the GUI should contain a panel to display the drawn cuboid. Here you may choose to use and modify the ContainerPanel java class provided. Any other application outputs should be displayed through print statements in the command window.

The application should provide features and corresponding buttons that allow the user to do the following:

a) Add a CuboidContainer object to the list where the user input text fields should be used to specify values for width/height, length, colour and id).
b) Display all the ids of CuboidContainer from the list in the command window.
c) Search for and retrieve a CuboidContainer object specified by its id (using one of the input text fields). Draw the corresponding cuboid on the central panel of the application. The cuboid should be drawn using the dimension and colour attribute values specified in the retrieved CuboidContainer object.
d) Sort the list of CuboidContainer objects in ascending order of their ids and display the contents of the list in the command window. Here you need to specify an appropriate compareTo() method in the CuboidContainer class and explore the use of Collections.sort() to automatically sort the contents of the list. See Java API and an example of using it here: https://www.geeksforgeeks.org/collections-sort-java-examples/ (note the use of generics).

Id attribute values may contain only whole digits, so should not contain any negative digits, leading zeros, or other characters. The ids should be a fixed length of 6 digits when input by the user. You should not allow the list to store objects with duplicate ids.

For each action specified above an appropriate message should be displayed in the command window confirming the action, e.g.

"Cuboid '126658' has been added to the list".

An appropriate error message should be displayed for any failed action e.g.

"The Cuboid '12786n' was not added to the list as a valid id was not provided."

The list for storing the CuboidContainer objects should be implemented using an `ArrayList`

User input exceptions should be captured by the program using exception handling or acceptable error checking.

You should aim to make your user interface easy to use with a clear layout that makes appropriate use of spacing between the displayed components.

You many have defined more than one Java class for your application. It is normally good practice to save these in their own physical .java files. Your code should contain one class that contains your `main()` method from where you the application executes. All your code files must be clearly identifiable with your student registration number. You must check that the code runs prior to submitting it as marks will be lost if the markers are unable to compile and run the code.

Commenting the Programs

The programs should contain brief comments indicating precisely what each method in the code does and what each instance variable is used for. You should not write many comments stating what individual lines of code do. In places it may be appropriate to simply state what a block of code does (e.g. "check that the input is a valid id").

The programs should be laid out neatly and be readable with clear indentation.

Program Testing Output

The program produced should be fully tested to demonstrate that each of the features you are asked to implement is working correctly.

You are therefore asked to include either a Word or PDF document where you clearly show for each feature the correct output of your code. This can be shown as a series of screenshots that demonstrate the correct response of the program to each button press or user input / button press event. For example, where you are asked to display all the CuboidContainer objects by ids from the list, you should display a screenshot where this is shown to be correctly output on the command line following the button press event.

Please briefly explain or label the screenshots for each feature being tested for example: 'b), c)…'

The Word of PDF document should include your student registration number and should be named according to the following naming convention:
Assignment 1_Testing_RegNo_<*registration number*>.

Submission

You are required to submit the source code files. You need to also include the Word/PDF file containing your program testing output. All the files should be placed in a single folder, which should then be *zipped* for submission to the online system, i.e. the submission should be a single file in *zip format*. **Other formats (e.g. *rar* format) and submissions without the source code will <u>not</u> be accepted and will result in a mark of zero.**

Assessors will need to be able to run your code as is without any modifications required. **So, you must make sure that your code is able to run correctly prior to submission.**

Marks will be awarded for following *all* the requirements (including naming of files, layout of the interface etc.) and applying object-oriented programming principles appropriately.

Marking Criteria

1. Program compiles and runs	**/20%**
2. Correct layout, display of GUI components and program outputs	**/20%**
- Correct layout display of program outputs and correct display of drawn cuboid on GUI panel using stored parameters and colour	
3. Correct button functionality	**/20%**
- Adding CuboidContainer objects based on unique id as specified	
- Display all CuboidContainer objects by id in list	
- Search and retrieve a CuboidContainer objects by its id and display it	
- Sort CuboidContainer objects by Id in ascending order as specified using Collections.sort()	
- Correct use of ArrayList java collections data structure	
4. Exception handling or other acceptable error checking approach	**/20%**
5. Overall elegance and correctness of code	**/20%**
- Commenting of code	
- Object-orientated approach including the use of Generics	
Total	**/100%**

For each of the 5 marking criteria, marks will be awarded out of 20% based on the following bands:

Poor: <=8%
Satisfactory: 9-10%
Good: 11-13%
Excellent: 14-16%
Outstanding: 17-20%