Family Name:. . . . . **Other Names:** . . . .

Student ID:. . . . . **Signature**. . . . .

COMP 102: Lab Assessment 3

2021, Dec 17

Instructions

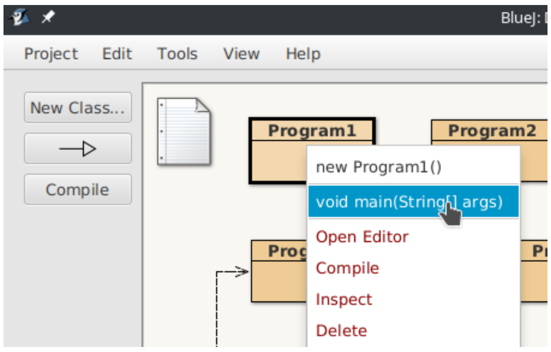
- Time allowed: **TWO AND A HALF HOURS**
- Attempt **all** the questions. There are 120 marks in total.
- To write your answers in BlueJ, follow the instructions on page 2.
- If you think a question is unclear, ask for clarification.
- This test contributes 40% of your final grade.
- You may access the online Java Documentation.
- You may access the lecture notes for COMP102.
- You may use dictionaries and calculators.
- You may not access any other web sites or online help of any kind.
- You may write notes and working on this paper, but make sure your answers are clear.

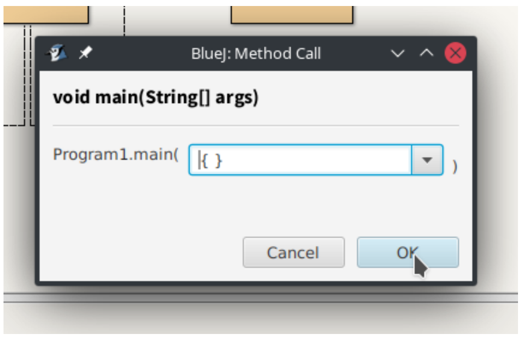
Questions:

1. Java Basics	[25]
2. Design a class	[25]
3. Files	[15]
4. ArrayLists of Objects	[30]
5. 2D Arrays	[25]

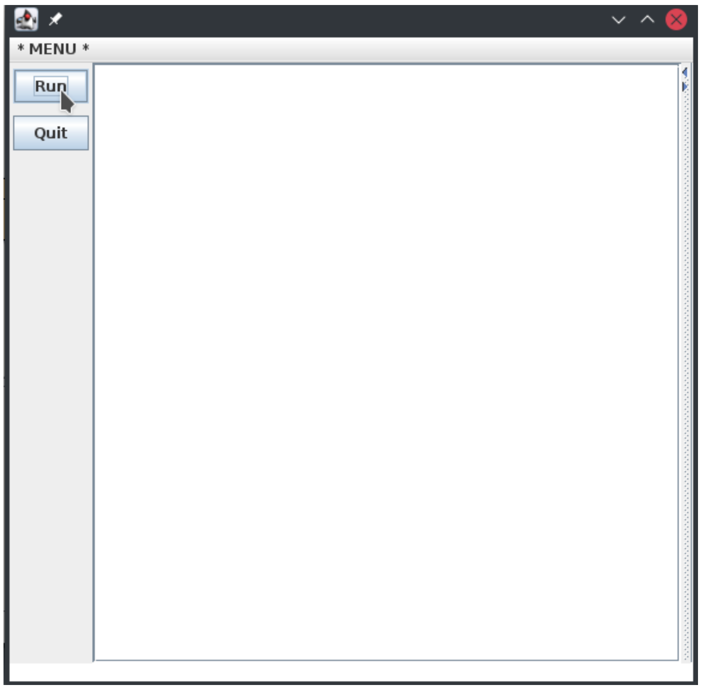
Running the code:

1. Download the COMP102-LabAssessment-3.zip file from
   https://ecs.wgtn.ac.nz/Courses/COMP102_2020T3/LabAssessment3
2. Unzip the file, and open the project in BlueJ.
3. Each question has been given its own class. To run them, right-click and select 'void main(String[] args)'. Hit 'OK' when it prompts you for arguments.





4. The UI that pops up will have a 'Run' button that will execute the code for the question, once you've written it.



Question 1. Java Basics **[25 marks]**

Answer this question in the Program1 class.

(a) **[8 marks] Conditionals**.

Complete the method calculateVolume(...) in Program1 which calculates the area of a cube or a sphere.

The method has two parameters:

1. shape, a string which is either ”cube” or ”sphere”
2. size, a double that is the size of the shape
   - if shape is ”cube”, this is the length of a side
   - if shape is ”sphere” this is the radius of the sphere

Complete the method so that it correctly determines which calculation to use, and returns the correct answer. You do not need to check for invalid shapes – if the shape isn’t one of the options, default to the other.

Note:

- The volume of a cube is given by: $l \times l \times l$, where l is the length of one of its sides
- The volume of a sphere is given by: $(4.0/3.0) \times \pi \times r^3$, where r is the radius

E.g.

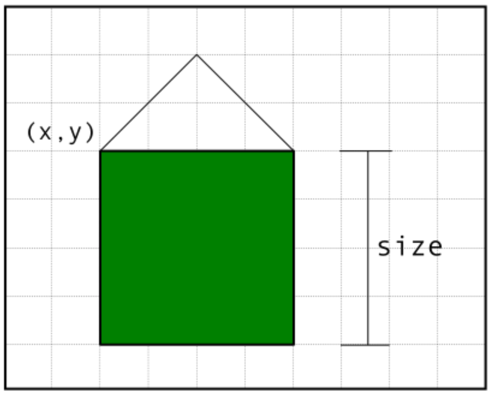
```
INPUT:   shape: "cube", size: 2.0
OUTPUT:  8.00

INPUT:   shape: "sphere", size: 2.0
OUTPUT:  33.51
```

(Question 1 continued on next page)

(Question 1 continued)

(b) **[9 marks] Defining and calling methods**.

Complete the drawHouse(...) method in Program1 so that it draws a house in the graphics pane. A house is a filled square with two lines drawn to represent a roof, as shown below.

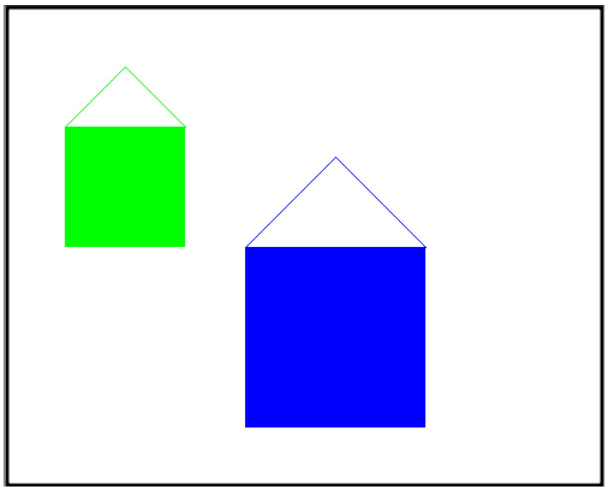


The method should have **four** parameters:

- the coordinates of the top-left corner of the square
- the size of the square
- the colour of the house

The roof is drawn as two lines. The centre point of the roof is (size/2) units above the house.

Then, complete the draw2Houses() method to call the drawHouse(...) method twice to draw the picture shown below. Precise sizes and co-ordinates are given in the code.



(Question 1 continued on next page)

(Question 1 continued)

(c) **[8 marks] Loops and Arrays**.

The field nums in Program1 contains an array of integers:

```
private int[] nums = new int[]{2,4,6,2,-3,3,5,4,3,2,1,4,5,7,2,0,9};
```

Complete the arrayRange() method in Program1 to print out the range of the array in the following format:

The range is [difference] ([min] to [max])

For the array in nums, this method should print out

The range is 12 (-3 to 9)

Question 2. Design a class **[25 marks]**

Complete the BoardGame class in BlueJ to specify BoardGame objects that are used by the methods in Program2.

You do not need to modify the Program2 class, only the BoardGame class.

1. Each board game should have fields that store infomation about the game. You will need to add four fields to the BoardGame class so that it stores:
   - The name of the board game
   - The minimum number of players required
   - The maximum number of players allowed
   - How long the game takes to play, based on the number of players (specified in minutes / player)
2. The constructor should create a BoardGame object with the specified name, minimum and maximum numbers of players, and play time (in minutes per player)
3. The print() method should print the game information, in the following format:

   [name] is for [min]-[max] players, and takes [minTime] - [maxTime] minutes

   Where minTime and maxTime are calculated based on the min/max players and minutes per player

   For example, Gloomhaven takes 1-4 players, and 30 minutes per player. This prints:

   Gloomhaven is for 1-4 players, and takes 30-120 minutes to play
4. The checkPlayable(...) method should checks if the game is playable with the specified number of players, within the specified time.

   It should determine whether the game is playable with the constraints given by the parameters.
   - If there are not enough players, it should print:
     "Not playable. [name] requires more players than you have."
   - If there are too many players, it should print:
     "Not playable. [name] requires fewer players than you have."
   - If there are the right number of players *but* the game would take longer than the indicated time, it prints:
     "Not playable. [name] requires more time than you have."
   - Otherwise, it should print:
     "[name] is playable within [time] with [players] people."

   E.g. Gloomhaven requires 1-4 players, and 30 minutes per player.

   If you call checkPlayable(5, 120), the method will print the following, as 5 is too many players:

   Not playable. Gloomhaven requires fewer players than you have.

   If you call checkPlayable(3, 60), the method will print the following, as 60 minutes is not enough time for 3 players:

   Not playable. Gloomhaven requires more time than you have.

   But if you call checkPlayable(2, 90), the method will print:

   Gloomhaven is playable within 90 minutes with 2 people.

Question 3. Files **[15 marks]**

This question uses the data contained the collection.txt file.

The file contains entries in a book collection. Each book uses *three* lines, and is in the following format:

```
[Book Title]           e.g.    Gideon the Ninth
[Book Author]                  Tamsyn Muir
[Rating] [Genre]               4.2 Science Fantasy
```

The title, author, and genre are all strings, and can all contain multiple words. The rating is a double.

Complete the displayCollection() method in the Program3 class, which should read in the file, and then print out each book as a single line (one line per book) following the format:

'[Book Title]' by [Book Author] ([Rating]; [Genre]

For example, the first three books will print out as follows (... indicates that the line continues, but will not fit on the handout).

```
'Gideon the Ninth' by Tamsyn Muir (4.2;  Science Fantasy)
'Harrow the Ninth' by Tamsyn Muir (4.4;  Science Fantasy)
'This Is How You Lose the Time War' by Amal El-Mohtar and Max Gladstone (...
```

Question 4. ArrayLists of Objects **[30 marks]**

This question concerns a program for tracking the sports at the Olympics, including their disciplines and team sizes.

There are two classes: Program4 and a Sport class. You must complete the Program4 class.

Program4 has a olympicSports field which contains an ArrayList of Sport objects.

The Sport class below defines Sport objects, which stores the name, discipline, and team size of Olympic sports events.

- The name and discipline are both strings.
- All sports will have a name, but some sports will have no discipline (ie: it will be null)
- The team size is stored as an integer. Solo sports have a team size of 1.

```
class Sport {
    private String name;
    private String discipline ;
    private int teamSize;

    /** Creates a new sport object with the given name, discipline , and team size */
    public Sport(String name, String discipline , int teamSize) {
        this.name = name;
        this. discipline = discipline ;
        this.teamSize = teamSize;
    }

    /** Gets the name of the sport */
    public String getName() { return this.name; }

    /** Gets the discpline of the sport (can be null) */
    public String getDiscipline () { return this. discipline ; }

    /** Gets the size of the team for this sport event */
    public int getTeamSize() { return this.teamSize; }
}
```

(Question 4 continued on next page)

(Question 4 continued)

(a) **[8 marks]** Complete the printSportsTable() method that will print every sport in the ArrayList according to the following format:

[Name] [Discipline] [Team Size]

- If the sport does not have a discipline, it should print "–" for that value.
- If the sport has a team size of 1, it should print "Solo", otherwise it should print the size

For example:

```
Archery                 --              Solo
Archery                 --                 3
Artistic swimming       Aquatics           2
Artistic swimming       Aquatics           8
```

Note: Some sports have multiple events for teams of different sizes. These are represented by extra line in the table (as shown above).

(b) **[12 marks]** Complete the findSport(...) method which has two parameters:

1. discipline, a string
2. teamSize, an integer

The method should search the list of sports and print the **names** of all sports that have matching disciplines **and** team sizes.

HINT: Some sports do not have a discipline. Remember to check for nulls.

E.g. findSport("Aquatics", 2) should print:

```
Artistic swimming
Diving
```

(c) **[10 marks]** (*This question is harder – feel free to skip it and come back to it later)

Complete the findSoloSports() method, which should find all of the sports in the ArrayList that have a team size of 1, and save them into a **new** Arraylist.

It should then return the new list.

Question 5. 2D Arrays **[25 marks]**

The Program5 class contains a program for a grid-based game where the user has to reveal tiles to increase their score without revealing a 0.

- The grid field is declared to hold a 2D array of ints, representing the tiles in play.
  - The grid is 5 x 5
  - The values are 0, 1, 2, or 3
  - The first index is the row, the second index is the column
- The rowSums field is an array that contains the sums of each each row (ie: rowSums[0] is the sum of row 0, etc)
- The colSums field is an array that contains the sums of each each column (ie: colSums[0] is the sum of column 0, etc)

During play, this information is presented to the user, with the sum of each row and column appearing around the grid, and the number of 0 tiles appearing in () next to it.

For example, if a row says "5 (0)", you know that it has no zeros, and will sum to 5 (i.e. a row of 1 tiles)
Likewise, if a column says "6 (3)", you know that it will have three 0 tiles and two 3 tiles.

Your task is to complete three methods that are needed to play the game.

(a) **[5 marks]** Complete the multiplyScore(...) method in Program5 so that it updates the score by multiplying it by the value at the specified location in the grid.

For example

- If the user's score is currently 1, and they click on a square that contains a 3, the score should become 3 ($1 \times 3 = 3$).
- Then, if they click on a 2, it should become 6 ($3 \times 2 = 6$)
- If the user clicks on a 0, their score becomes 0 (anything $\times 0 = 0$)

NOTE: Assume the first index is the row, and the second index is the column.

(b) **[10 marks]** Complete the doRowSums() method in Program5 so that it:

- Calculates the sum of every row in the grid
- Stores the sum in rowSums, in the appropriate location

(c) **[10 marks]** Complete the doColSums() method in Program5 so that it:

- Calculates the sum of every column in the grid
- Stores the sum in colSums, in the appropriate location

* * * * * * * * * * * * * * *