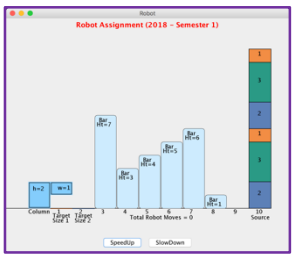
Programming Techniques (COSC1284)
Semester 1 (2018)
Assignment 1



Robot

Due: **Tuesday 27th March, 2018**
(Week 5 - 11:59 P.M. Midnight)

Assignment Type : **Individual**

Total Marks: **10 marks (10%)**

Assignment Objective

The objective of this assignment is to develop your programming and problem solving skills in a step-by-step manner, assisted with visualization.

Visualization helps you to easily identify any problems with your algorithm.

Students are expected to devise a strategy before attempting to code any part
(trial and error will cause you to go in circles).

The different stages of this assignment are also designed to gradually increase the complexity of the problem space.

What you need to do

Write a program to move the stack of blocks from source to a specified target location.

Movement of Robot arms, picking and dropping are controlled using the Robot methods described below.

For all sections below you are required to complete the control() method of the RobotControl class shown below.

The control() method will be automatically called when the program is run. RobotControl class has a reference r set to refer to the Robot object.

```
class RobotControl
{
     private Robot r;

     public RobotControl(Robot r)
     {
          this.r = r;
     }

     public void control(int barHeights[], int blockHeights[])
     {
          run(barHeights, blockHeights);
      }

     public void run(int barHeights[], int blockHeights[])
     {
          // Write your code here i.e. r.up()
     }

}
```

How to run the program

1. Select the project **Robot_Start_Up**

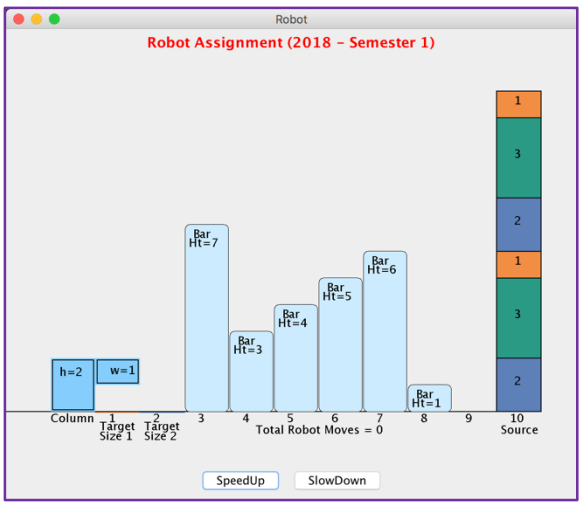
2. Click on the Run menu and choose Run
   (NOTE: These are example configurations to help you test your code, your Robot should work for any configurations that are supplied.)

```
****************** ROBOT TEST HARNESS MENU ******************

IMPORTANT: YOU MUST PASS EACH TEST BEFORE MOVING ON TO THE NEXT TEST
For example to be awarded marks in Part C, you must first pass all
scenarios
for Part B.

IMPORTANT: ANY TEST AFTER A FAILED TEST WILL NOT BE ASSESSED.
*****************************************************************

1. Stage A Test 1 - bars = 777777, blocks = 3333

2. Stage B Test 1 - bars = 734561, blocks = 3333

3. Stage B Test 2 - bars = 137561, blocks = 3333

4. Stage B Test 3 - bars = 171717, blocks = 3333

5. Stage B Test 4 - bars = 137652, blocks = 3333

6. Stage C Test 1 - bars = 734561, blocks = 231231

7. Stage C Test 2 - bars = 222222, blocks = 321113

8. Stage C Test 3 - bars = 444444, blocks = 311123

9. Stage C Test 4 - bars = 777777, blocks = 222111111

10. Stage C Test Custom (user supplies bar / block config)

11. Run All Tests (1 - 9)

0. Exit Test Harness

IMPORTANT:
      YOU MUST PASS EACH TEST BEFORE MOVING ON TO THE NEXT TEST
      IMPORTANT: ANY TEST AFTER A FAILED TEST WILL NOT BE ASSESSED.

Enter selection:
```

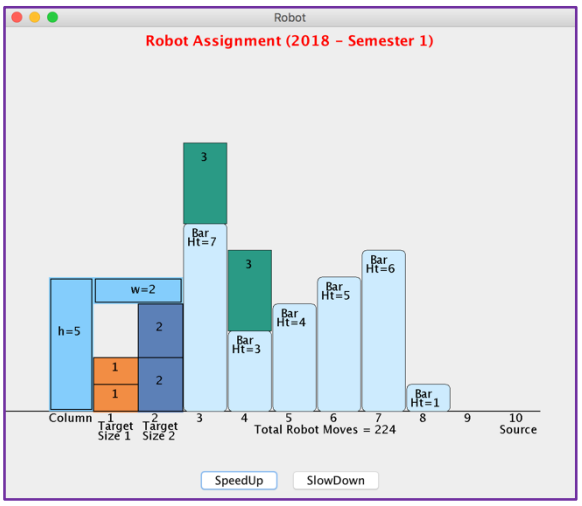
Restrictions (You can assume the following when supplying configurations)

- The height of individual blocks can be between 1 and 3.
- The total height of all blocks must not exceed 12.
- The height of individual bars must not exceed 7
- The target for all blocks is indicated by their size.
- Blocks of size 1 should be placed in the first column labeled Target 1.
- Blocks of size 2 should be placed in the second column labeled Target 2.
- Blocks of size 3 should be placed on top of the bars beginning with column 3.
- Only one block can be placed on a bar, so after a block has been placed on column 3, then the next block of size 3 should be place on column 4 etc.
- Blocks must be place directly on their targets, you cannot drop the block from a height and let it fall into place.

Start



After running



How to control the robot

You control the Robot by calling an appropriate method on the Robot such as 'up', 'down' etc.

The methods of Robot class:

Methods	Description
up() | down()	The height (**h**) of the main arm can be changed by 1 unit using methods ***up***() and ***down***(). The height (**h**) of this arm must lie in the range 2 to 13, and is initially set to 2.
extend() | contract()	The second arm can be moved horizontally by 1 unit using ***extend***() or ***contract***(). The width (**w**) must lie in the range 1 to 10, and is initially set to 1.
raise() | lower()	The depth (**d**) of the third arm can be changed by 1 unit using ***lower()*** or ***raise*()**. The depth cannot be less than 0 but must be less than the height of the main arm (***d < h***). In the initial position depth d is set to 0 (it is not visible).
pick()	An item can be picked from the top of the stack of blocks at source using ***pick().***
drop()	It can be dropped at the top of the stack of blocks using ***drop()***
speedUp() | slowDown()	Robot class has two other methods **slowDown**(int factor) and **speedUp**(int factor) to adjust the time between moves by the specified factor.

If the code you write for the Robot results in hitting an obstacle or violates the business requirements of the program, then the program will terminate immediately with an appropriate Error Message.

The requirements for each part of the assignment are detailed in the next section.

Students are advised to tackle each of the requirements in turn – when you have completed the code for one stage of the assignment (e.g. Requirement A) then you can comment that section out and proceed to the next stage.

Important:

> The previous point is so you can return to working code if when trying to working on the next part, you get confused and want to return to a point where your code was working for the previous part.

However, as you proceed, the code that your write for Part B should also work for Part A. The code that you write for Part C should work for Parts A & B also. You should not write different code for each of the tests or sections.

To be assess for Part B, you must have functional code for Part A (this does not include achieving the minimum moves, just that should complete the Part A test before moving on to Part B. The same applies for Part C, it will only be assessed if you have Parts A & B completing all the test scenarios.

Also note that the test scenarios are examples only to help you in testing your code. We should be able to provide any custom configuration and your code should still complete the task successfully.

You should leave the last **FUNCTIONAL** stage you have completed uncommented and comment out previously completed stages (don't delete them just in case).

Requirements

Your solution must use Java 1.7 or above and be compatible with the environment supplied in the computer laboratories.

You should make sure that your solution includes examples of constants to represent fixed values in your program.

You should NOT use streams from Java 1.8. This assignment is geared at assessing your ability to work with developing an algorithm for working with arrays via the use of loops and if/else statements.

You can use your own computers to develop your assignment, but you must make sure that your assignment can be compiled and run on the environment supplied by RMIT My Desktop.

The specification for each task is set out below. Note these are cumulative so that a requirement set for Part C also includes the previous requirements from Parts A – B except where the new requirement contradicts the older one. This means that any code that your write for Part B should also complete the Part A scenario. Any code that your write for Part C must also successfully compete the scenarios in Parts A & B.

Methods	Description
Part A	There are six bars are of height 7. There are four blocks of height 3.
Part B	There are six bars of varying height determined by the configuration supplied. The bar heights supplied must be in the range 1 – 7. There are four blocks of height 3.
Part C	There are six bars of varying height determined by the configuration supplied. The bar heights supplied must be in the range 1 – 7. The number and height of the blocks will be supplied by configuration. Each block can be a different height, but values supplied must be in the range 1 – 3. In addition the total height of all the blocks cannot exceed 12.

Part A: There are six bars are of height (7) and four blocks are of height (3).

Complete the ***control()*** method of the RobotControl class to move all the blocks to the respective targets as indicated in the screen shot below.

The target is determined by the height of the block being moved.

- All blocks of size 3 must be placed on top of the bars.
- Blocks of size 3 should be placed on top of the bars beginning with column 3.
- Only one block can be placed on a bar, so after a block has been placed on column 3, then the next block should be place on column 4 etc.

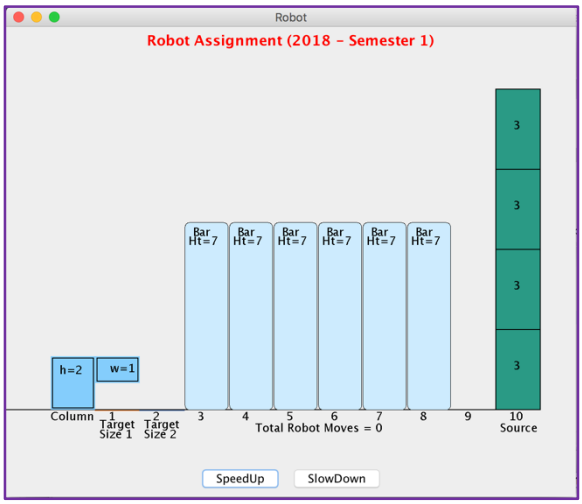
Note: During assessment the actual configuration that will be supplied is presented below.

In Parts B & C the configuration will change.

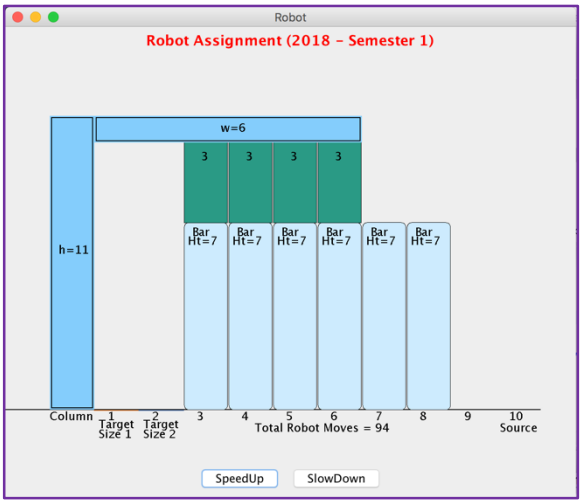
See the respective sections for more detail.

Actual configuration: ***777777 3333***

Start



After running



Part B: There are six bars of varying height and there are four blocks height 3.

Complete the ***control()*** method of the RobotControl class to move all the blocks to the respective targets.

Allow the user to supply the heights of the 6 bars as command line arguments.

There are four blocks of height (3).

The target is determined by the height of the block being moved.

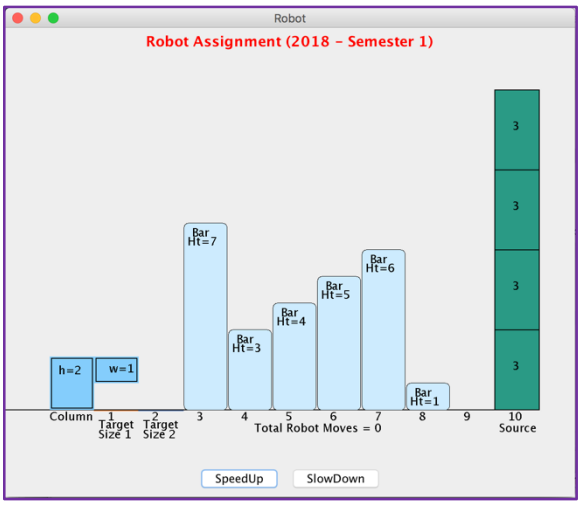
- All blocks of size 3 must be placed on top of the bars.
- Blocks of size 3 should be placed on top of the bars beginning with column 3.
- Only one block can be placed on a bar, so after a block has been placed on column 3, then the next block should be place on column 4 etc.

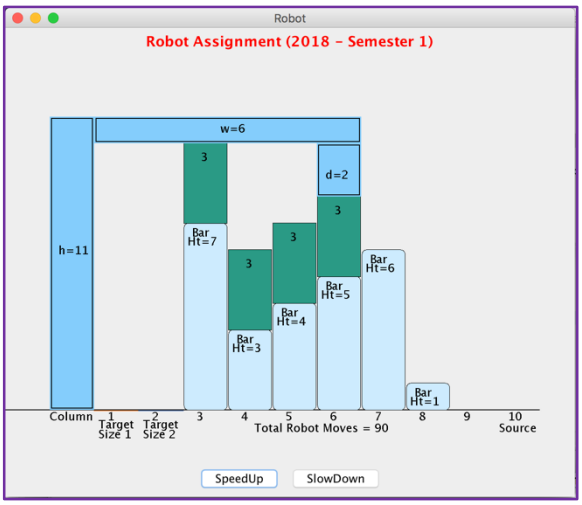
Note: During assessment the actual configuration of the bar and the number and heights of the blocks will change so your program needs to be able to complete successfully with different configurations.

This means that you should thoroughly test each stage of your assignment with multiple configurations in order to be confident that your code will not fail when an alternate configuration is supplied at assessment time.

Sample Configuration: ***734561 3333***

Start **After running**





Part C: There are six bars of varying height and the number of blocks as well as the heights of each block may vary.

Complete the ***control()*** method of the RobotControl class to move all the blocks to the respective targets.

Allow the user to supply the heights of the 6 bars as well as the number and height of the blocks as command line arguments.

Individual block heights may vary but must lie within the range of 1 – 3.

The total block heights must not exceed 12.

The target is determined by the height of the block being moved.

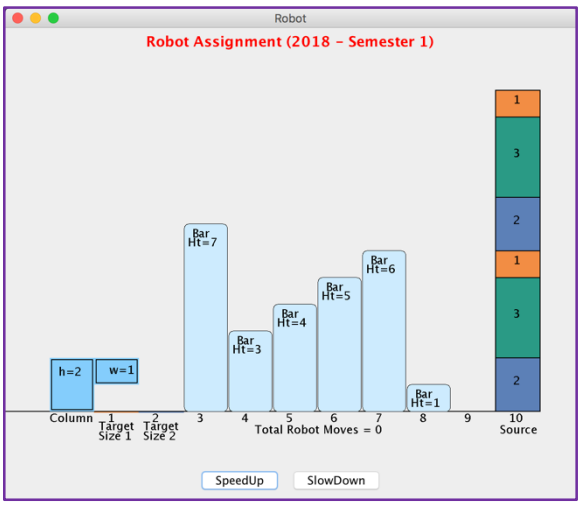
- All blocks of size 1 must be placed in column 1 labelled Target 1
- All blocks of size 2 must be placed in column 2 labelled Target 2
- All blocks of size 3 must be placed on top of the bars.
- Blocks of size 3 should be placed on top of the bars beginning with column 3.
- Only one block can be placed on a bar, so after a block has been placed on column 3, then the next block of size 3 should be place on column 4 etc.

Note: During assessment the actual configuration of the bar and the number and heights of the blocks will change so your program needs to be able to complete successfully with different configurations.

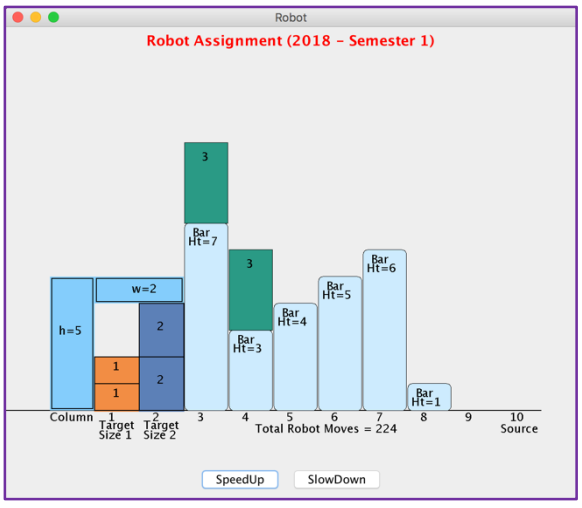
This means that you should thoroughly test each stage of your assignment with multiple configurations in order to be confident that your code will not fail when an alternate configuration is supplied at assessment time.

Sample Configuration: ***734561 231231***

Start



After running



Assessment

The assignment is designed to test multiple competencies required for successful software development. The assessment evaluates all skills required not just the code itself. You will be assessed on your ability to comply with requirements as set out in this document.

The assessment will include competencies in:

1. **Business Requirements**: such as adherence to a style guide, documentation of your program and compliance with processes such as submission of your work prior to assessment.

2. **Java Programming**: in Java demonstrating a mastery of the language and the accepted conventions for developing applications.

3. **Software Development Lifecycle**: practices such as an iterative approach to development that includes thorough testing of your solution.

4. **Algorithm Development**: that demonstrates that you have a clear idea of the logic required to complete the necessary tasks.

   The algorithm should aim to be efficient. For example the blocks should be moved to their respective targets in the minimum number of moves.

Note: You are expected to use loops and nested loops to avoid repeating the code for moving each single block for all parts of the specification.

Due Date

You must submit your solution to Canvas by the due date.

This due date of the assignment is listed on the first page of this specification. You must submit your final version prior to this date/time to avoid incurring late penalties.

Any late submissions will attract a 10% penalty per day.

If you are unable to submit by the due date you **ARE REQUIRED** to provide additional documentation such as a medical certificate as per university policies.

Extensions **WILL NOT BE GRANTED** without supporting documentation.

The details of the extension requirements and an outline of the policies and procedures can be found here:

http://www1.rmit.edu.au/students/assessment/extension

Late Submission Period

There is a late submission period of 5 days for this assignment. Late submissions that are received before the end of this late submission period will attract a late penalty of 10% per day (or part thereof) of the total marks awarded on the assignment, unless an extension has been granted by the school or as part of an existing EAA provision (see below for details).

Extensions (Up to 7 Days)

Submissions that are received after the late submission period expires will not be assessed.

The Teaching Assistant (Rodney Cocker) can only grant extensions of up to 7 days provided the correct form and documentation has been submitted.

If you require an extension you must apply prior to the due date by filling out an 'extension of time' request and attaching the requisite documentation.

As outlined here:

http://www1.rmit.edu.au/students/assessment/extension

The form and documentation should be e-mailed to rodneyian.cocker@rmit.edu.au.

Extensions (Greater than 7 Days)

For extensions of greater than 7 days or for those with EAA provisions organised by the DLU:

http://www.rmit.edu.au/students/specialconsideration

If you are a student who has EAA provisions organised by the DLU then you must negotiate any extension you may require with the instructor well in advance of the submission deadline (at least 72 hours in advance of the on-time submission deadline would be ideal).

All questions regarding this specification should be directed to the Assignment 1 forum.

Any requests for assistance regarding implementing the requirements set out in this assignment or problems you are having with your program should be directed to the Assignment 1 forum.

Coding style / structure

You are required to adhere to all standard java coding conventions, which include correct and consistent indentation of code, use of appropriate identifiers for variables and comments describing the main logic program.

Your code should include examples of:

- variables to represent the changing state of the program whilst it is running.
- constants to represent fixed values that do not change throughout the running of the program.
- code blocks refactored into methods to reduce code repetition.
- commenting that contributes to an understanding of the code.
- unused code should be removed and not just commented out in your final submission.

However, avoid commenting code that is obvious to the reader. You can also make the program structure clearer by subdividing your code into blocks or methods.

An example of a good comment:

```
// Find the maximum bar height
for (int i = 0; i < barHeights.length; i++)
{
	if (barHeights[i] > maxBarHeight)
	{
		maxBarHeight = barHeights[i];
	}
}
```

An example of a bad comment:

```
// declare an int value for storing the current height of the
arm
int currentArmHeight = 0;

// declare an int value for storing the current height of the
arm
int x = 0;
```

When awarding marks, the clarity of code and efficiency of code will also be considered. Once, you have made your code to work you are expected to refactor your code removing any duplicated, redundant or misleading code and improving the efficiency by removing unwanted moves.

Marking Guide

Please review this carefully as this will be used by the marker when marking your assignment.

Please ensure that the code you submit does not contain any compilation errors

Code submitted with compilation errors will incur a marking penalty of 2 marks.

The marking guide can be found in the assignment section of Canvas.

Submission Instructions

Where you do make use of other references, please cite them in your work.

Note that you will only be assessed on your own work so the use of third party code or packages is not permitted.

You should submit a single Java file (RobotContro.java) that contains your code for controling the Robot.

This is the file that is provided in the startup project to which you will have added your programming logic.

You may submit your assignment more than once, the last submission will be the one used for assessment.

You should submit your assignment each time you complete a major section of the assignment. This will make sure that you are familiar with the process and don't have any last minute issues with your final submission.

Startup notes for Eclipse

1. **Installation instructions for eclipse users:**

a. Download the archive file robot_startup_code.zip.

b. Choose import option in file menu item.

c. Expand the **General** tab and select **Existing project into workspace** option.

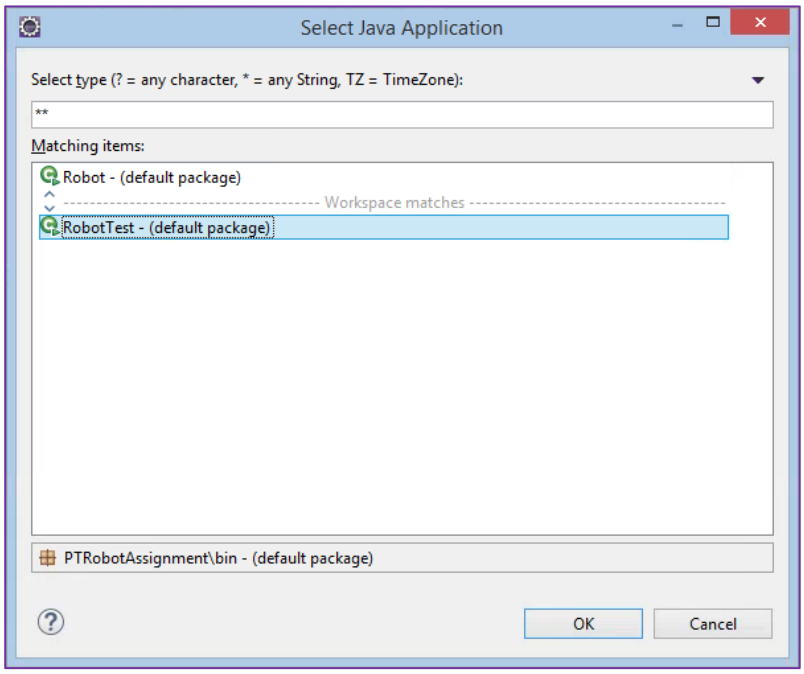
d. Select the radio button **archive-file**
(Note if using a Mac and when you dowload the startup code, some Mac are configured to automatically uncompress the zip file. In this instance you should choose Exisiting Project, not the archive option.)

e. Click Browse and navigate to the downloaded file, and press **Finish.**

f. Click on the Run Menu and Choose Run.

g. If prompted choose to run as a Java Application.

h. When prompted Choose RobotTest.



i. The control method in RobotControl.java has some code to get you started.

j. Test the application.

k. Rename your project to your student number + _ + A1
(Right-click on the project, choose Refactor, then Rename)

e.g. s123456_A1