CSY1020: Problem Solving & Programming
Assignment 1: Programming (Java) (50%)

Starting Date: **Week beginning 5th October 2020**
Finish Date: **Sunday, 17th January 2021, by 23h59.**

E-Submission through Turnitin on NILE as TWO separate WORD doc/x.
Document 1 = Report & Document 2 = Appendix (Full source code listing).
To do this go to the NILE site for this module and under assignments use the link labelled 'Submit your work', then the 'Programming Report', 'Appendix'. Please remember this is an individual assignment and all assignments will be submitted through turnitin.

Brief:
Produce a technical report and accompanying Java GUI application, simulating control of a baby being navigated around a maze with walls using left, right, up and down buttons/keys to get from one side of the maze to the other.

The problem is designed to be open within the rules below, to enable you to develop **your solution(s) to the problem**.

Rules (Basic) Create a simulation of the baby moving around the maze, where:

- The baby can only move anywhere within the maze where is no wall or perimeter i.e the aim is to move the baby around a maze from side of the maze to the other
- If the baby touches the wall or the perimeter it remains in the same position
- The baby must move one whole 'white' block at a time every time a movement button (via a direction button (<, > v ^)) /key is pressed (when movement is possible).
- The solution must use the scenario provided

Rules (Intermediate and advanced) Create a simulation of a baby moving around the maze, where:

- Your solution must still use the scenario provided (all the basic features above).
- Add appropriate extra features to the solutions, e.g. a) reconfigure the maze from vertical walls to horizontal walls b) the system can move the baby from one side of the maze to the other c) Adjust the speed to get the baby from one side of the maze to the other (a, b or c for **Intermediate,** a, b & c for **Advanced**).
- For higher grades on the solution part of the assignment see the marking scheme at the end of the brief.
- Your mazes cannot be changed during execution and the layout and all changes should still meet the criteria of **Rules (Basic**).

Attempt to emulate the following application:

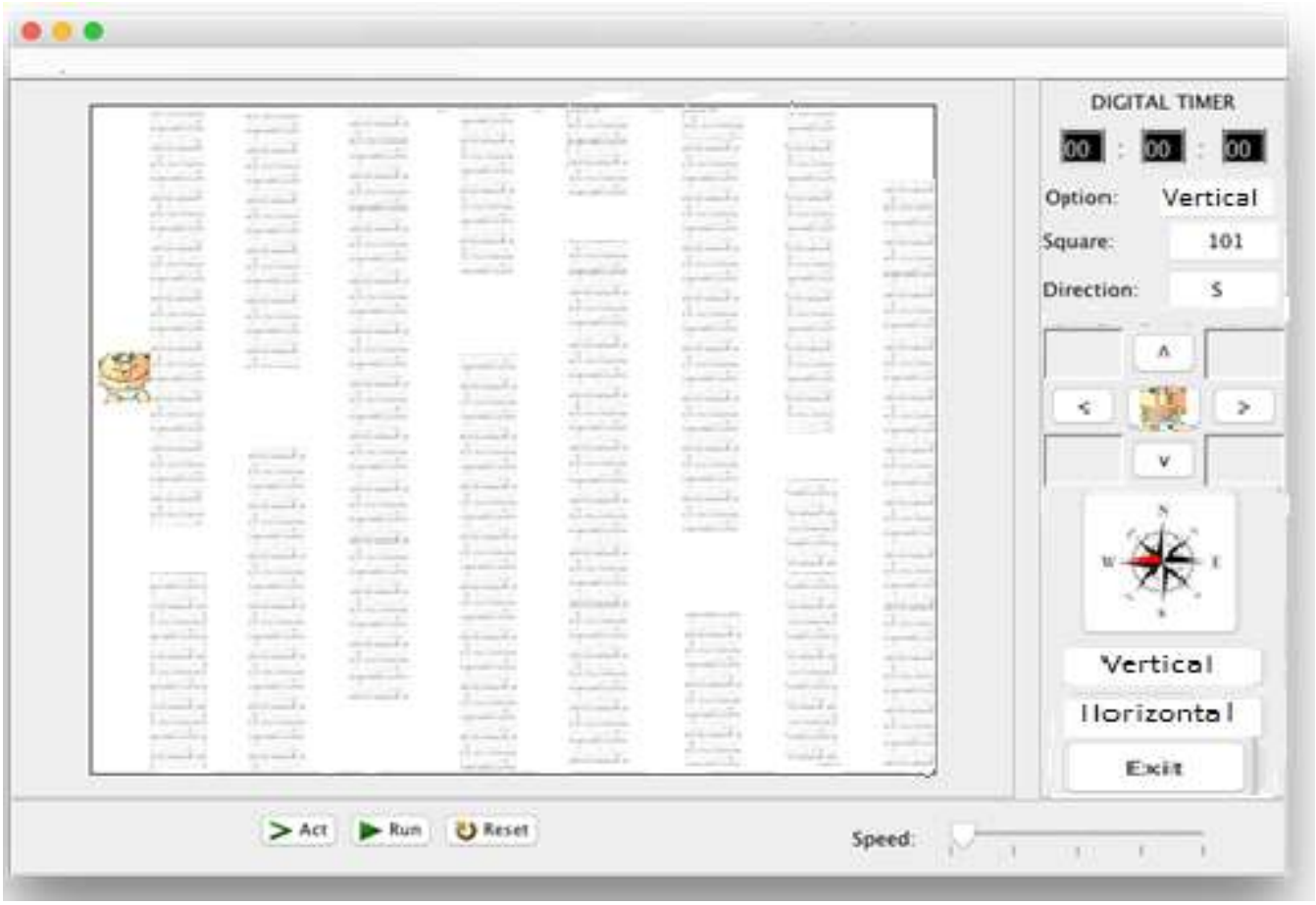

Figure 1: Problem

The images (e.g the baby and the compass) for the system that can be seen in the snapshot above can be downloaded from the assignment folder on NILE

System Requirements

Essential (Graphical User Interface):

- 13 x 16 grid of **JButton**'s or Icon's.
- 3 **JButton**'s for the game options '*Vertical'* ,'*Horizontal'* and 'Exit'
- 3 **JButton**'s for 'Act', 'Run' and '*Reset*'.
- 9 **JButton**'s for '*Forward >*', 'Backwards <', 'Up ^', 'Down v' should move the baby in the appropriate direction by one square for each press
- The compass icon (**JButton)** should illustrate the current direction for the baby.
- **3 JLabel**'s for 'Digital Timer' and 'Option', '*Square*' and '*Direction*'.
- **3 JTextField**'s for the current 'Option', *'Square'* and *'Direction'* of the baby. The 'Option' will be set to either the text 'Vertical' or the text 'Horizontal' depending on which of the 'Vertical' or 'Horizontal' button have been pressed. The 'Square' will be set to the current position of the baby i.e it will be in the range 0 to 207 where 0 is the top left square and 207 is the bottom right square. The 'Direction' will be set to the direction the baby was going from the previous square i.e N, S, E or W.
- 3 **JLabel's** for the 'DIGITAL TIMER' and the two ':', with 3 **JTextField'**s for the hours, minutes and seconds.
- Create a **JFrame** application, which opens to the set size (825 * 585).

Additional (Functionality & Complexity):

- Application icon for the **JFrame** used.
- The 'Run' **JButton** should show the baby moving through the maze by itself. The code behind this button should not be fixed code but done by your code checking if the squares around the baby are the wall or the perimeter or a gap in the wall to get through.
- Digital Timer should start and stop when *Run* is pressed and stopped when a baby reaches the other side of the maze from where it started. When the baby reaches the opposite side of the maze from where it started the time it took to do so is displayed from the digital timer.
- The 'Reset' **JButton** should clear/reset the application to its starting/default opening state which is as shown in the image above
- The 'Act' **JButton** should step through the above 'Run' sequence one move at a time.
- A Speed **JSlider** to increase/decrease the speed of the baby moving when the 'Run' button has been pressed
- The 'Vertical' **JButton** should display a vertical configurations of bricks as shown in the image above – in this case the aim is for the baby to get from the left of the maze to the right of the maze
- The 'Horizontal' **JButton** should display a horizontal configurations of bricks – in this case the aim is for the baby to get from the top of the maze to the bottom of the maze
- The 'Exit' **JButton**'s should will exit and close down the system
- Discuss and implement the different options for the 3 configurations.
- Create a **JFrame** application, which is not resizable.
- Discuss the possibilities for incorporating intelligence/checks for whether moves are valid.
- Implement intelligence/checks for whether moves are valid for the **Run** and **Act** buttons

The application **must** be demonstrated (see below) by a video recording in kaltura. The source code file containing the **main()** method and the compiled byte code **class** files should be named as follows:

DELIVERABLES
All requirements (both A & B) **MUST** be delivered to achieve a pass grade for this assignment.

There will be 2 separate labelled submission boxes on NILE to submit deliverables A and B below.

A) Technical Report
Document 1 = Report
The report should be **2500 words (minimum)** and should be structured as follows:

- Title Page
- Table of contents (to include List of Tables and List of Figures)
- **Link to your video**
- **1. Design** for the task in either pseudo-code or flowchart form (include GUI design fully labelled). Include a routine suitable to successfully complete the task.
- **2. Implementation** (Explanation and Morphology of the code and resulting GUI).
- **3. Testing** (A test plan for the task with supporting evidence).
- **4. Conclusion & Recommendations** (which should include a critical appraisal of the strengths and weaknesses of your design and what improvements could be made; also list what you have implemented and what you have not implemented).
- **References** (Using **Harvard** referencing).
- **Bibliography** (If needed, using **Harvard** referencing).

B) Code Listing
Document 2 = Appendix

- Appendices (**The Appendices should include a commented file listing of all source code inside a word file**).

It is expected that your report should make appropriate use of screen shots, particularly during the Implementation and Testing sections.

Viva/demonstration
You must provide a video demo in kaltura of your assignment and provide link in your report. The demo should be 10-15 minutes long (no longer than 15 minutes), and should cover all of your work in a logical way. Your voice needs to be clear for the marker to hear. Please include a walkthrough of using the software and emphasise the key features. You may be called in for a viva-voce should there be any doubts on the originality (plagiarism aspects) of your submission. You must demonstrate what works and state what doesn't work.

Use Kaltura for recording your video demonstration. For instructions on using Kaltura go to the following website:

https://libguides.northampton.ac.uk/learntech/students/nile-guides/kaltura

To record your demonstration you can use Kaltura's software available from here:

https://northampton.mediaspace.kaltura.com/

To record your demo select Add New > Capture Recorder.

It is your responsibility to ensure that the permissions of your video are correct otherwise you will get a maximum grade of F. Likewise, if you do not provide a link or your link is invalid then you will also get a maximum grade of F.

Please do not use other ways to create a video such as youtube or zoom because we are not responsible for external sites

Assessment Breakdown

Assessment Criteria:
Design of System - 15%
Implementation - 45%
Testing - 15%
Code Layout and Documentation (5%)
Quality of Report – 20%

Useful resources:
Rough Guide to Harvard Referencing:
http://studyskillshub.files.wordpress.com/2013/10/harvardrefquickguide2.pdf

---

Learning Objectives.

a) Appreciate the principles and practice of analysis and design in the construction of robust, maintainable programs, which satisfy their specifications.
b) Design, write, compile, test and execute straightforward programs using a high level language; appreciate the principles of programming.
c) Appreciate the need for a professional approach to design and the importance of good documentation to the finished programs.

d) Use an appropriate programming language to construct robust, maintainable programs, which satisfy their specifications.
e) Design, write, compile, test and execute programs taking into consideration principles of programming.
f) Apply skills to enable the solution of problems with the construction of appropriate algorithms and a computer program.

Please use the rubric at the end of this document to help you with the assignment and to help with self-feedback.

Personal Development & Key Skills (for your PDP)
This assignment provides an opportunity to add to your personal development portfolio as indicated below:

PDP: Personal development elements (for your PDP), this assignment provides an opportunity to add to your personal development portfolio the following:

- Problem-solving – the whole assignment is concerned with this area;
- Use of number – tasks involve manipulating numbers to solve a problem;
- Managing the Learning process – Successfully completing the assignment on time.
- Communication skills – presenting your work and critically appraising it in a clear form.

Key Skills	**Y/N**
1 Managing the Learning Process: Ability to evaluate learning styles, identify strategies for approaching study tasks, manage and organise oneself taking responsibility for decision-making, target setting and delivery of action.	**Y**
2 Communication Skills: The ability to express, discuss and present knowledge, ideas and viewpoints to a variety of audiences with confidence and clarity.	**Y**
3 Groupwork: The ability to work harmoniously and productively as a member of a group in a variety of roles, demonstrating an awareness of group dynamics, appropriate inter personal and interactional skills.	**N**
4 Information Skills: The ability to identify information needs, access and evaluate a range of relevant sources, organise and use information efficiently and effectively for both academic and professional purposes.	**Y**
5 Problem Solving: The ability to identify problems and to apply concepts, principles and techniques in order to generate solutions, choose between alternatives and take appropriate action.	**Y**
6 Use of IT: The ability to effectively use key information technology and appropriate software to assist in the learning process through research and retrieval, communication and manipulation of information in various forms.	**Y**
7 Application of number: The ability to understand, interpret and use numerical and graphical information accurately and effectively.	**Y**

Grading Criteria:
This Standard Front Sheet gives a clear indication of how the grade for this assignment is achieved. In general the following criteria will act as an overall guide to what you should expect:
A **bare pass (D)** will involve incorporate most of the 'Basic System Requirements:' above and accompanying technical report covering all appropriate sections.
A **good pass (B to C)** will incorporate all of the 'Basic System Requirements:' above and accompanying technical report covering all appropriate sections.
A **very good pass (A)** will incorporate most of the 'Additional System Requirements (functionality & complexity):' above and accompanying technical report covering all appropriate sections.

Viva/demonstration
This is a **COMPULSORY** activity, which is considered an essential element of this assignment – you will not pass the assignment **if you don't attend – a maximum grade of an F** will be given to any assignment submission that has not been demonstrated during the allocated session above.

Technical Report
All requirements (both A & B) **MUST** be delivered to achieve a pass grade for this assignment. A) Technical Report - Document 1 = Report & Document 2 = Appendix – Code Listing. If the Technical Report is not submitted a G grade will be awarded and if submitted up to 1 week late a maximum of D- will be awarded.

Notes on achieving the most from this assignment and where marks can be lost:

1. Carefully read and read again the assignment brief. Following the instructions given carefully and check the front sheet to see where marks are allocated.
2. Always write in the third person i.e. **NO** *I* thought this, *I* did this etc. If you need to refer to yourself use "The author felt" etc.
3. The application **must** be demonstrated by video. Anyone not demonstrating receives a maximum of F.
4. If an assignment is not submitted (even if the application was demonstrated) a mark of G is given.
5. The brief states that "The source code file containing the **main()** method and the compiled byte code class files should be named as follows: **BabyMaze.java** & **BabyMaze.class**".
6. A Title Page should be attached to the assignment.
7. **Implementation** should be a commentary of the code explaining how the design has been applied. Screen shots should shows snippets of the code and the outcome in the GUI.
8. The FULL source code in the Appendix should have a header indicating the usual information e.g.

```
/**
Program:    Assignment 1: Application – Baby Maze
Filename:   BabyMaze.java
@author:    © Gary Hill (200WXYZ)
Course:     BEng/BSc/HND Computing Year 1
Module:     CSY1020 Problem Solving & Programming
Tutor:      Gary Hill
@version:   2.0 Incorporates Artificial Intelligence!
Date:       23/11/20
*/
```

9. Meaningful variable names should be used in code e.g. jBRotate (for the rotate JButton).
10. Your code should have meaningful comments e.g. //button to move object down/south.
11. The aim of the **Conclusion & Recommendations** section is to demonstrate that you can develop your own conclusions and recognize the limitations of what you have produced. To achieve this you should clearly indicate:
    - the aim and objectives of the assignment from the assignment brief;

- which objectives and items of general and advanced functionality have been met;
- which have not and how you think they could have been solved given more time;
- what are the limitations of what you have been asked to produce;
- how would your approach differ given the opportunity to do the assignment again?

12 **Design:** The design should, ideally, be as explicit as possible so that there is no ambiguity (only one way that it could be read). This means avoiding instructions that are too general such as "move forward". Does the routine check which direction is forward, as forward can be in any direction depending on the amount of rotation, also not only do you need to know which direction, but also the current location. It is often better to take the general instruction and split it into a number of very specific simple instructions. Include also the requirements for the robot.

13 **Implementation:** The implementation should be a commentary of the code explaining how the design has been applied. Screen shots should shows snippets of the code and the outcome in the GUI, these should also demonstrate/illustrate key areas of the code e.g. nested for loop. Quality of coding should also be considered e.g. use of meaningful variables, meaningful comments, neatness/readability, attempt to use a coding convention, header/title etc.

14 **Testing:** Testing here, is **not just about proving that you design worked but also, what did not work or what happens if an unexpected or unusual event occurred?** Therefore, even if the design is not a fully working design it is still important to include some test data to show what would happen with your design. Also, include any criticism of the design or **improvements** that were identified.

Remember to Reference your Software Code

Author (Year) *Title of Program* (Version Number) [format type] (computer program, software or code, Place of publication: publisher (if available). Available from: URL (if online).

E.g.

In Text or Code: (Eclipse, 2014)

In Reference list:
Eclipse (2014) Eclipse Lunar (Version 4.4.1) [Software] Eclipse.org. Available from: http://www.eclipse.org/downloads/ [Accessed 12 Feb 2014]

1 Coding Conventions

A set of coding conventions should be followed throughout, which help in the creation of code which is more efficient, maintainable, robust and most importantly – readable. The following guidelines should be used:

Naming Rules:
Variables were prefixed with lowercase mnemonic indicating nature of object. E.g. jBVariable(JButton), iconCar(ImageIcon), fVariable(float), dVariable(double), boxVariable(Box), bVariable (boolean), jTHello (JTextfield), jLDirection (JLabel), nSquare (int) bgVariable (BranchGroup) etc.

Align brackets:
By default eclipse places starting bracket for a code construct at end of the starting line, and aligns the closing bracket with the first column of the construct. This was changed to align

the starting and ending brackets on the same columns, so that a reader can clearly identify the borders for a code construct. E.g.:

```
for (int i=0; i<20; i++)
{
      // … Place code here
 }
```

Code Indentation:
A practice often ignored by most programmers, indenting is a good practice that helps immensely with code readability and debugging.

Commenting:
All important areas of the code should be commented, clearly explaining the functionality. As with some of the techniques above, this helps with code readability, maintenance and debugging.

2 References

Eclipse Foundation (2014). Eclipse SDK 4.4.1 [online]. Available from: http://www.eclipse.org [Accessed 12 Feb 2014]

Sun Microsystems (2014a) Lesson: Exceptions. Available from: http://java.sun.com/docs/books/tutorial/essential/exceptions [Accessed 20 may 2014].

Wikipedia (2014) Code Refactoring [online]. Available from: http://en.wikipedia.org/wiki/Refactoring [Accessed 12 May 2014].

Notes on achieving the most from this assignment and where marks can be lost:

15 Carefully read and read again the assignment brief. Following the instructions given carefully and check the front sheet to see where marks are allocated.

16 Always write in the third person i.e. **NO** *I* thought this, *I* did this etc. If you need to refer to yourself use "The author felt" etc.

17 The application **must** be demonstrated. Anyone not demonstrating receives a maximum of F.

18 If an assignment is not submitted (even if the application was demonstrated) a mark of G is given.

19 The brief states that "The source code file containing the **main()** method and the compiled byte code class files should be named as follows: **BabyMaze.java** & **BabyMaze.class**".

20 **Implementation** should be a commentary of the code explaining how the design has been applied. Screen shots should show snippets of the code and the outcome in the GUI.

21 The FULL source code in the Appendix should have a header indicating the usual information e.g.

```
/**
Program:    Assignment 1: Application – Baby Maze
Filename:   BabyMaze.java
@author:    © Gary Hill (200WXYZ)
Course:     BEng/BSc/HND Computing Year 1
Module:     CSY1020 Problem Solving & Programming
Tutor:      Gary Hill
@version:   2.0 Incorporates Artificial Intelligence!
Date:       23/11/20
*/
```

22 Meaningful variable names should be used in code e.g. jBRotate (for the rotate JButton).

23 Your code should have meaningful comments e.g. //button to move object down/south.

24 The aim of the **Conclusion & Recommendations** section is to demonstrate that you can develop your own conclusions and recognize the limitations of what you have produced. To achieve this you should clearly indicate:

- the aim and objectives of the assignment from the assignment brief;
- which objectives and items of general and advanced functionality have been met;
- which have not and how you think they could have been solved given more time;
- what are the limitations of what you have been asked to produce;
- how would your approach differ given the opportunity to do the assignment again?

25 **Analysis:** It is expected in the analysis that you will identify issues such as for example:

- What inputs are needed and the form of the inputs?
- What is the form of the output or outputs?
- What rules are needed?

26 **Design:** The design should, ideally, be as explicit as possible so that there is no ambiguity (only one way that it could be read). This means avoiding instructions that are too general such as "move forward". Does the routine check which direction is forward, as forward can be in any direction depending on the amount of rotation, also not only do you need to know which direction, but also the current location. It is often better to take the general instruction and split it into a number of very specific simple instructions. Include also the requirements for the robot.

27 **Implementation:** The implementation should be a commentary of the code explaining how the design has been applied. Screen shots should shows snippets of the code and the outcome in the GUI, these should also demonstrate/illustrate key areas of the code e.g. nested for loop. Quality of coding should also be considered e.g. use of meaningful variables, meaningful comments, neatness/readability, attempt to use a coding convention, header/title etc.

28 **Testing:** Testing here, is **not just about proving that you design worked but also, what did not work or what happens if an unexpected or unusual event occurred?** Therefore, even if the design is not a fully working design it is still important to include some test data to show what would happen with your design. Also, include any criticism of the design or **improvements** that were identified.

Remember to Reference your Software Code

Author (Year) *Title of Program* (Version Number) [format type] (computer program, software or code, Place of publication: publisher (if available). Available from: URL (if online).

E.g.

In Text or Code: (Eclipse, 2014)

In Reference list:
Eclipse (2014) Eclipse Lunar (Version 4.4.1) [Software] Eclipse.org. Available from: http://www.eclipse.org/downloads/ [Accessed 12 Feb 2014]

1 Coding Conventions

A set of coding conventions should be followed throughout, which help in the creation of code which is more efficient, maintainable, robust and most importantly – readable. The following guidelines should be used:

Naming Rules:
Variables were prefixed with lowercase mnemonic indicating nature of object. E.g. jBVariable(JButton), iconCar(ImageIcon), fVariable(float), dVariable(double), boxVariable(Box), bVariable (boolean), jTHello (JTextfield), jLDirection (JLabel), nSquare (int) bgVariable (BranchGroup) etc.

Align brackets:
By default eclipse places starting bracket for a code construct at end of the starting line, and aligns the closing bracket with the first column of the construct. This was changed to align the starting and ending brackets on the same columns, so that a reader can clearly identify the borders for a code construct. E.g.:

```
for (int i=0; i<20; i++)
{
        // … Place code here
 }
```

Code Indentation:
A practice often ignored by most programmers, indenting is a good practice that helps immensely with code readability and debugging.

Commenting:
All important areas of the code should be commented, clearly explaining the functionality. As with some of the techniques above, this helps with code readability, maintenance and debugging.

2 References

Eclipse Foundation (2014). Eclipse SDK 4.4.1 [online]. Available from: http://www.eclipse.org [Accessed 12 Feb 2014]

Sun Microsystems (2014a) Lesson: Exceptions. Available from: http://java.sun.com/docs/books/tutorial/essential/exceptions [Accessed 20 may 2014].

Wikipedia (2014) Code Refactoring [online]. Available from: http://en.wikipedia.org/wiki/Refactoring [Accessed 12 May 2014]