WebElevate Programme

ASSESSMENT BRIEF

NFQ Level : HETAC Level 8

Course Name : WebElevate 2.1

Assessment Details

Course		WebElevate 2.1		
Stream		STREAM A	**Module**	Fundamentals of Programming
Title/s		CodeBreaker java programming assignment	**Lecturer**	Conor O'Reilly
Date Issued		19/05/2013	**Submission Date**	14/07/2013

Assignment Overview

Based on the Java programming language and the javabook class library, write a Codebreaker game in Java based on the details supplied in the detailed brief section of this document. The game will have a purely text based interface, no graphics, as depicted in the detailed brief section.

Grading Overview

With regard to programming assignments. The most marks are given for working code, associated documentation and references. While the code may not fulfill ALL the criteria, constructs or requirements of the assessment, having working code demonstrate your ability to follow programming practice – building from the small to the complex. A detailed marking rubric is available at the end of the document, the following is an overview

Evidence	**Total %**	**% for this Assignment towards overall Grade**
Basic Working code demonstrating a basic set of the programming features and disciplines required by the assignment.	**0 – 20%**	**40%**
Practical Working code demonstrating most of the programming features and disciplines required by the assignment.	**20% - 30%**	
Expert Working code demonstrating a ALL of the programming features and disciplines required by the assignment. Whilst demonstration independent research, critical thinking and a deep understanding of programming processes e.g	**30% - 40%**	

approaches\error handling, plus the design constructs required by the assignment.		

Submission Instructions

The assessment material (unless otherwise stipulated in this brief) MUST be submitted via Moodle it cannot be submitted through e-mail.

The submission MUST be a single file in ZIP format the zip file MUST contain the following:

- A zip of the java program (.java and .class files), it MUST contain a run.bat that allows the code to be executed. So if the code was written on a MAC it must be tested on a PC to ensure that it works.
- An eLabBook containing all your program documentation as per the format we have used up to now.

The submission file should be named as follows:

YourFirstname_YourFamilyname_WE21_CODEBREAKER.zip

Ensure that the eLabBook containing your documentation on the code contains the following information

- Your Name
- DIT Student Number
- Your stream
- Your E-Mail
  (in case there are issues running the code, or clarifications are required)

All code files should contain the following headed

```
/**********************************************************
*  Author: yourname
*  Assignment: WE21, codebreaker assignment, Digital Skills Academy
*  Student ID: xxxxx
*  Date : YYYY/MM/DD
*  Ref:  website link to code referenced or the book, authors name and page number
**********************************************************/
```

Reused Code should be indicated by inline comments as follows where they are only a segment of code within a file, if the full file then provide the same information in the header comments

```
// Code reuse
// Ref: http://www.site.com/page#ref
// Ref: Book Title, Author, Publisher, Edition, page number
```

// Ref: Code provided by lab assistant: name or fellow participant: name

Additional Notes

If you have used any code or information from a book or an internet site then you MUST reference the source material in your eLABbook or within the documentation included with the code.

If code has been used then the reference to the source MUST be included in the header of the code file in which it was used. It is ok to use this material so long as you reference it and modify it (with the exception of libraries) according to your needs. Demonstrating critical thinking and the synthesis of material from multiple sources through research to solving a particular problem or provide a particular solution. Understanding of the reference code or libraries used is essential e.g. it's function, error conditions, dependencies and limitations. You must be able to demonstrate your understanding if asked by the assessor. **In all cases final marks for an assignment may be contingent on a presentation by you of your code or an interview whereby you are asked to explain your submission.**

Note: your submitted material cannot be the complete source code from a particular reference or combination of references you must be able to demonstrate your own unique take and solution to the problem.

Taking into account the above, an act of Plagiarism would be considered in the following circumstances:

- Submitting another student's work as your own, with or without that person's consent.
- Allowing another student to use your entire code.
- The submission of code from a previous running of the course.
- Not being able to demonstrate an understanding of the overall structure, intent and functionality of the code.
- Utilising code with no understanding of its function, purpose and limitations where it has not been tailored to the specific requirements of the assignment.
- Reusing of code from other sources which has not been substantially modified.

So if you reuse any code,

- Clearly identify the section of code and reference in the header or within inline comments what has been reused and what is your own contribution
- Acknowledge and refer to the source and the author, providing URL and Book name, author and page number
- Be aware that taking code and making minor changes (e.g., changing class names, variable names or methods) is still considered to be reuse

In every day programming there is a significant reuse of code. You will not be marked on the code reused but rather on your contribution to the code which demonstrates your understanding in relation to the learning outcomes of the assignment and the course in general.

Detailed Brief

OVERVIEW

Write the code breaker program as per spec below with text based display using the Javabook class library

Following an introduction screen that explains the game. The game starts by choosing the code patch which is a sequence of four colours from the following available colours red(R), orange (O), yellow (Y), green (G), blue (B), Indigo (I), and violet (V). The code patch is not displayed to the user only 4 lines are displayed (_ _ _ _) and lives = 8.

The user enters a string of four characters which is their guess at the sequence of 4 letters choose by the computer.

The user input is compared to the code patch and the following feedback is provided:

- If the two code patches match, the game says **YOU WIN, do you want to play again (Y/N)?**
- If one or more colours between the two colour patches match, display the positions where the colours match. If the colour is correct but the position is wrong for one or more colours give the user a clue as to how many colours there are in the in the users patch that are not in the correct position. If the same colour is used twice in the users patch and twice in the computers patch but the positions are wrong the clue will have a value of 2. if the colour patch has not been guessed then a life is lost and the user is asked : **Enter a sequence a 4 character sequence from ROYGBIV or 0 to exit:**
- If the number of lives is zero following this guess and the user has not won, then display, the code patch and. **YOU LOOSE, do you want to play again (Y/N)?**
- If instead of entering in a code sequence the user enters 0, or there are 0 in the code patch entered, exit the game (boss kill switch)
- If the same sequence is entered twice or more, inform the user that duplicate patches are not allowed and ask then to re enter a new code patch. No life is lose for a duplicate entry.

Display the game as per the example screens on the next page.

Introduction

The fate of the world rests on your shoulders. A lethal virus is about to be on leashed on the web. You can stop the viruses release if you can guess the sequence of 4 colors that delete the virus.

The possible colors are

R - Red

O - Orange

Y - Yellow

G - Green

B - Blue

I - Indigo

V - Vilot

You have only 8 chances to guess the code. Are you ready to save the world (Y/N)?

```
Lives: 8
Code: _ _ _ _  Guessed : _ _ _ _  Clues:
Enter a sequence a 4 character sequence from the following values
ROYGBIV or 0 to exit:
```

```
Lives: 4
Code: _ _ _ _  Guessed : _ _ _ _  Clues:
Code: R _ _ _  Guessed : R O Y G  Clues: 1
Code: R _ _ _  Guessed : R B G I  Clues: 2
Code: R G _ _  Guessed : R G V I  Clues: 2
Code: R G I V  Guessed : R G I V  Clues: 0
YOU WIN !!
Play again Y/N:
```

```
Lives: 0
:
:
Code: R _ _ _  Guessed : R V V V  Clues: 0
YOU LOOSE, the code was : R G I V
Play again Y/N:
```

ADDITIONAL REQUIREMENTS

The program should be submitted on Moodle in a zip file which contains

- A zip of the java program (.java and .class files), it MUST contain a run.bat that allows the code to be executed. So if the code was written on a MAC it must be tested on a PC to ensure that it works.
- An eLabBook containing all your program documentation as per the format we have used up to now

You MUST use the Javabook classes for data entry and displaying of the hangman screens

A text based version of codebreaker is what is required NOT a picture version

SUPPLEMENTAL MATERIAL

Supplemental material regarding this assignment maybe added to Moodle as questions regarding the assignment are addressed in labs, at lectures or online in discussion groups. It is your responsibility to ensure that you regularly review the modules Moodle site for assignment updates, clarifications or additional material.

Marking Rubric

- **Compiling and Running Code** 15%
- **Documentation** 20%
- **Functionality Implemented** 45%
  - Marks per element implemented and how it was implemented
  - Understanding off OO principles
- **Code Design and Presentation** 20%
  - Code layout
  - User interface

Each of the marks above is considered on the following scale

None = component not present, Basic = present but not complete or has error. Practical= present and good to very good implementation based on requirement. Expert demonstrates good design, process and understanding	None	Basic	Practical	Expert
	0	1 - 4	5 -7	8 - 10

Caution

- Deferral of assessment can only be made under exceptional circumstances and by applying directly to the Quality Assurance Manager (qa@webelevate.ie) and completing a Personal Circumstances form (obtained from Moodle under the "course documents" section in Training and Support) with details in the participant handbook section 6.1.3. These must be **received and agreed in advance of the submission date** or work will be deemed as not received.

- Re-check and re-mark procedures are also outlined in the participant handbook, sections 6.1.4 and 6.1.5 (with forms obtained from Moodle under the "course documents" section in Training and Support)
- DSA policy on plagiarism and collusion are detailed in the participant handbook, sections 6.3 and 6.4.
- There is a two week period after results are released, for any general enquiries on assessment results, after which quality assurance queries **will not be considered.**
- Submissions must **strictly adhere to the instructions** on the Assessment Brief, including the **correct format for digital submissions and the naming conventions associated with files.**
- Participants must sign off when handing in work

APPENDIX A – DEVELOPMENT APPROACH

As you should be aware from lectures, your approach to development should be as follows where functionality is built in built up in stages, where each stage or sprint (agile methodology) produces working code with a set number of features. That code is saved so as you approach the assignment deadline you always have a version of working code to submit.

The process described below is provided as a reminder of how you should be approaching the assignment.

STAGE 1: DESIGN

Before writing code your code the design should be documented in the eLabbook or the programs documentation as follows:

1. Problem Definition
   - What is the objective
   - What is the program to do

2. Design
   - Draw a picture of the execution steps
   - Write down in words the execution steps
   - Draw the design Object or Class diagrams

3. Test Cases (how will you test it)

   - Write what you will use for testing that it runs and creates the right answer
     - Test Case 10 : 1 + 1 = 2 Simple Case
     - Test Case 20: 5 + 9 = 14 Normal Case
     - Test Case 30: 0 + 9 = 9 Edge condition
     - Test Case 40 : 5 + 0 = 5 Edge condition

   If this was division we could have division by zero issues and very small answers

     - Test Case 50 : 166666666666 + 788777777777777 = 788944444444443 test the very big

STAGE 2: BUILD 7 TEST IN SPRINT

Once you have documented your approach you should proceed to do the following:

1. Write Code
   - Step by step, on piece of functionality at a time, get it working, save a copy of that working version ***addTwoNumbers_v1.code*** in your _Attic directory, add the next bit of functionality
2. Test Code with your test cases
   - Debug the code, change ONLY ONE thing at a time, KEEP SAVING VERSIONS

STAGE 3: DOCUMENTATION

The code once written should be documented in the eLabBook or in the program documentation under the following headings:

1. Overview
   - State the problem, user goal or objective the code was designed to solve
2. Screens
   - Take snapshots of the program screens and copy them into the eLabBook
3. Documentation
   - How to use the program documentation
   - Object and class diagrams showing the implementation
   - Database or data definitions used by the code
   - Error Handling and limitations of the design
   - File structure of the source and build code
   - Installation instructions
   - Maintenance instructions
     i. Configuration options
     ii. Housekeeping requirements
     iii. Extending the code, best approaches
   - How could the code be improved
4. Code
   - Insert code fragments which need some explanation beyond object and class diagrams
   - Insert flowcharts, maps or sequence diagrams where necessary to explain the flow of execution of the code
5. Test Records
   - Records of the tests you performed and the results
6. References
   - Any websites or code you looked up or used in the creation of your program