Session Overview

Java Codebreaker Assignment

Aim

- To review the Java Codebreaker Assignment requirements.

Objectives

- Review Assignment Brief
- Abstraction of Java Codebreaker Assignment
- Sample Code Review

Review Assignment Brief

Read the assignment Brief **THOROUGHLY!**

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%	

eLabbook Assignment = 30%
Codebreaker = 40%
UCD Web Prototype = 30%

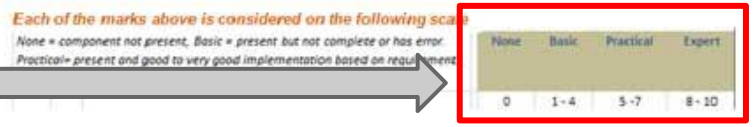
Breakdown of Codebreaker's 40% into Assignment Requirements

Level Of Functionality Achieved

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

How Individual Assignment Requirements are Assessed



Marking Rubric

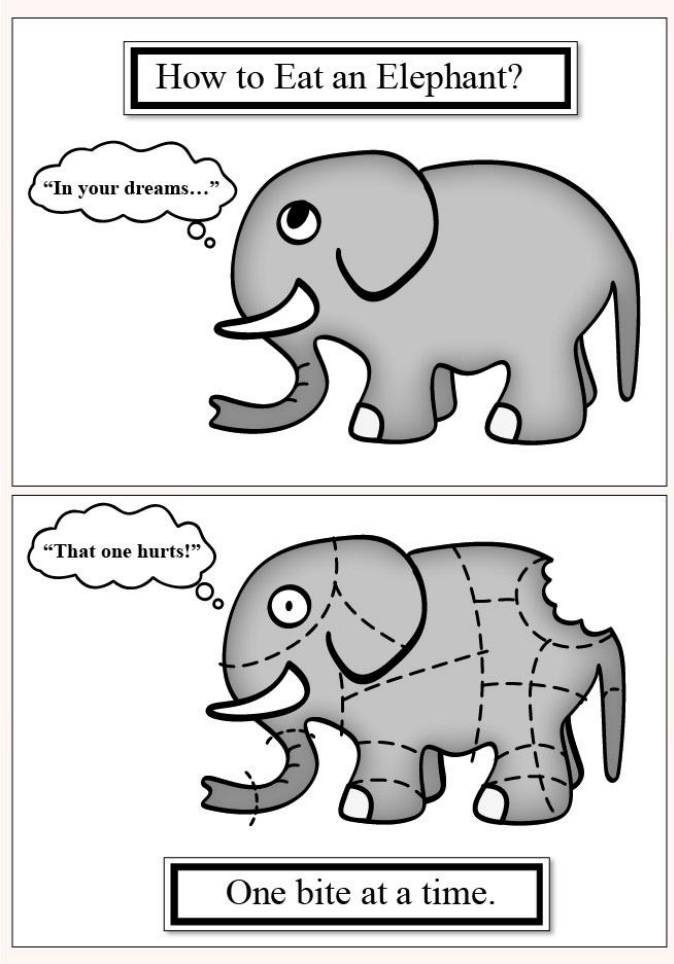
Compiling and Running Code **15%**

If your code does not compile and run the functionality cannot be assessed!

Write small blocks of code that are complete and that will compile,then add more complete blocks

Save each step in _Attic so you have something to go back to if your next change breaks everything.

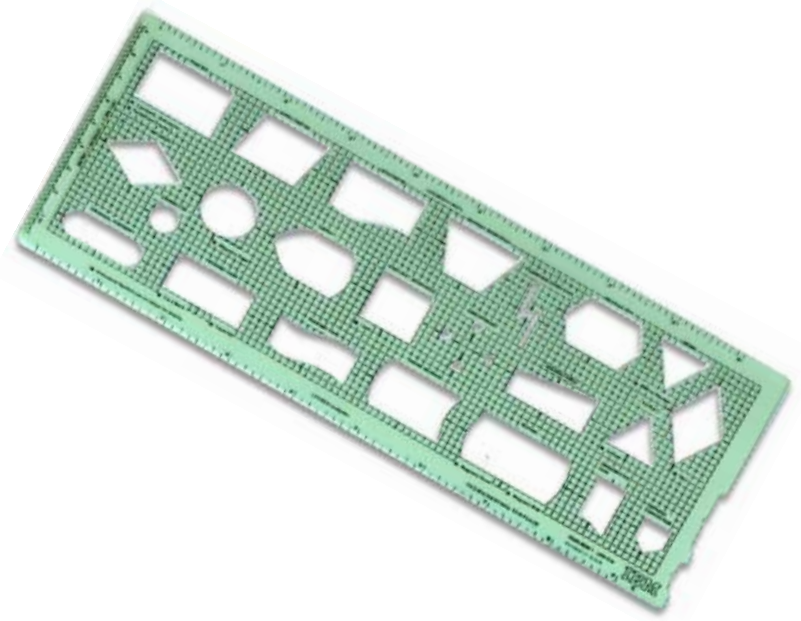
Moodle wk4 lecture7 Instantiable Classes (pg75)



Marking Rubric

Documentation **20%**

eLabbook - Document your approach



Moodle week1 - Lab1 - ExampleOfFilledInDocumentation.pdf

- Problem Description

Objective, Input, Processing, Output

- Design

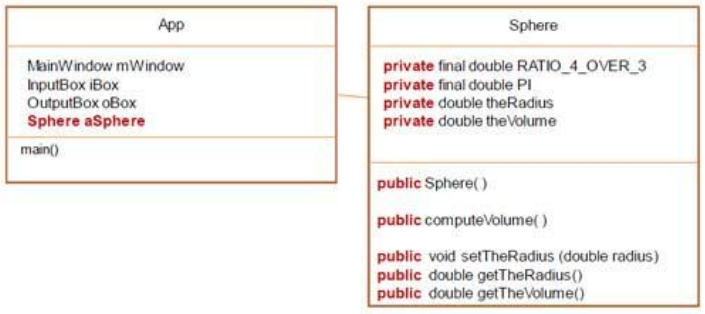
UML, Flowcharts, Class Diagrams, Object Diagrams, Pseudo Code, Structured English

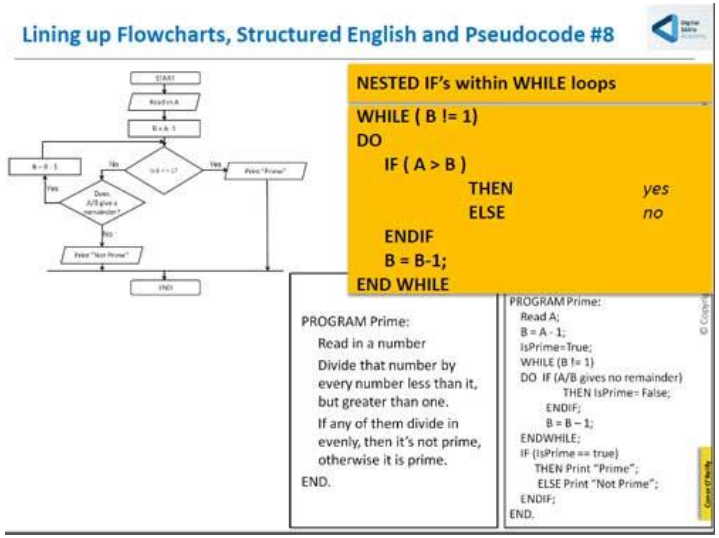
- Implementation

Skeleton, Input & Output, Processing, Make it Pretty

- Testing

Test Cases & Test Results make up the Test Log



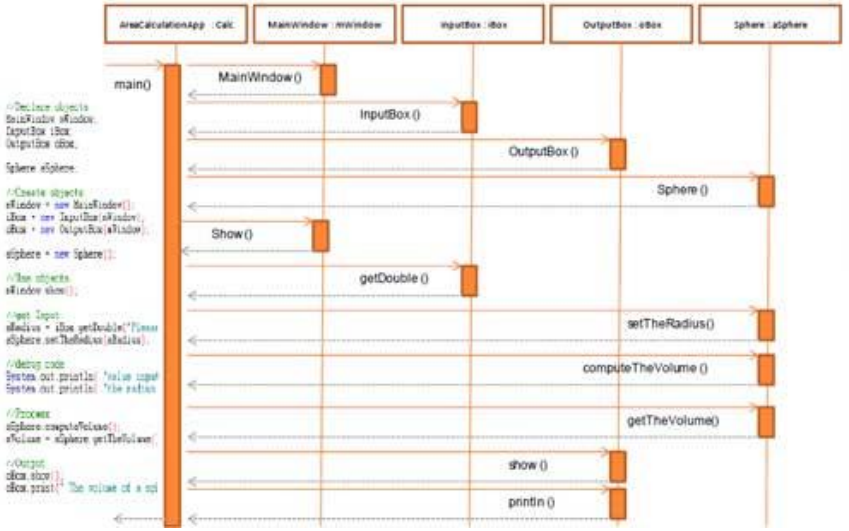


Moodle week2 - Lecture 3 - Representing Code

Marking Rubric - Documentation

Moodle week5 - java programming assignment 2 - brief (pg8)

- Overview
- Screens
- Documentation
- Code
- Test Records
- References



- Created Circle Class and compiled it by it's self to make sure ok before joining it up with the App.class

- http://introcs.cs.princeton.edu/java/11style/

IDENTIFIER	NAMING RULES	EXAMPLE
Variables	A short, but meaningful, name that communicates to the casual observer what the variable represents, rather than how it is used. Begin with a lowercase letter and use	maxx hourlyWage

Testing

Code: AreaCalculator
Version: 1
Author: Conor O'Reilly
Tester: Conor O'Reilly
Test Date: Sun, 10 Jun 2012 11:00pm

		circle	Triangle		Rectangle			
Ref	Description	radius	base	height	length	breath	Expected output	Actual Result
001	Check 0 exits							
002	Check 1 starts Circle calculation							
003	Check 2 starts Rectangle calculation							
004	Check 3 starts Triangle calculation							
005	Area of Circle calculation	15					706.8583471	
006	Area of Circle calculation	67					14102.60967	

Moodle week4 - AreaCalculatorExam.pdf

Marking Rubric

Code Design and Presentation **20%**

BasicAreaOfACircleApp.java

```
1  /**
2   *  Calculate the area of a circle of radius 15
3   *
4   *  @author Conor O Reilly
5   *
6   */
7
8  class BasicAreaOfACircleApp
9  {
10     public static void main(String args[])
11     {
12
13         //Declare variables
14         double pi = 3.14;
15         double radius = 15;
16         double theArea;
17
18
19         // get Input
20         // no need as radius value is already set above
21
22         //Process
23         theArea = pi * ( radius * radius);
24
25         //Output
26         System.out.println();
27         System.out.println(" The area of a circle with a r
28         System.out.println();
29
30     }
```

Code Design:

- Indenting
- Commenting

Presentation:

- User Interface

Marking Rubric

Compiling and Running Code	**15%**
Documentation	**20%**
Code Design & Presentation	**20%**
Total	**55%**
Functionality Implemented	**45%**

The assignment offers percentage points for each marking rubric. Make good use of all of them.

It is as important to show an understanding for the process of developing a Java Application as it is to complete the assignment with full functionality implemented.

Abstraction of Java Codebreaker Assignment

1. *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**.*

Input = Code Patch
Process = generate code patch sequence from R O Y G B I V
Output = _ _ _ _ AND lives = 8

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

Input = User String (N.B. you are told the user entry is a four character String)

3. *The **User String** is **compared to** the **code patch**:*

Process = compare User Entry with Code Patch

and the following feedback is provided

Abstraction of Java Codebreaker Assignment

4. *If the **two code patches match**, the game says **YOU WIN, do you want to play again (Y/N)?***

Process = compare User String with Code Patch (as above)
Output = **YOU WIN, do you want to play again (Y/N)?**
Input = Y/N

5. ***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 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:***

Process = compare User String with Code Patch (as above)
a life is lost
Output = Position where colours match **AND/OR** Clues (*single or many*) **AND** Enter a 4 character sequence from ROYGBIV or 0 to exit:
Input = User String **OR**
0(zero) to exit **OR**
User String containing 0(zero) to exit

Abstraction of Java Codebreaker Assignment

6. ***If*** *the* ***number of lives is zero*** *following this guess* ***and*** *the* ***user has not won****, then display.* ***YOU LOOSE, do you want to play again (Y/N)?***

Process = Lives = 0 **AND** compare User String with Code Patch (as above)
Output = **YOU LOOSE, do you want to play again(Y/N)?**
Input = Y/N

7. ***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)***

Input = 0(zero) in User String
Process = Check User String for 0 (zero) input
Exit The Game (boss kill switch)

8. ***If*** *the* ***same sequence is entered twice or more****, inform the user that* ***duplicate patches are not allowed and ask them to re enter a new code patch****.* *No life is lose for a duplicate entry.*

Input = User String
Process = Compare User String with Previous User Inputs.
No life lost for a duplicate entry
Output = **If** User String **Equals** Previous User String
Duplicate patches are not allowed, please enter a new code patch:

Abstraction of Java Codebreaker Assignment

Input = Code Patch
Input = User String
Input = Y/N
Input = User String **OR** 0(zero) to exit **OR** User String containing 0(zero) to exit
Process = Generate code patch sequence from R O Y G B I V
Process = The two code patches match
Process = One or more colours between the two colour patches match
Process = Colour is correct but the position is wrong for one or more colours
Process = Same colour is used twice in the users patch and twice in the computers patch but the positions are wrong
Process = User has not won **AND** Lives = 0
Process = Lose a Life
Process = Check User String for 0 (zero) input
Process = Exit The Game (boss kill switch)
Process = Compare User String with Previous User Inputs.
Process = No life lost for a duplicate entry
Output = _ _ _ _ AND lives = 8
Output = **YOU WIN, do you want to play again (Y/N)?**
Output = Position where colours match **AND** Clues (*none, one or many*) **AND** Enter a 4 character sequence from ROYGBIV or 0 to exit:
Output = **YOU LOOSE, do you want to play again(Y/N)?**
Output = Duplicate patches are not allowed, please enter a new code patch:

4 Processes have similar functionality!

And what about this?

Sample Code Moodle wk5 Guess a Letter

App.java

Does not use Javabook

main() method

- Instantiates **App()** constructor

Contains Variables for both Guess a Number and Guess a Letter.

Uses Private Variables

Uses an array to store previously guessed letters

Uses a String to store the Users Input

App() Constructor

- Sets numberOfLives
- Creates an Array of length "numberOfLives"
- Creates theLetterGenerator and someInput objects
- Calls **playBoard()** method
- Waits for user entry to exit program

playBoard() method

- Uses a do-while loop and if statement to control "Play Again (YN)?"
- Calls playGame() method

Sample Code Moodle wk5 Guess a Letter

App.java cont'd

playGame() method

Selects a random letter from LetterGenerator Class via *theLetterGenerator.getLetter();*

Creates the User Interface

Clears the *lettersEnteredArray[];*

Loops the *numberOfLives;*

Selects the 1st character of the user input and converts it to upper case

Checks the User Input is in the allowed range (A, E, I, O, U) via *theLetterGenerator.checkLetter (this.guessLetter);*

Checks if the letter (User Input) has been entered before

Compares the letters

Counts down a life if the letter entered is not the same as the random letter

Uses the this.guessed variable to decide if the user has guessed the random number correctly or not

Sample Code [Moodle wk5 Guess a Letter]

LetterGenerator.java

GETTER = **getLetter()** method

PROCESS = **checkLetter(char letterToCheck)** method

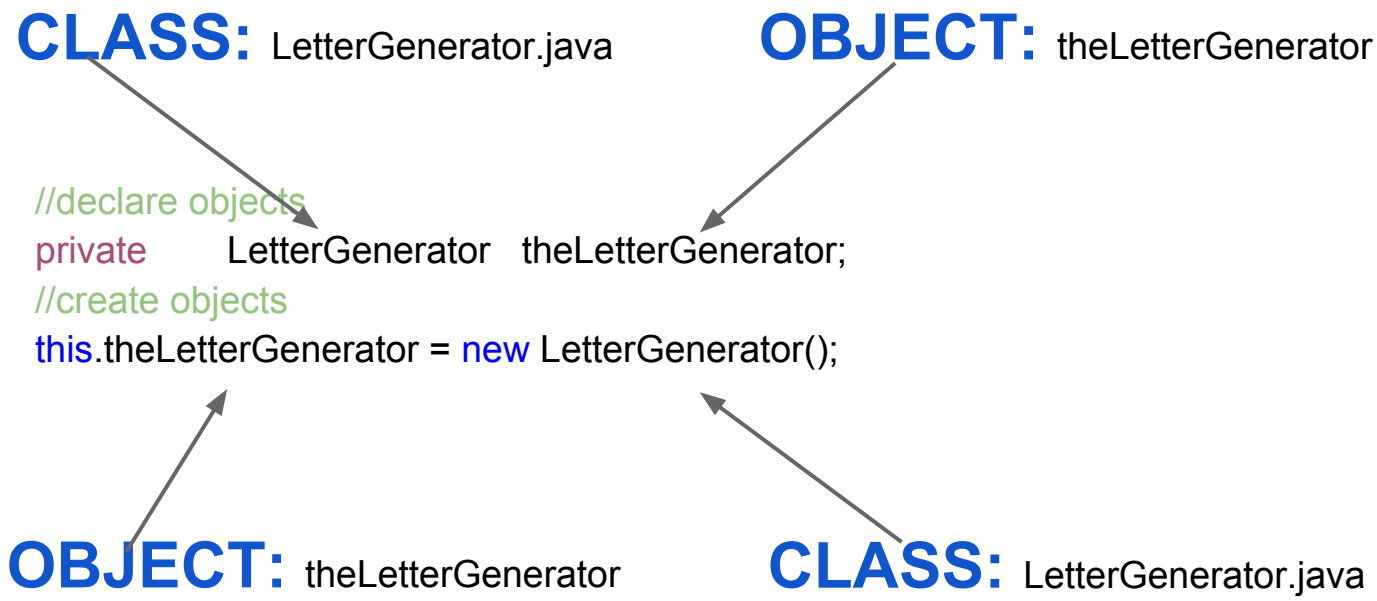
GETTER = **testLetters(int numberToGenerate)** method

NO SETTER? The LetterGenerator class has no need for a SETTER as information does not need to be passed(set) by App.java class.

NOTE: Sample Code being dissected here is *Guess a Letter*. There is also Sample Code for a *Guess a Planet* game in the same section of moodle. Dissecting that Sample Code could be a worthwhile exercise.

And 1 other thing.....

Classes & Objects and Objects & Methods



The **LetterGenerator.Java Class** has:

Constructor	- public LetterGenerator()
Method	- public char getLetter()
Method	- public boolean checkLetter(char letterToCheck)
Method	- public char[] testLetters(int numberToGenerate)

When you create **theLetterGenerator** Object you can access the **Methods** in **LetterGenerator.java** Class

Classes & Objects and Objects & Methods

CLASS
LetterGenerator.java

OBJECT
theLetterGenerator

METHODS
getLetter()
checkLetter(char letterToCheck)
testLetters(int numberToGenerate)

```
class App
{
        public static void main(String args[])
        {
                App anApp = new App();
        }
        // declare object
        private LetterGenerator theLetterGenerator
        // create object
        this.theLetterGenerator = new LetterGenerator

        // get a random number
        this.letterToGuess = this.theLetterGenerator.
getLetter();

        if (this.theLetterGenerator.checkLetter(this.
guessLetter())
}
```

```
class LetterGenerator
{
        public char getLetter()
        {
                // code removed
        }

        public boolean checkLetter( char letterToCheck )
        {
                // code removed
        }


        public char[] testLetters( int numberToGenerate )
        {
                // code removed
        }
}
```