

|  |
| --- |
| **COURSEWORK ASSESSMENT SPECIFICATION** |

|  |  |
| --- | --- |
| **Module Title:** | *Programming 1* |
| **Module Number:** | *CG0047* |
| **Module Tutor Name(s):** | *Alan Maughan* |
| **Academic Year:** | *2014/15* |
| **% Weighting (to overall module):** | *15%* |
| **Coursework Title:** | *Homework 2* |
| **Average Study Time Required by Student:** | *8 hours* |

**Dates and Mechanisms for Assessment Submission and Feedback**

|  |
| --- |
| **Date of Handout to Students:**  Week 4 |
| **Mechanism for Handout to Students:**  *via elp* |
| **Date and Time of Submission by Student:**  During Week 5 Lab Class |
| **Mechanism for Submission of Work by Student:**  Papers collected in lab |
| **Date by which Work, Feedback and Marks will be returned to Students:**  Marks & feedback will be given as the assessment is marked in the lab week 5. |
| **Mechanism for return of assignment work, feedback and marks to students:**  Marks & feedback will be given as the assessment is marked in the lab |

**Further Information**

*(Please ensure the assessment specification includes the following items)*

**Learning Outcomes tested in this assessment (from the Module Descriptor):**

1. Design a program from a specification;
2. Formulate solutions to a number of basic programming problems using an appropriate design notation;
3. Make effective use of basic data types, arrays and structured programming control constructs: sequence, selection and iteration.
4. Understand and make basic use of functions/procedures.

## Assessment Criteria/Mark Scheme:

|  |  |  |
| --- | --- | --- |
| Category | Mark | Description |
| Coding style | 2 | use of header comments  use of method comments  layout and naming convention  visibility |
| Program compiles | 1 | Address class and revised Account class compile. |
| Address class | 1 | Use of ‘this’ (1) |
| Account class  **(Must contain student name in @author tag or 0 marks for class!)** | 3 | address attribute  Constructors initialises address  Correctly uses Address class |
| 3 | removePoints()   * amended * will not allow negative value * produces correct message |

**Nature of the submission required:**

Paper copies of source code. Code execution in lab.

**Instructions to students:**

*This is an individual piece of work.*

**Referencing Style:**

*N/A*

**Expected size of the submission**:

Under 12 pages – mostly printed copies of source code

**Academic Conduct:**

You must adhere to the university regulations on academic conduct. Formal inquiry proceedings will be instigated if there is any suspicion of misconduct or plagiarism in your work. Refer to the University’s regulations on assessment if you are unclear as to the meaning of these terms. The latest copy is available on the university website.

# Homework 2

This work is due at the start of your lab in week 5. It counts for 15% of the overall module mark.

This homework is due to be marked at the start of your lab class in week 5. You must bring with you a printed copy of your source code (the .java files). These should be produced before you come to the lab. Do not come to the lab and attempt to print copies then. These will be collected by the tutor when they mark your homework – make sure that they have your name / id on them. These will be retained for audit and internal moderation. If these files are not submitted (and printed before the lab) then you will score 0 (zero)!

You must work on the program on your own, outside any formal classes and it must be ready to execute at the start of the scheduled laboratory class. All code must be completed using the BlueJ IDE. Any work utilising other IDEs will score zero.

All code (in this and in all subsequent homeworks) must:

* Have the class header and all methods commented to ‘Javadoc’ standards using @author, @version, @param and @return tags as appropriate.
* Be coded to required layout (e.g. indentation) and naming standards.

Notes on the above were supplied in week 1 and there are numerous examples in the code you will have seen. Failure to meet these standards will result in loss of marks.

You may be asked questions about your program to confirm your understanding and that it is your own work. Failure to answer the questions may result in a deduction or total loss of marks.

***The work must be wholly your own. Collusion counts as academic misconduct and will be punished according to the University’s regulations detailed in “Assessment Regulations for Northumbria Awards” (ARNA) a copy of which is available on the University website.***

## The Task

Create a BlueJ project called “HW2”.

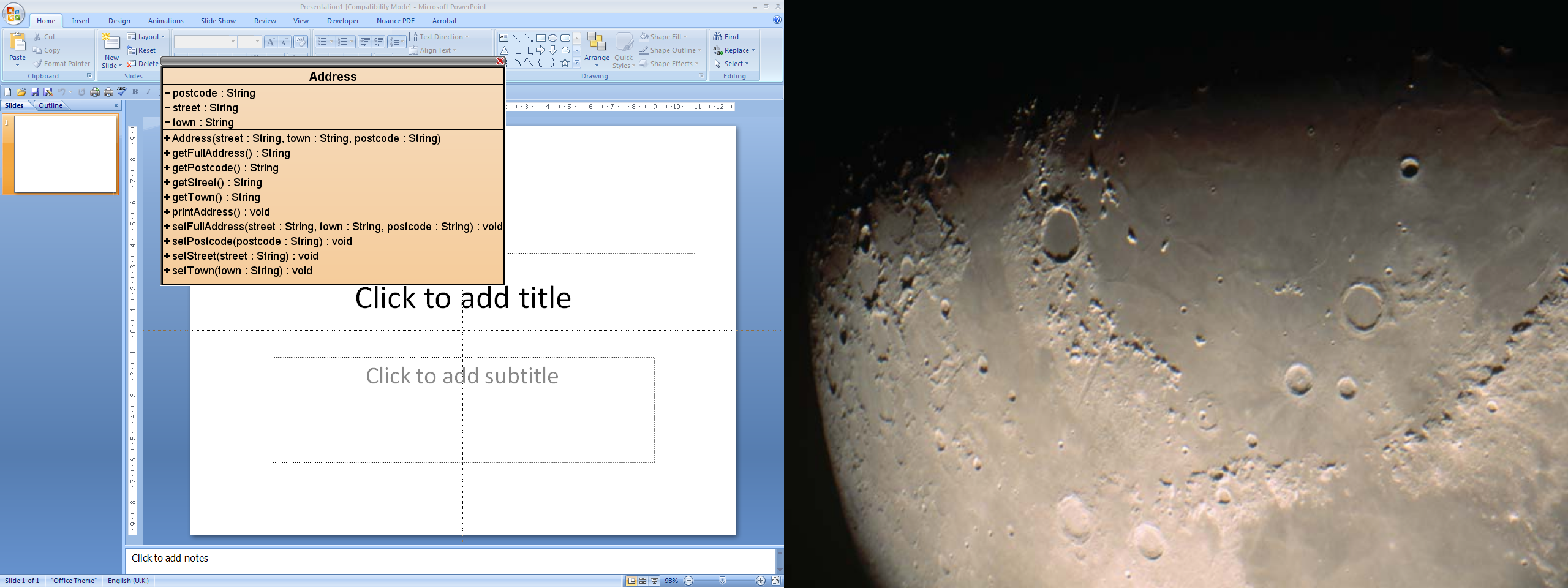
For this homework you are expected to write a program based upon the **Account** class from homework 1. You will create an **Address** class and modify the **Account** class to use it. (The work to do this will be similar to that you undertake in week 4’s lab using **Student** and **Name** classes). Your application will consisting of two classes:

* **Account**
* **Address**

Whilst you may use your version of Account you will find a copy of the class on Blackboard with this homework specification. Add the class to your project. The class will need to be modified to use an Address object. Please ensure that the Account class has your name and version in the header or it will not be accepted and you will get no marks for this part of the assessment.

### Part 1

There is no copy of the Address class. You need to write this class. The structure of the class is:



The constructor will receive parameters of type String to represent an address. The constructor should make use of the ‘this’ keyword (introduced in the week 4 lectoral).

Most methods are simple accessors / mutators. However:

**getFullAddress()** should return the address as a formatted String, one item per line:

e.g. 23 High Street

Newcastle

NE1 1NE

***NOTE:*** *The inclusion of the String “\n” will cause any following output to start on the next line. This can be used to avoid multiple println() statements.*  (e.g. String output = variable1 + “\n” + variable2;)

***NOTE***: *In BlueJ the full address will be shown on one line as it is displayed in a textfield. It does however show the formatting where you have used "\n" to create new lines.*

**printFullAddress()** should print out the full address to the console window as *in* the above example.

**setFullAddress()** should set all the address values.

### Part 2

Once you have written, compiled and checked that Address works as required you should amend the Account class. Modify it as described below.

* The class should also have an attribute address of type Address which should be initialised via values give to the constructors.
* Add the methods **getAddress()** and **printAddress()** to return / print the address utilising the methods of the Address class.
* Add **setAddress()** using the structure shown on the class diagram. This should allow a new address to be input.
* Amend the existing **removePoints()** method so that the number of points cannot be reduced to a value below zero. Where this would happen the pointsHeld attribute should NOT be altered and instead the following message should be output to the console window:

**Transaction refused: Insufficient points available.**

* Amend printAccountDetails() to print the account’s details in the following format:

|  |  |  |
| --- | --- | --- |
| firstName [space] lastName  street  town  postcode  “Account Number: “[space] accountNumber  “Number of points: “ [space] noOfPoints |  | Bert Brownlow  22 The Willows  Newcastle  NE1 1NE  Account Number: 1234  Number of points: 1 |

The structure of your finished Account class should be as follows:

