

|  |
| --- |
| **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 3* |
| **Average Study Time Required by Student:** | *8 hours* |

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

|  |
| --- |
| **Date of Hand out to Students:**  Week 2 |
| **Mechanism for Hand out to Students:**  *via eLP* |
| **Date and Time of Submission by Student:**  During Week 3 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 3. |
| **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:**

See below.

**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 10 pages – 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 1

This work is due at the start of your lab in week 3. It counts for 15% towards the module mark.

This homework is due to be marked at the start of your lab class in week 3. You must bring with you a printed copy of your source code (the .java file). This should be produced before you come to the lab. Do not come to the lab and attempt to print a copy then. The code will be collected by a tutor when they mark your homework (make sure that it has your name / id on it - @author your name / your id). This will be retained for audit and internal moderation. If this file is not available (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.

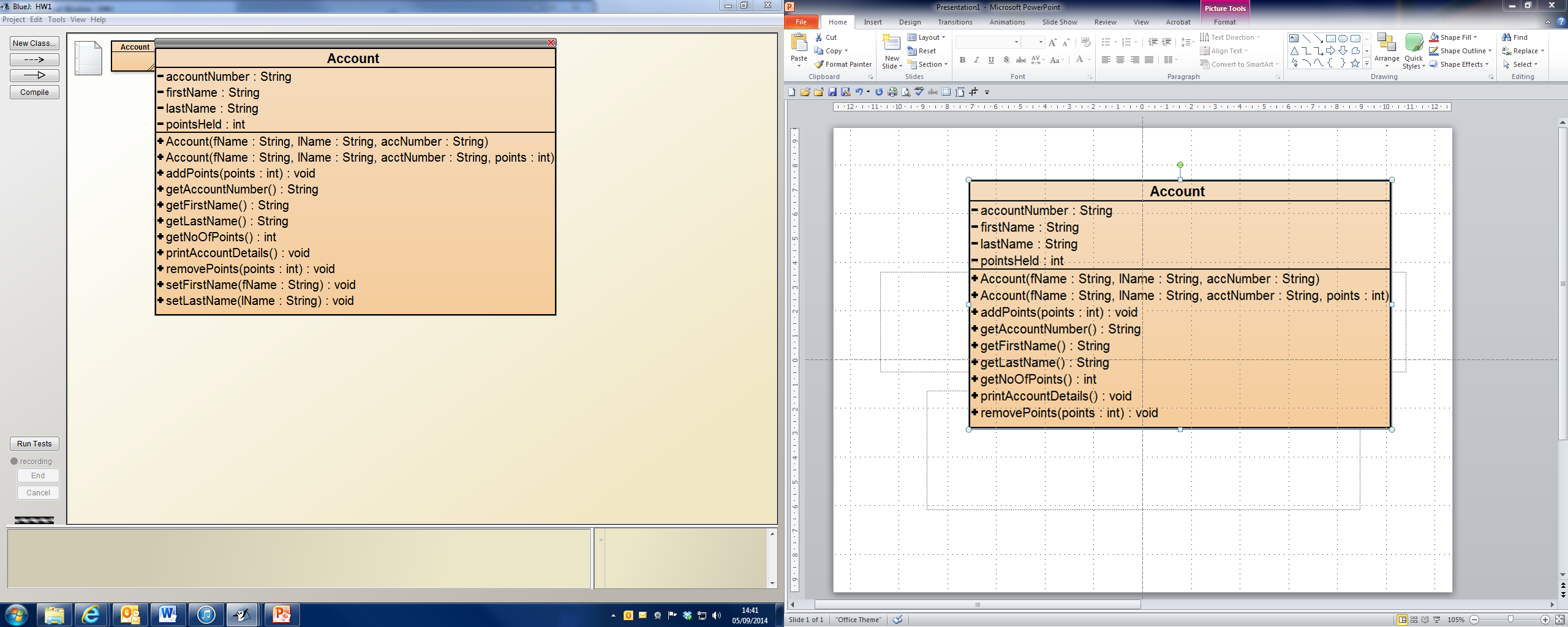
You may be asked questions about your program to confirm your understanding and that it is your own work. Failure to answer the questions may result in a deduction or total loss of marks.

***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.***

A local chain of stores wish to operate a ‘rewards’ scheme where customers will be able to obtain an account card and earn points based upon the value of their purchases. Similarly points held in the account may be redeemable – but the number of points cannot be reduced below 0 (zero). Code written for this homework will be developed in those that follow. However, it should be understood that these homeworks are to test your understanding of Java programming as it develops – you will not be developing a real world product.

## The Task

For this homework you are expected to write simple class Account representing a single loyalty account. The BlueJ project should be called “HW1”. The structure of the class is as per the following class diagram (Note: The attributes / methods do not have to be written in alphabetical order)



There are two constructors. Both will receive parameter data for first name, last name and account number (which you can assume will be a 4-digit string) and will assign these values to the relevant attributes.

* One constructor will set the number of points held to ‘0’. This would be where an account is created without making a purchase at the same time.
* The other constructor should receive a further parameter to specify the number of points to add when creating the account. This is where a purchase is made at the same time the account is created.

NOTE: For this homework there is no need to worry about a ratio between points & purchase value. Your program should just handle the points value given.

The class should be written as per the class diagram – do not use different names for attributes / methods; add additional code / functionality; etc. If you do then you will lose marks. If in doubt ask your tutor.

**Methods**

addPoints / removePoints will increase / reduce the number of points held by the given amount and produce the following message to the console window:

**Points now held: [the number of points]**

(Note: For this homework there is no need to ensure that the number of points cannot be reduced to a figure below 0.)

getFirstName / getLastName / getAccountNumber / getNoOfPoints will return the value held by the relevant attribute.

setFirstName / setLastName are simple mutator methods (they change the value of the attribute).

printAccountDetails should all print details of the account to the console window in a string formatted as per the following example:

**Adam Adamant**

**Account Number: 1234**

**Number of points: 123**

NOTE:

* Don’t forget the space - “Adam Adamant” not “AdamAdamant”

The tutor will expect you to:

* Compile the class;
* Create two Account objects for data given at the start of the lab;
* Be able to demonstrate to a tutor that each method works as expected according to a script given at the lab.

**Marking Scheme: 10 marks / 15% of module total**

|  |  |
| --- | --- |
| Description | Mark |
| use of header comments  use of method comments  layout and naming conventions  visibility | 1  1  1  1 |
| constructors | 2 (1 mark for each) |
| Simple accessors: getFirstName, getLastName getAccountNumber getNoOfPoints | 1 for all |
| Simple mutators: setFirstName, setLastName | 1 for all |
| addPoints / removePoints | 1 |
| printAccountDetails() | 1 |