

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

|  |  |
| --- | --- |
| **Module Title:** | *Programming 1* |
| **Module Number:** | *CG0048* |
| **Module Tutor Name(s):** | *Alan Maughan* |
| **Academic Year:** | *2014/15* |
| **% Weighting (to overall module):** | *7.5%* |
| **Coursework Title:** | *Homework 2* |
| **Average Study Time Required by Student:** | *3 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 6 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 6. |
| **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.

**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 to be marked at the start of your lab in week 6. It counts for 7.5% towards the module mark.

You must bring with you a printed copy of your source code (the files Account.java, PersonalAccount.java and TradeAccount.java). These 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 CG0047 and there are numerous examples in the code you will have seen. Note there are no marks for this but you will lose a mark if these requirements are not fully complied with!

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.

**Scenario**

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 redeemed – 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

This homework is intended to test your understanding of inheritance, polymorphism and abstract classes. NOTE: This homework includes material from weeks 3 - 5. You may start the homework now but you will not be able to complete it until after then.

For this homework you are expected to write a program representing subtypes types of Account. An account may be either a TradeAccount or a PersonalAccount. Both should inherit from the Accountclass. However you should not be able to create instances of type Account and your code should ensure this.

The structure of the program is:

AccountList

Account

Name

Address

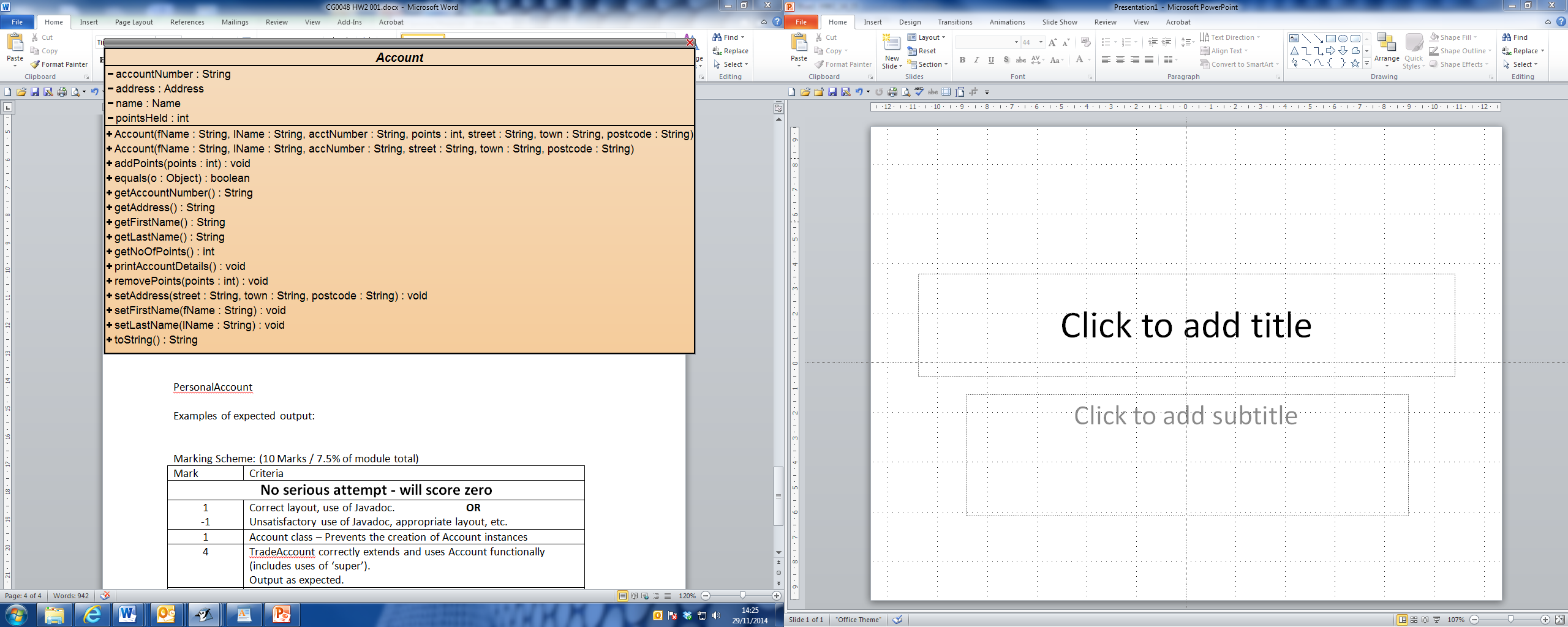
TradeAccount

PersonalAccount

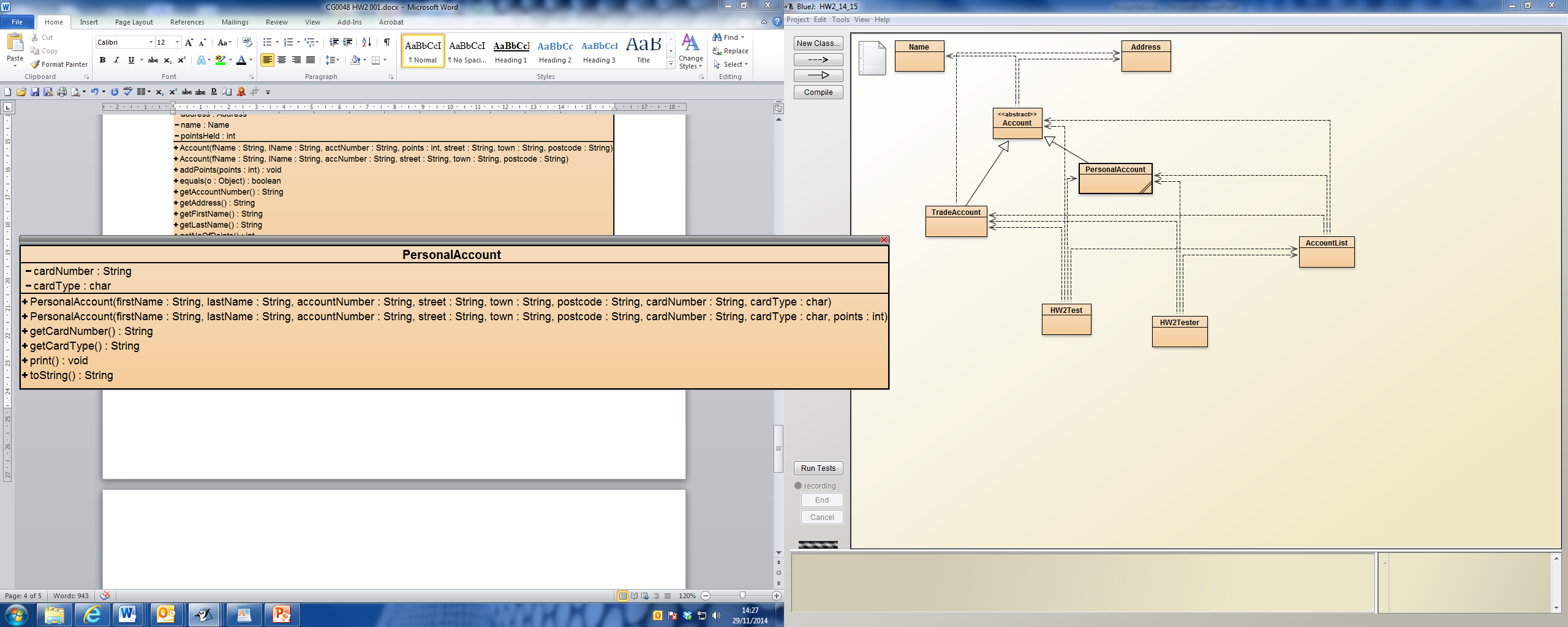
The Name, Address and AccountList classes are provided for your use; they must be used and ***may not be*** amended. You must write the classes Account, TradeAccount and PersonalAccount which should be subtypes of Account. The Account class is slightly different from that used in CG0047. Please ensure that the classes you write conform to the class diagrams. Please note that a TradeAccount has both the Account’s address and a company address.

There is also a class, RunHW2 – see later.

Account



PersonalAccount



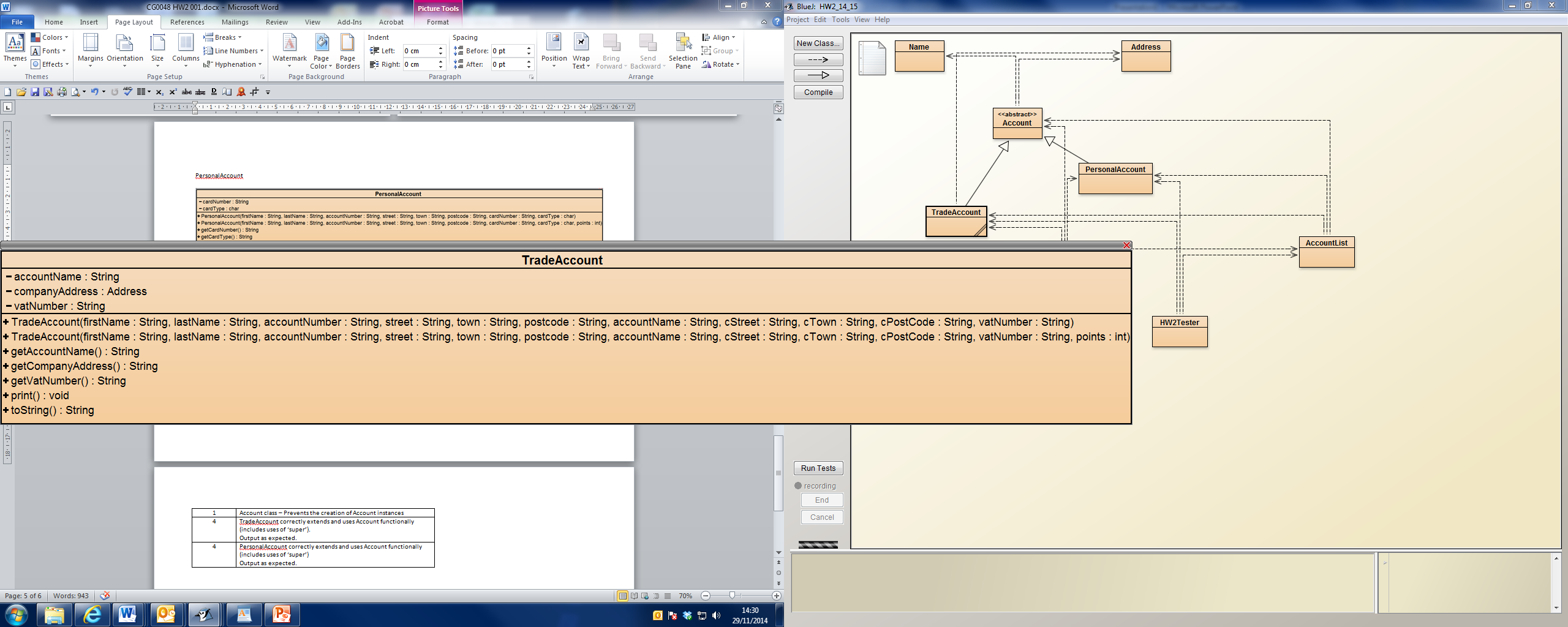
Note:

cardNumber should be an 16 character string e.g. 111222333444. getCardNumber() should return this as the formatted version

e.g. 1111 2222 3333 4444. (Hint. You may wish to look at the substring() methods of the String class.

cardType should be one of the following ‘D’ ‘M’ or ‘V’ and getCardType() should return “Debit” “Mastercard” or “Visa” accordingly.

TradeAccount



Note: vatNumber should be seen as an 8 character string where the first two characters are digits and the remaining six characters are numeric e.g. “GB123456”.

There is no need to need to validate input data for any of your classes. Rather than have you create four objects at the beginning of your lab a class RunHW2 is supplied. This should be used to create the objects and should be ran when the tutor comes to mark your work.

Example of expected output:

Name: A Student ID: 11111111

Personal Account: 1000 Ann Archer

1 Amble Way, Amble, AA1 1AA

Points Held: 0

Card Number: 1111 1111 1111 1111 Card Type: Debit

Personal Account: 1001 Barbara Bach

2 Blyth Boulevard, Blyth, BB2 2BB

Points Held: 20

Card Number: 2222 2222 2222 2222 Card Type: MasterCard

Trade Account: 1002 Colin Cowdry

3 Consett Crescent, Consett, CC3 3CC

Points Held: 0

Account: Cowdry Construction, 33 Chopwell Close, Chopwell, CD3 3CD

VAT Number: GB333333

Trade Account: 1004 Dave Dee

4 Durham Dwellings, Durham, DD4 4DD

Points Held: 40

Account: Dee Design, 44 Darlington Drive, Darlington, DE4 4DE

VAT Number: GB444444

Student should now print out the contents of the terminal window.

|  |  |  |
| --- | --- | --- |
|  |  |  |

Marking Scheme: (12 Marks / 7.5% of module total)

|  |  |
| --- | --- |
| Mark | Criteria |
| **No serious attempt / Does not compile - will score zero** | |
| -1 | Unsatisfactory use of Javadoc, appropriate layout, etc. |
| 2 | Account class produced and behaves as expected. |
| 4 | TradeAccount correctly extends and uses Account functionally (includes uses of ‘super’).  Output as expected. |
| 4 | PersonalAccount correctly extends and uses Account functionally (includes uses of ‘super’)  Output as expected. |
| 1 | Card number correctly formatted. |
| 1 | Card type returned as a formatted value rather than the original single character. |