

1

# BIT694 Designing Applications with C#

## Assignment 1 (TMA1)

### Due date

- ▶ Please refer to the Study plan for the due date.

### Weighting

- ▶ Assignment 1 is worth 30% of the total course mark.
- ▶ You will need to achieve minimum of 40% to pass this assignment.

### Assignment overview

You are required to write a C# program that mimics basic functions of a banking operation, and provide a short written description of the purpose and technical aspects of the program.

Please ensure you have completed all topics and programming activities from Block 1 before attempting this assignment. You should be capable of passing this assignment after you have studied Block 1 and completed the practical programming activities.

All work submitted must be your own.

### Aim

The aim of this assignment is for you to demonstrate the application of object oriented programming using C#, Visual Studio Integrated Development Environment and the Dot Net framework.

**Part 1:** You are provided an opportunity to apply what you have learnt from reading and writing files, manipulating arrays, objects and classes, inheritance and polymorphism to produce a simple application for summative assessment. In addition, this assignment reinforces core programming skills.

**Part 2:** A self-assessment and review section is provided as useful target to aim for in achieving the most from this assignment. The self-assessment is also a checklist and an opportunity to reflect on your progress, recognise your achievements and make plans to improve your learning.

This assignment relates to learning outcomes one and two:

1. Demonstrate adequate and effective use of integrated development environment features
2. Describe and apply the principles and techniques required for modular software development using object oriented concepts and core programming techniques.

## Tasks

- ▶ You must write a C# program that mimics basic functions of a banking operation
- ▶ Complete the self-assessment and review.
- ▶ Submit a zip file of the entire C# assignment project including screenshots to demonstrate the functionality of the program. Submit the zip file using the appropriate submission link.

## Part 1

- (a) Write a C# program to meet the following requirements:
  - ▶ read a customer data file and store it using objects
  - ▶ display a menu to support the basic operations which include withdrawal, deposit and displaying customer details and balance
  - ▶ handle two types of customers: regular and VIP customers
  - ▶ demonstrate inheritance and polymorphism.
- (b) Write a brief description about the program. The brief shall:
  - ▶ describe the purpose of this program (150 words)
  - ▶ describe and use diagrams to explain how you demonstrated inheritance and polymorphism in your programming techniques (150 words).

## Requirements and implementation guidelines

The program receives the data file (input\_assignment\_1 which is a text file) as an input. The file contains 20 lines, each of which represents customer data (comma delimited) and is formatted as follows. The values in each line of the customer data (a row) is referred to as the 'basic information':

1. First name
2. Last name
3. Date of birth
4. Bank account id
5. Bank balance
6. Optional: 'VIP' (This string may not be present for some lines of data in the file, meaning the corresponding customer is not a VIPCustomer)

**Example**

Bob,Jones,20-06-1995,1111,540.56,VIP or Carol,Smith,20-06-1990,1112,1260.06

In addition to the Main class, the program must have two classes: Customer and VIPCustomer where the VIPCustomer is a subclass of Customer. The program is required to demonstrate inheritance and polymorphism.

### **The rules for deposit and withdraw operations are as follows:**

1. For non VIP customers a \$3 fee is deducted from the balance per operation. No fees are deducted from VIP customers
2. For non VIP customers, the withdraw operation will be rejected if it results in a negative balance. This rejection does not apply to VIP customers.

### **The Main class**

As you know, the main class includes the Main method. The following steps should be implemented by the Main method (or supporting methods).

1. Define an array of 20 pointers to customers. At this stage, this is an array of pointers and no object has been created. (Hint: at this stage it is undecided which objects are customers or VIP customers.)
2. Read the input file: input\_assignment1 (the last section shows an example of content) which includes 20 lines, each of which contains the basic information of a customer. When reading a line, the program creates a corresponding object and updates the array of pointers (the created object can be a customer or a VIP customer).

3. After reading the input file, the program displays a menu that reflects the activities below. (See the screen shots of the model answer.)

Menu	
0	<b>Quit</b>
1	<b>Deposit</b> <ul style="list-style-type: none"> <li>▶ Enter the amount and the account ID</li> <li>▶ Deposit the amount and display an appropriate message</li> <li>▶ Return to the main menu</li> </ul>
2	<b>Withdraw</b> <ul style="list-style-type: none"> <li>▶ Enter the amount and the account ID</li> <li>▶ Withdraw the amount and display an appropriate message</li> <li>▶ Return to the main menu</li> </ul>
3	<b>Max balance</b> <ul style="list-style-type: none"> <li>▶ Display the basic information for the customer with maximum balance. In addition, display a 'Happy Birthday' message if today is the customer's birthday.</li> <li>▶ Return to the main menu</li> </ul>
4	<b>Most active</b> <ul style="list-style-type: none"> <li>▶ Display the basic information for the customer with maximum number of operations (deposit and withdraw)</li> <li>▶ Return to the main menu</li> </ul>
5	<b>Youngest customer</b> <ul style="list-style-type: none"> <li>▶ Display the basic information for the youngest customer</li> <li>▶ Return to the main menu</li> </ul>
6	<b>Leap year</b> <ul style="list-style-type: none"> <li>▶ For all customers born in a leap year, display: <ul style="list-style-type: none"> <li>a. The basic information</li> <li>b. The zodiac sign</li> </ul> </li> <li>▶ Return to the main menu</li> </ul>

**Notes:**

- ▶ Add supporting methods, variables and classes whenever you think it is appropriate.
- ▶ Some details were omitted on purpose. A part of learning programming is to make choices.

## Screenshots for testing and highlighting the activities

These screenshots were taken using the content for the input file, as they appear in the last section.

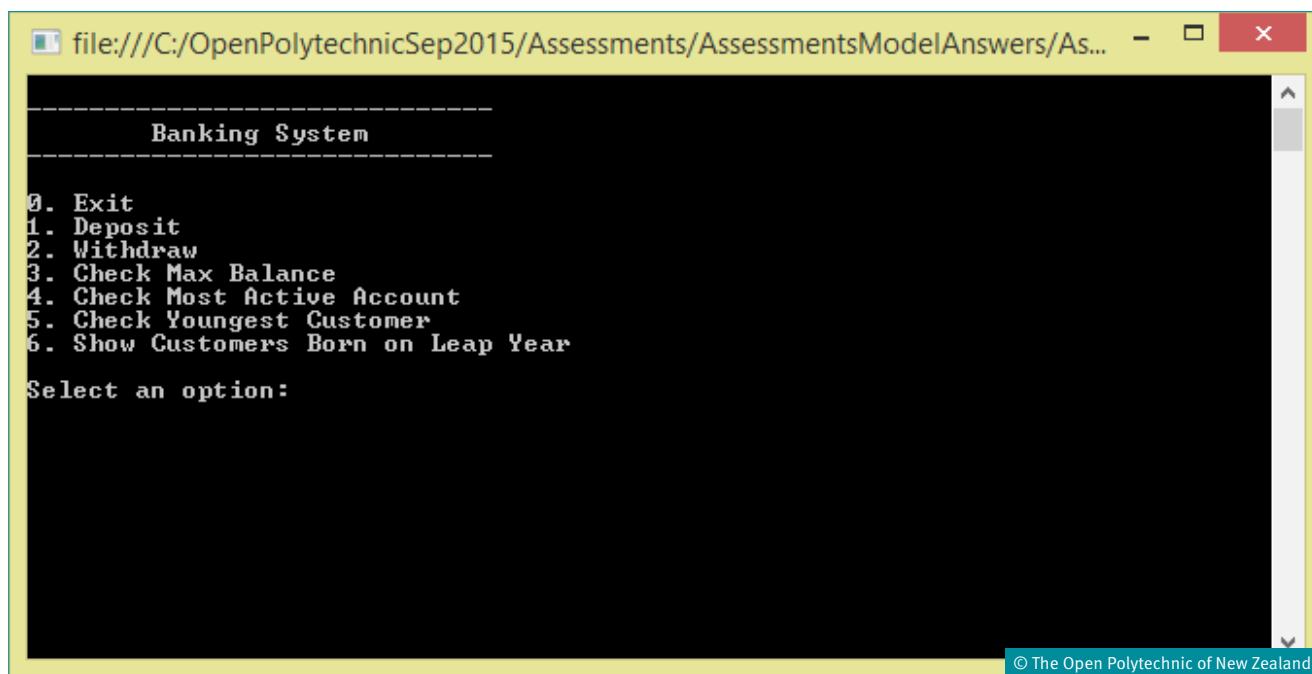


Fig 1. Screenshot 1

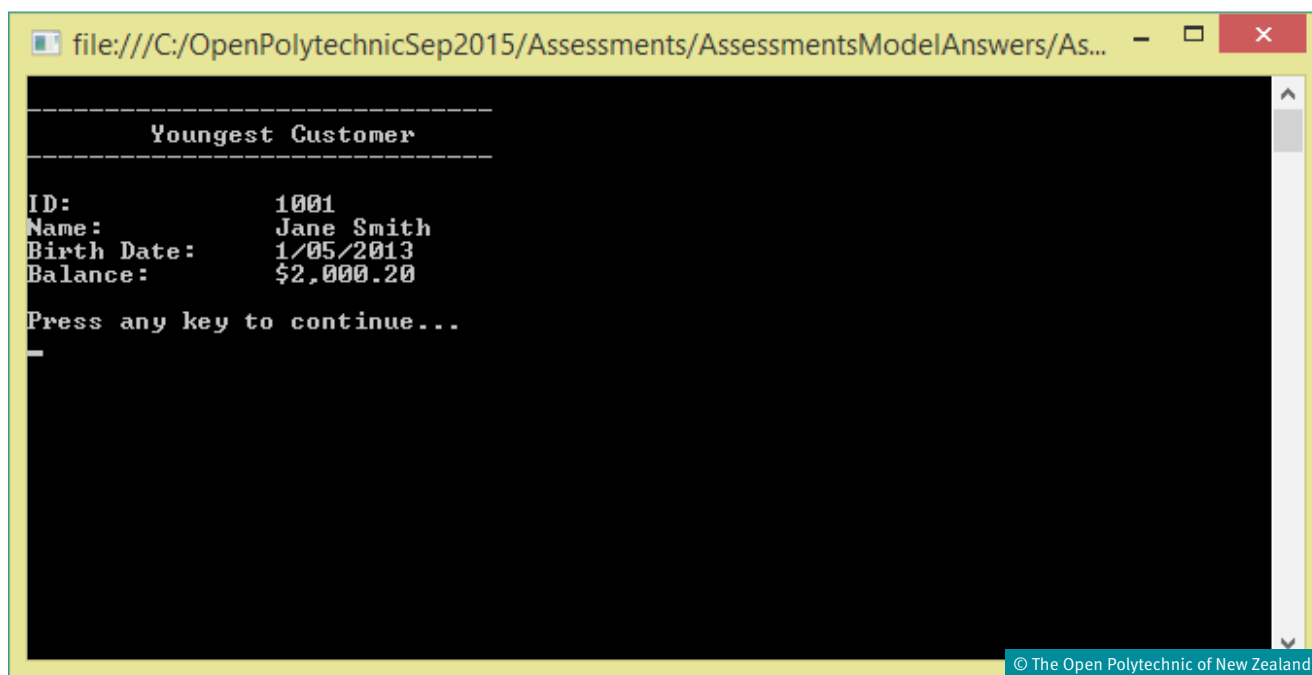
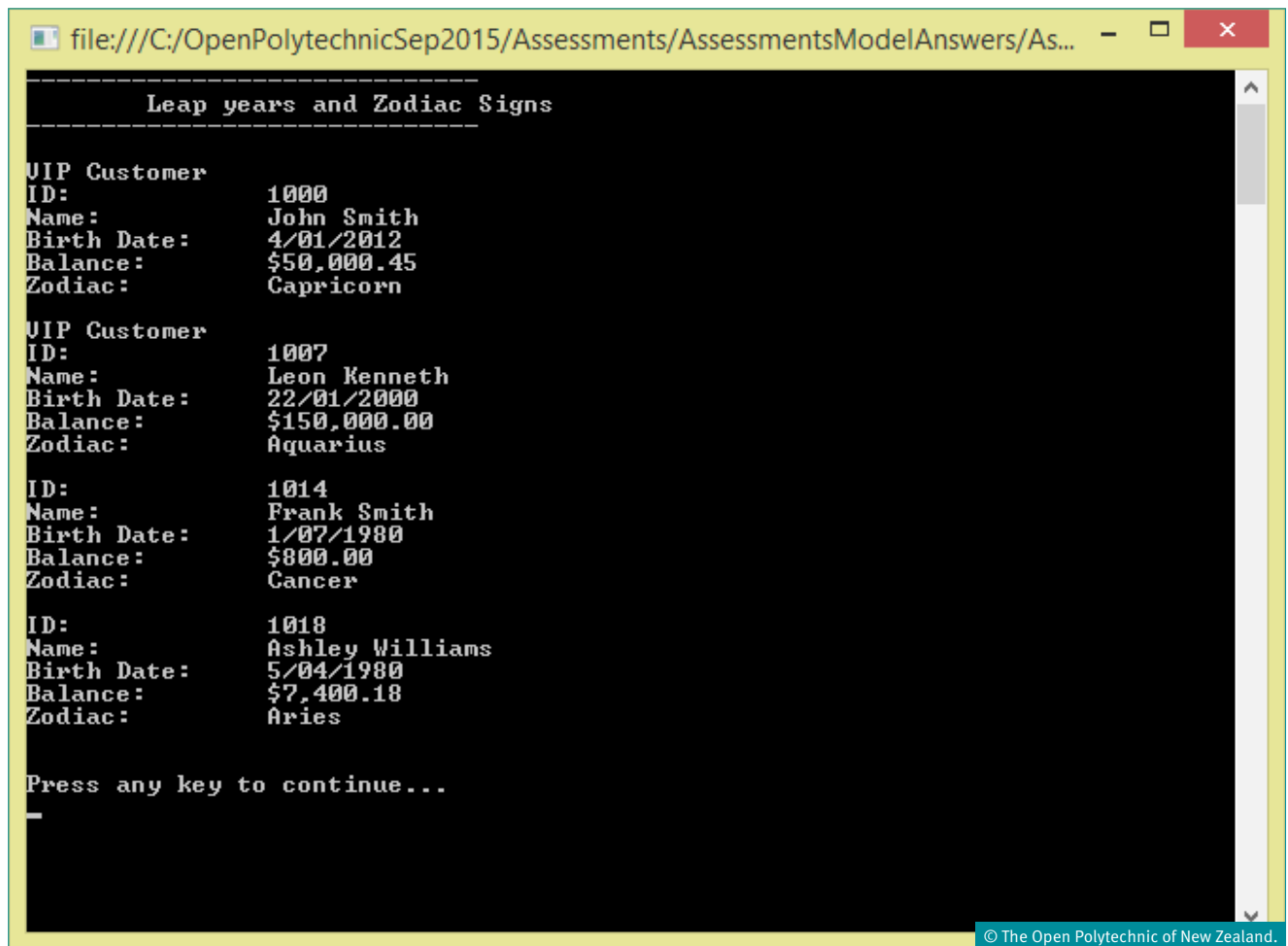


Fig 2. Screenshot 2

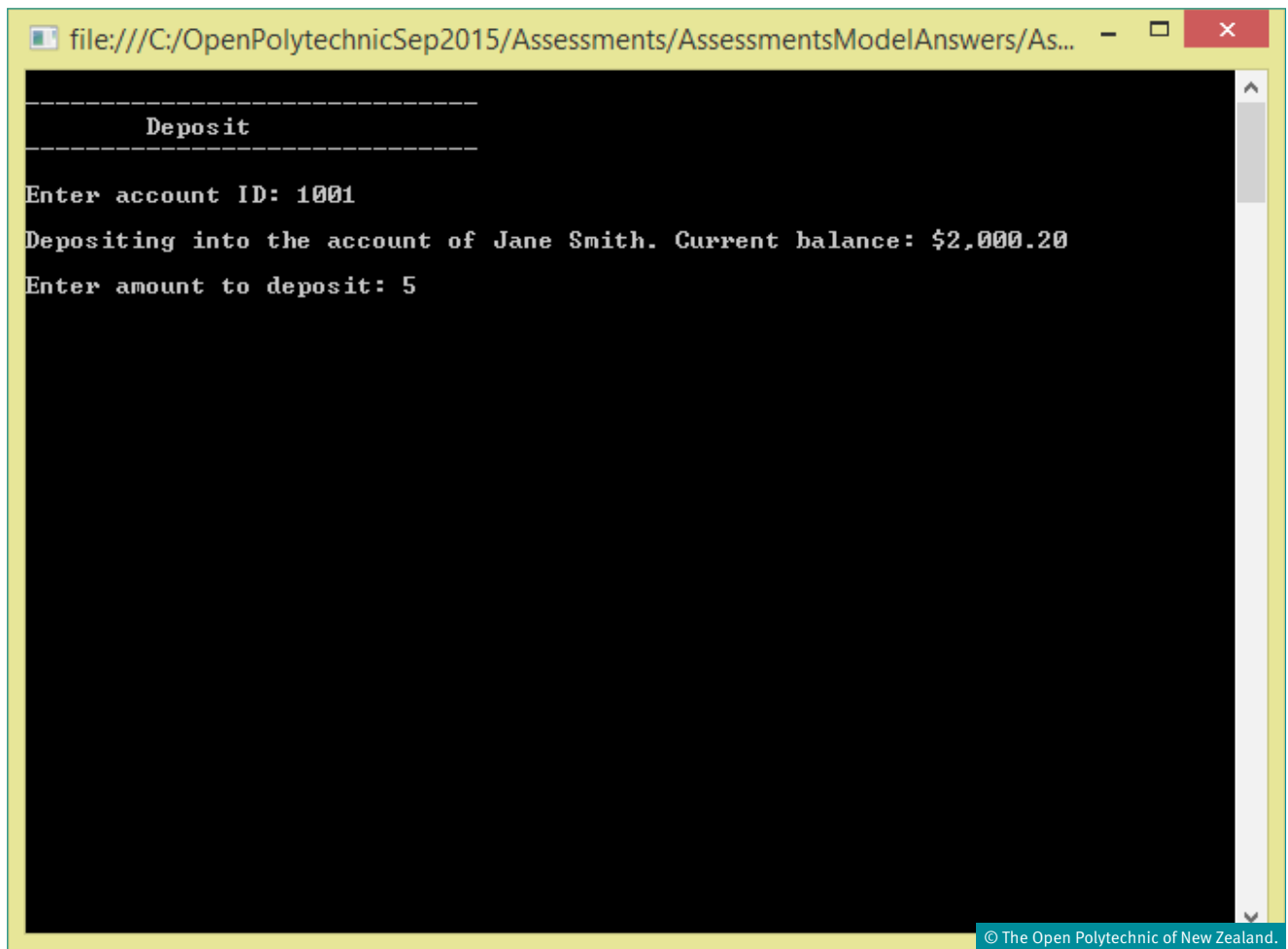


file:///C:/OpenPolytechnicSep2015/Assessments/AssessmentsModelAnswers/As...

```
-----  
Leap years and Zodiac Signs  
-----  
  
UIP Customer  
ID: 1000  
Name: John Smith  
Birth Date: 4/01/2012  
Balance: $50,000.45  
Zodiac: Capricorn  
  
UIP Customer  
ID: 1007  
Name: Leon Kenneth  
Birth Date: 22/01/2000  
Balance: $150,000.00  
Zodiac: Aquarius  
  
ID: 1014  
Name: Frank Smith  
Birth Date: 1/07/1980  
Balance: $800.00  
Zodiac: Cancer  
  
ID: 1018  
Name: Ashley Williams  
Birth Date: 5/04/1980  
Balance: $7,400.18  
Zodiac: Aries  
  
Press any key to continue...  
-
```

© The Open Polytechnic of New Zealand.

Fig 3. Screenshot 3

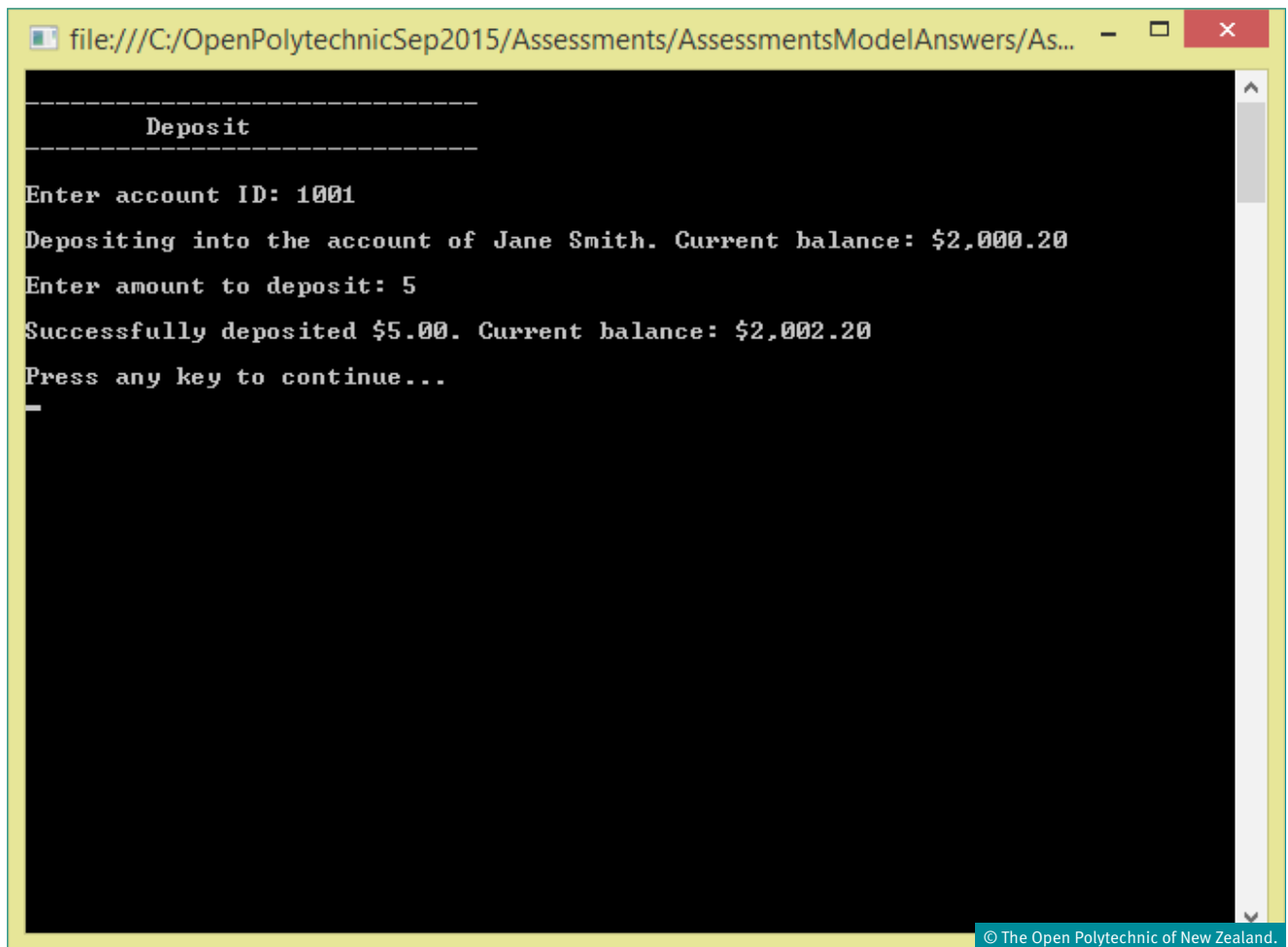


A screenshot of a terminal window with a yellow title bar. The title bar contains the file path: `file:///C:/OpenPolytechnicSep2015/Assessments/AssessmentsModelAnswers/As...`. The terminal has a black background with white text. The text displayed is as follows:

```
-----  
Deposit  
-----  
Enter account ID: 1001  
Depositing into the account of Jane Smith. Current balance: $2,000.20  
Enter amount to deposit: 5
```

At the bottom right of the terminal window, there is a small teal box containing the text: `© The Open Polytechnic of New Zealand.`

Fig 4. Screenshot 4



```
-----  
Deposit  
-----  
Enter account ID: 1001  
Depositing into the account of Jane Smith. Current balance: $2,000.20  
Enter amount to deposit: 5  
Successfully deposited $5.00. Current balance: $2,002.20  
Press any key to continue...  
_
```

© The Open Polytechnic of New Zealand.

Fig 5. Screenshot 5



## **An example of content for the file input\_assignment\_1**

- ▶ John,Smith,04-01-2012,1000,50000.45,VIP
- ▶ Jane,Smith,01-05-2013,1001,2000.20
- ▶ Bob,Washington,09-10-1991,1002,150000,VIP
- ▶ John,Doe,01-10-1989,1003,7400
- ▶ Frank,Smith,31-01-1985,1004,800
- ▶ Chuck,Green,30-05-1974,1005,50000,VIP
- ▶ Katie,Green,28-01-1989,1006,2000
- ▶ Leon,Kenneth,22-01-2000,1007,150000,VIP
- ▶ Claire,Bluefield,01-01-1990,1008,7400
- ▶ Chris,Bluefield,01-01-1990,1009,800
- ▶ Sheva,Alona,05-09-1990,1010,50000,VIP
- ▶ Albert,West,12-07-2003,1011,2000
- ▶ Anna,Willis,18-05-2002,1012,150000,VIP
- ▶ William,Birren,06-01-1990,1013,7400
- ▶ Frank,Smith,01-07-1980,1014,800
- ▶ Stacey,Smith,01-08-1983,1015,50000,VIP
- ▶ Jeremy,Smith,09-11-1986,1016,2000
- ▶ Bruce,Wilson,08-12-1900,1017,150000,VIP
- ▶ Ashley,Williams,05-04-1980,1018,7400.18
- ▶ Kaiden,Jameson,10-07-1981,1019,800

## Part 2: Self-assessment and review

### Instructions

- ▶ Self-assessment and review gives you the opportunity to reflect on your progress, recognise your achievements and make plans to improve your learning.
- ▶ You are required to review your completed assignment, and score it using the same assessment criteria as the lecturer

### Self-assessment review

#### Take note



Assess your assignment using the marking schedule criteria below. Give yourself a mark from the score available for each criterion and then add these up to get your total score.

Criteria	Maximum mark	Self-assessed mark
Writing the class Customer with basic variables and methods.	20%	
Demonstration of inheritance and polymorphism.	10%	
Good use of programming techniques and modular programming <ul style="list-style-type: none"> <li>▶ The code in the Main function is kept to a minimum</li> <li>▶ Functions are short, not too long</li> <li>▶ Helper functions are defined when appropriate.</li> </ul>	30%	
Error handling <ul style="list-style-type: none"> <li>▶ Input file does not exist</li> <li>▶ Lines have the wrong format</li> <li>▶ The program is unable to convert string to an integer</li> <li>▶ The program receives unexpected value from the user</li> <li>▶ Additional error handling where appropriate.</li> </ul>	10%	

Criteria	Maximum mark	Self-assessed mark
Comments and clarity <ul style="list-style-type: none"> <li>▶ All variables declared as class instance fields need comment statement for their usage; all lines of code need to be indented properly according to their logical scope</li> <li>▶ Clear use of private, protected and public properties</li> <li>▶ Clear description of your program: purpose, inheritance and polymorphism.</li> </ul>	30%	
<b>Total</b>	<b>100%</b>	