Object Oriented

Programming

MOD004881

Submission date: 09/08/2019

Assignment 2 (element:011)

From: 1625601  
To: Ioannis Kypraios

**Table of Contents**

Table of Contents i

1. Introduction 4
2. Discussion 4-8

2.1 What is Inheritance 4-6

2.2 What is Polymorphism 4-6

2.3 PPL vs OOP 4-6

2.4 System UML Design 4-6

2.5 Top-down Stepwise Refinement 4-6

2.5.1 Manager’s Top-Down Refinement 4

2.5.2 Client’s Top-Down Refinement 4

2.1 Client Login & Options Menu 4-6

2.1 Manager Login & Options Menu 4-6

2.1 Client Create a Booking 4-6

1. References

**What is Inheritance?**

Inheritance in JAVA allows you to create a super-class also known as a parent class and be able to extend that class to a sub class also known as a child class. Brekelmans, B. et al 2014. Inheritance is used at the time of compiling the code which gives the benefit of re-usability where you can re-use the code multiple times through the java package. Jinghua Zhang, et al 2013 Inheritance also allows you to write smaller pieces of code and combine them into bigger pieces of code, this modular capability allows the code to be more organised and efficient to only load the code that is necessary rather than loading code that was not needed.

The airline management system design takes advantage of the inheritance capabilities for its XML reader and its user login system, the XML reader is set as a superclass which extends to the sub-classes ClientReader and BookingReader to give separate viewing options for both the client and the manager. Another example in the airline management system design is the User set as the superclass which extends to the sub-classes Client and Manager to include separate options for both the client and the manager as well.

**What is Polymorphism?**

Polymorphism in JAVA allows an object to have the ability to take multiple forms using 2 methods, you can either overload or you can override and object in JAVA using polymorphism. Fain, Y., 2011 Overloading is when you create methods with the same name but they have different parameters, Overriding is when you have the same name, same parameters but the body is different by re-defining a superclass inside a subclass which in turn will change the behaviour of that method. Rountev, A., et al 2004. Polymorphism is used at code time rather than compile time.

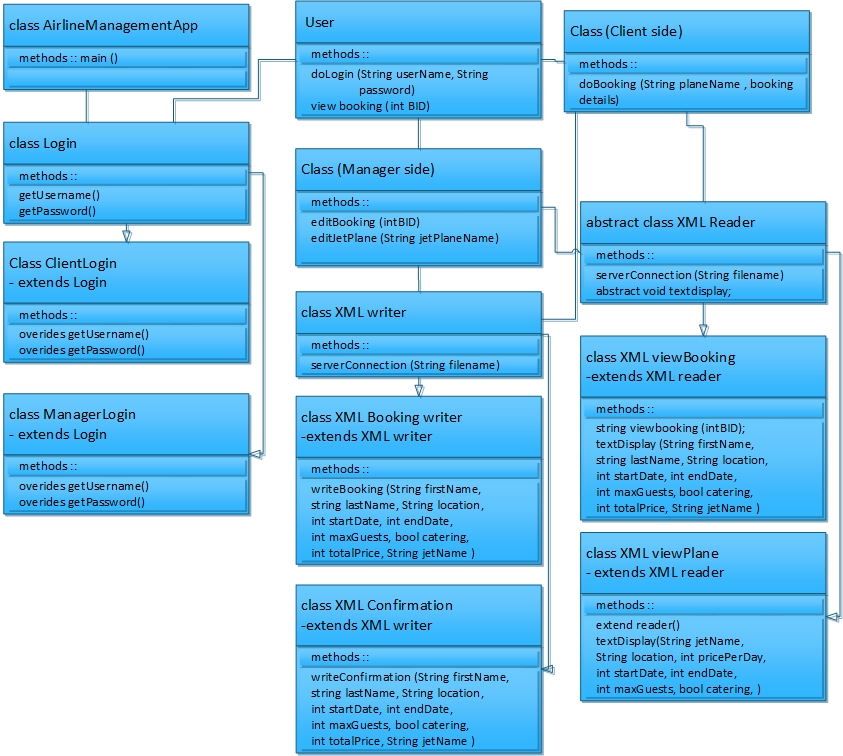
Using polymorphism comes with its modularity benefits and re-usability as you are able to write smaller pieces of code in a more organised broken down way and be able to use the code as and when needed brought forward by a main class later on.

**PPL vs OOP**

Procedural Programming Languages like Basic, Pascal & C run code with a step by step execution with a Top-Down method, this means that the code is like a giant list and runs each bit of code one after the other and is better for hard ware implemented code. This is a faster language to run but requires a lot more thought to be able to implement as you need to think about where you allocate memory for each bit of code to run and has to be set manually. Procedural Programming Languages also use pointers which are coded by the programmer unlike other languages where this is done for you in the background. With so much of the code being manual for Procedural Programming Languages it takes a lot longer to develop software in languages like C, Pascal and Basic and it also becomes a lot harder to debug and maintain the code over time due to its complex nature making it very difficult to design.

Object Oriented Programming Languages like C#, C++ & Java run code with four main concepts being Classes, Objects, and Inheritance & Polymorphism. Classes and Objects make it much easier to organise and be able to break down smaller pieces of code that you can combine together in bigger classes later on giving the programmer a modularity advantage when coding making the development thought process less difficult whilst being able to achieve the result necessary. Working with an Object Orientated Programming system also makes it much easier to maintain in the long run and be able to modify or add new classes and objects onto the system for progressive development.

Object Oriented Programming Languages like Java, C++ & C# make it much easier to work with real life scenario’s where software can work in your favour for everyday situations or even games and simulations where you can test scenario’s and different circumstances and gain statistical data in a simulated environment without the risk factors and dangers of real life. Procedural Programming Languages like Pascal, Basic & C make this very difficult to build such an environment due to its complex architecture and programming methods implemented. OOP is better for software working as a software and PPL is better for working with hardware and physical components.

**System UML Design**

**Top-down Stepwise Refinement**

**Manager**

|  |  |
| --- | --- |
| **First Refinement** | **Second Refinement** |
| * Initialise instance fields  -username  -password * Enter username / password * Enter an option   -Manager view all bookings  -Manager edit a booking  -Exit airline application | * Initialise instance fields  -String username = “ ”;  -String password = “ “;  -Int optionNumber = 1 * Read in username * Read in password * If username / password correct  -Read in optionNumber * Manager view all booking  -display booking text OR * Manager edits a booking   OR   * Manager exits application   -return to main login menu |

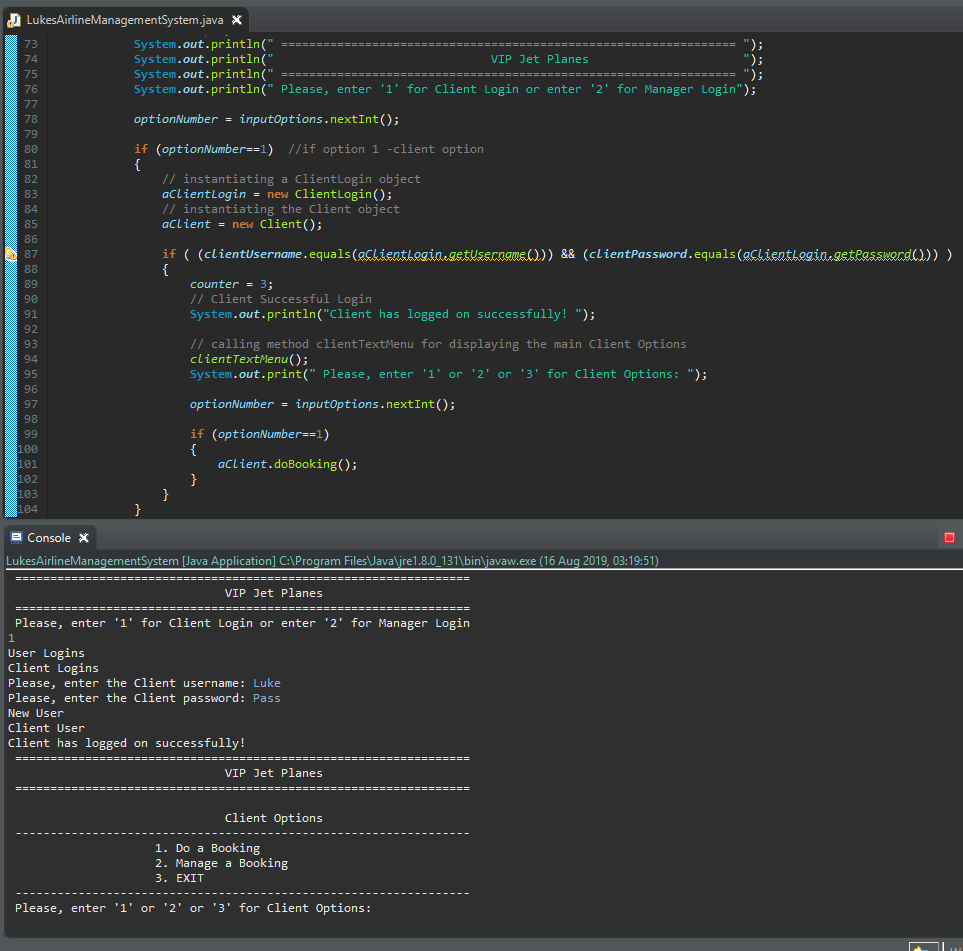
|  |  |
| --- | --- |
| **Third Refinement** | **Final Refinement** |
| * Initialise instance fields  -String username = “ ”;  -String password = “ “;  -Int optionNumber = 1 * Read in the username value using keyboard * Read in password value using keyboard * If username & password match the saved username value and the saved password value from the hardcoded details  -Read in the optionNumber value   using keyboard   * If optionNumber is 1  -manager is allowed to view all of   the bookings, displays text for  each booking   * If optionNumber is 2  -manager is allowed to edit a   booking  -Read in BookingID from XML  -If the BookingID exists  -then display booking  information text for editing   * If optionNumber is 3  -manager returns to the manager login   menu | * Initialise instance fields  -String username = “ “;  -String password = “ “;   -Int optionNumber = 1   * Read in the username value using keyboard * Read in password value using keyboard * While (username AND password match the saved username value and AND password value that are hardcoded)   -Then read in the optionNumber  value using keyboard   * If optionNumber is 1   -then manager is allowed to view all  the bookings and displays the text for  all the bookings   * Elseif optionNumber is 2   -then manager is allowed to edit  a booking  -Read in BookingID from XML file  While the Booking ID exists (while loop condition is true)  -then display the text with the booking  information for editing  Else (while loop condition is false)  -display text “Booking ID does not exist”  -return to start asking for login details   * Elseif optionNumber is 3   -then the manager returns to the main  manager login with the rest of the  manager options.   * Else  -display the message “incorrect option   has been selected !” |

**Top-down Stepwise Refinement**

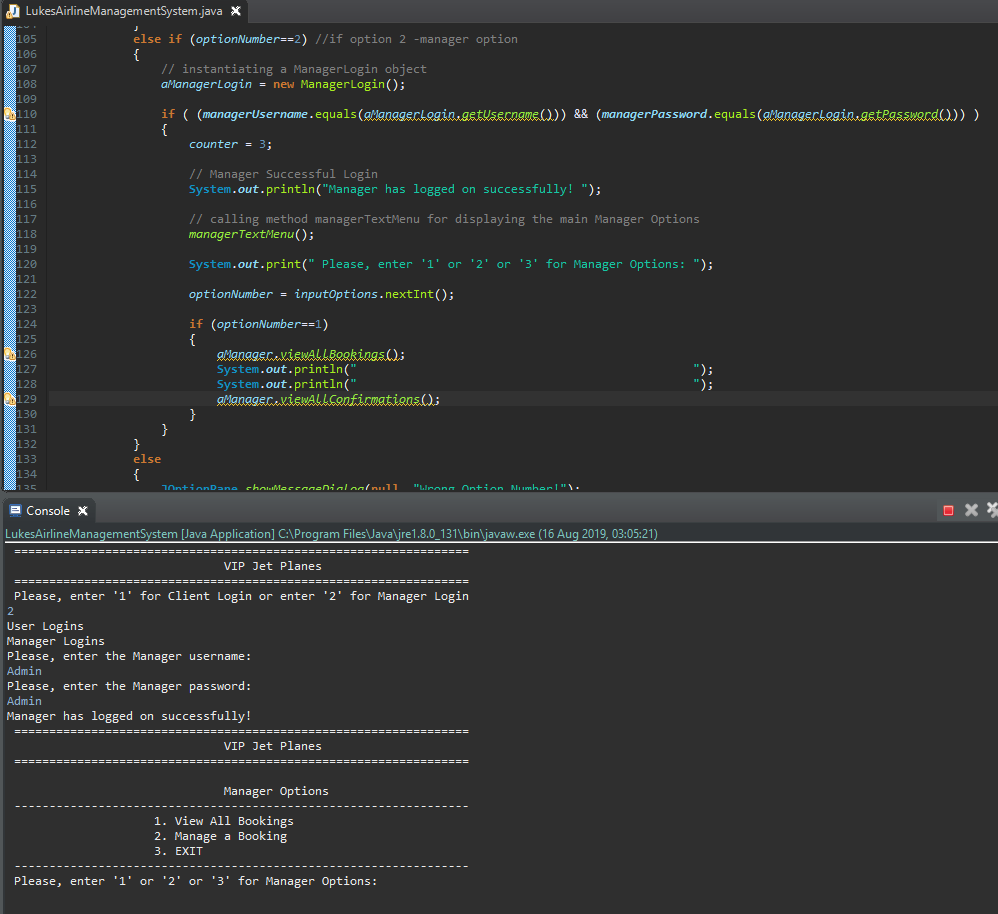
**Client**

|  |  |
| --- | --- |
| **First Refinement** | **Second Refinement** |
| * Initialise instance fields  -username  -password * Enter username / password * Enter an option   -Client views all planes  -Client creates a booking  -Client edits a booking  -Exit airline application | * Initialise instance fields  -String username = “ ”;  -String password = “ “;  -Int optionNumber = 1 * Read in username * Read in password * If username / password correct  -Read in optionNumber * Client views all planes  -display planes text  -client creates a booking OR * Client Edits a booking   OR   * Manager exits application   -return to main login menu |

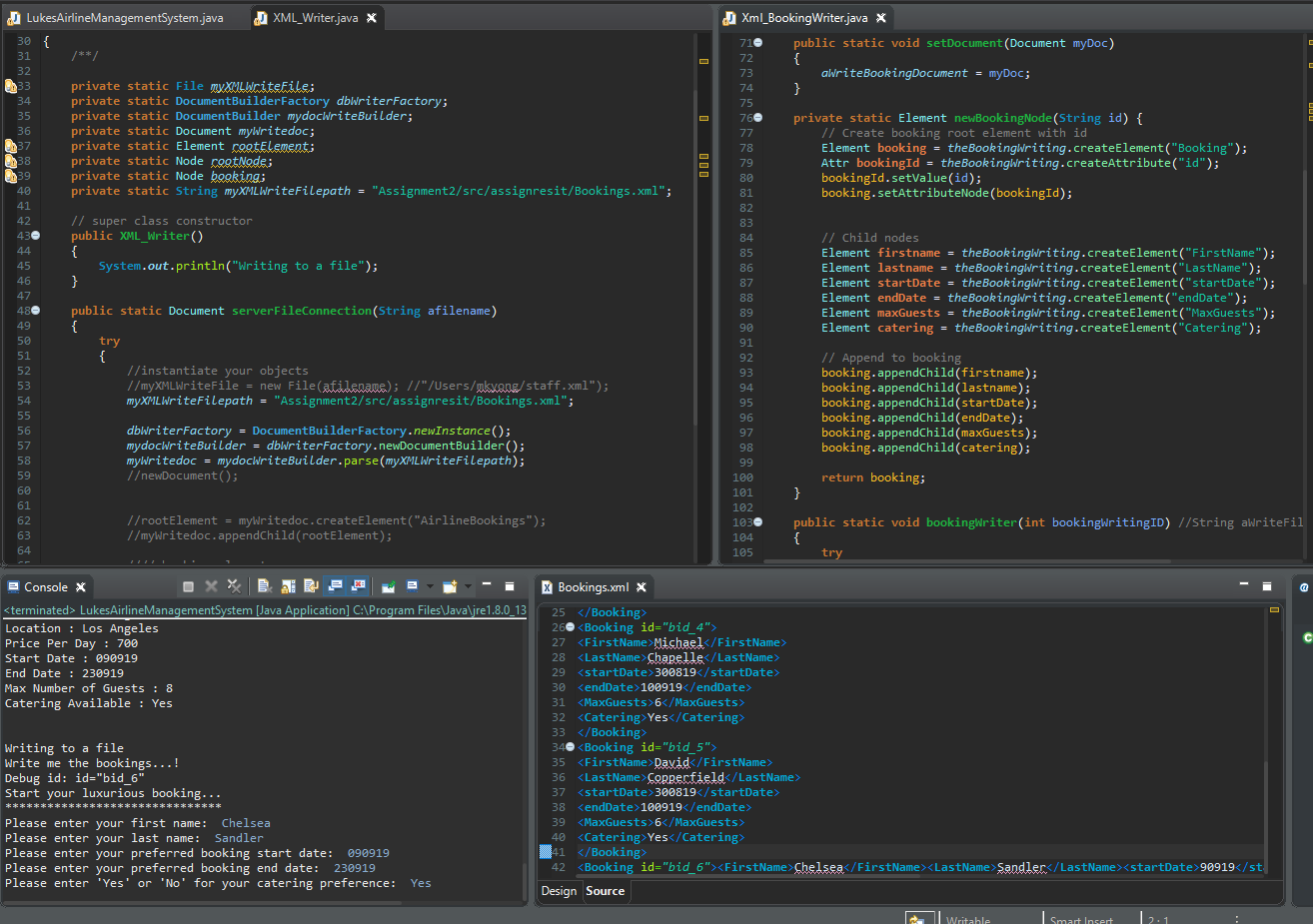
|  |  |
| --- | --- |
| **Third Refinement** | **Final Refinement** |
| * Initialise instance fields  -String username = “ ”;  -String password = “ “;  -Int optionNumber = 1 * Read in the username value using keyboard * Read in password value using keyboard * If username & password match the saved username value and the saved password value from the hardcoded details  -Read in the optionNumber value   using keyboard   * If optionNumber is 1  -client is allowed to view all of   the planes, displays text for each  plane  -then displays text for booking a  plane   * If optionNumber is 2  -client enters his Booking ID   -If Booking ID matches a BID in the  booking file  -then display text of previous  booking ready for editing   * If optionNumber is 3   -client exits application back to the  login stage  - | * Initialise instance fields  -String username = “ “;  -String password = “ “;   -Int optionNumber = 1   * Read in the username value using keyboard * Read in password value using keyboard * While (username AND password match the saved username value and AND password value that are hardcoded)   -Then read in the optionNumber  value using keyboard   * If optionNumber is 1   -client is allowed to view all of  the planes, displays text for each  plane  -then displays text for booking  information to be entered     * Elseif optionNumber is 2   -client is allowed to enter their   Booking ID  While ( Booking ID is true)   -then display the text with the booking  information ready to edit booking  Else (Booking ID is false)  -display text “Booking ID does not exist”  -return to start asking for login details   * Elseif optionNumber is 3   -then the client exits the application and returns to the login stage   * Else  -display the message “incorrect option has been selected !” |

**Client Login & Options Menu**

This part of the system allows a user to enter a number 1 into the console then be able to login using a hardcoded username and password which then continues by displaying an option menu to select from 3 options, do a booking, manage a booking or exit the airline application.

**Manager Login & Options Menu**

This part of the system allows a user to enter a number 2 into the console then be able to login using a hardcoded username and password which then continues by displaying an option menu to select from 3 options, view all bookings, manage a booking or exit the airline application

**Client – Create a Booking**

For the client create a booking the system outputs into the console all the planes and flight information that are available. The user can then input the information which is passed through the xml writer and the xml bookings writer to write the information input into the bookings.xml file. In the example above you can see the console being run and the information outputting into the bookings.xml on the right.

**References**

Brekelmans, B. & Software Analysis Transformation, 2014. What Programmers do with Inheritance in Java and C#. , p.2014.

Jinghua Zhang, E., Caldwell & Smith, 2013. Learning the concept of Java inheritance in a game. *Proceedings of CGAMES'2013 USA*, pp.212–216.

Fain, Y., 2011. *Java programming 24-hour trainer*, Hoboken, IN: John Wiley.

Rountev, A., Milanova & Ryder, 2004. Fragment class analysis for testing of polymorphism in Java software. *IEEE Transactions on Software Engineering*, 30(6), pp.372–387.