Pearson Higher Nationals in

Computing

Unit 20: Advanced Programming

Issue 1

Pearson

Higher National Certificate/Diploma in Computing

Assignment Brief

Student Name/ID Number	
Unit Number and Title	**20: Advanced Programming**
Academic Year	
Unit Tutor	
Assignment Title	**UML Documentation**
Issue Date	
Submission Date	
IV Name & Date	

Submission Format
Part 01 : • 10 min presentation about the OOP Concept • Formal report which analysis the relationship between OO paradigm and design patterns Part 02 : • The submission should be in the form of a series of UML diagrams, collated into a single PDF. Using a suitable UML tool. Part 03 : • Documentation for program implementation including the relevant class codes and design patterns • Report to evaluate the use of design patterns for the given purpose. Part 04 : • 10 min presentation to evaluate the different design patterns

Unit Learning Outcomes
LO1 Examine the key components related to the object-oriented programming paradigm, analysing design pattern types LO2 Design a series of UML class diagrams LO3 Implement code applying design patterns LO4 Investigate scenarios with respect to design pattern

Assignment Brief and Guidance

You are required to complete a program that implements a basic Ships Booking System (SBS). The SBS allows a user to create harbors, shipping company, and ships for the shipping company. Each Shipping company is associated with a set of ships. A ship has an originating harbor and a destination harbor. The originating and destination harbors cannot be the same. Each ship is associated with a three section (first class, second class and third class). Each ship class consists of seats organized in rows and columns.

1. **Create harbor**: A harbor must have a name consisting of exactly three alphabetic characters. No two harbors can have the same name.
2. **Create shipping company**: A shipping company has a name that must have a length less than 6. No two shipping company can have the same name.
3. **Create a ship** given shipping company name, the name of an originating harbor, the name of a destination harbor, a ship number, and a departure date: A ship has an identifier that is a string of alphanumeric characters.
4. **Create a section for a ship**: The number of seat rows and columns must be provided when creating a section.
5. **Find available ships**: Finds all ships from an originating harbor to a destination harbor with seats that are not booked on a given date.
6. **Book a seat**: Books an available seat from a given originating harbor to a destination harbor on a particular date, on a given ship.
7. **Print system details**: Displays attribute values for all objects (e.g., harbor, shipping company) in system.

Your program must include the following classes.

Systems Manager: This class provides the interface to the system. That is, users interact with the system by calling operations in the Systems Manager. The Systems Manager is linked to all the harbor and ship company objects in the system. When it is created, the Systems Manager has no

harbor or shipping company objects linked to it. To create harbor and shipping company, the createHarbor() and createShippingCompany() operations defined in this class must be invoked.

The class also contains operations for creating sections of ships (e.g., first class, second class and third class), finding available ships between two harbors, and booking a seat on a ship. A printout of information on all the harbors, shipping company, ships, ship sections and seats is obtained by invoking displaySystemDetails()

CreateHarbor (String n): Creates an harbor object and links it to the Systems Manager. The harbor will have a name (code) n; n must have exactly three characters. No two harbors can have the same name.

CreateHarbor (String n): Creates an shipping company object with name n and links it to the Systems Manager. A shipping company has a name that must have a length less than 6. No two shipping company can have the same name.

CreateShip(String aname, String orig, String dest, int year, int month, int day, String id): Creates a ship for an shipping company named sname, from an originating harbor (orig) to a destination harbor(dest) on a particular date. The ship has an identifier (id).

CreateSection(String shi, String SlID, int rows, int cols, SeatClass s): Creates a section, of class s, for a ship with identifier SlID, associated with an shipping company, scid. The section will contain the input number of rows and columns.

FindAvailableShips(String orig, String dest): Finds all ships from harbor origin to harbor destination with seats that are not booked.

BookSeat(String air, String fl, SeatClass s, int row, char col): Books seat in given row and column in section s, on ship sl of shpping company scid.

DisplaySystemDetails(): Displays attribute values for all objects (e.g., harbors, ships) in system.

Harbor: Objects of this class represent harbors. The only information maintained is the name, which must be exactly 3 characters in length.

Shipping company: This class maintains information about Shipping company. A Shipping company can have 0 or more ships associated with it. When created a Shipping company is not associated with any ships. All ships for a given Shipping company must have unique ship ids.

Ship: This class maintains information about ships. A ship can be associated with 0 or more ship sections. There can only be one ship section of a particular seat class in a ship, e.g., only one first class, one second class and one third class. The seat classes are defined by the enumerator type SeatClass which defines the values first, second and third. The major operations of Ships are summarized below.

ShipSection: This class maintains information about ship sections. A ship section has a seat class (first, second or third) and must have at least 1 seat. hasAvailableSeats() returns true if the section has some seats that are not booked, and bookSeat() books an available seat. A ship section can contain at most 100 rows of seats and at most 10 columns of seats.

Seat: This class maintains information about seats. Specifically, a seat has an identifier (a seat is identified by a row number and a column character, where the character is a letter from A to J), and a status which indicates whether the seat is booked or not.

Part 01 :

Before starting to implement the above project your senior officer ask you to prepare a presentation and the report which explain the object oriented concept to your juniors.

a) Create a 10 Min presentation about the object oriented programming paradigm including the following:

1. Different between object and class
2. Characteristics of the object oriented paradigm
3. Object oriented class relationship

b) Create a report to analysis the object oriented concept and design patterns. This report should determine a design pattern from creational, structural and behavioural pattern type. Should analyse the relationship between object oriented paradigm and design pattern.

Part 02:

As a program designer you should design and build a **Class Responsibility Collaboration** (**CRC**) **cards and** Class diagram for the above scenario by concentrating design patterns. You should use standard UML tool and need to show the relationship between identify classes.

Part 03

a) Implement above classes using preferred object oriented programming language and develop the code applying design patterns.
b) Create a report to evaluate the use of design patterns for the given purpose.

Part 04

Create a 10 min presentation to investigate and review the usage of range of design pattern with relevant examples. Critically evaluate the range of design pattern and justify your choice.

<table>
<tr><th colspan="3">Learning Outcomes and Assessment Criteria</th></tr>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">LO1 Examine the key components related to the object-oriented programming paradigm, analysing design pattern types</td><td rowspan="2">D1 Analyse the relationship between the object-oriented paradigm and design patterns.</td></tr>
<tr><td>P1 Examine the characteristics of the object-oriented paradigm as well as the various class relationships.</td><td>M1 Determine a design pattern from each of the creational, structural, and behavioural pattern types.</td></tr>
<tr><td colspan="2">LO2 Design a series of UML class diagrams</td><td rowspan="2">D2 Define/refine class diagrams derived from a given code scenario using a UML tool.</td></tr>
<tr><td>P2 Design and build class diagrams using a UML tool.</td><td>M2 Define class diagrams for specific design patterns using a UML tool.</td></tr>
<tr><td colspan="2">L03 Implement code applying design patterns</td><td rowspan="2">D3 Define/refine class diagrams derived from a given code scenario using a UML tool.</td></tr>
<tr><td>P3 Build an application derived from UML class diagram.</td><td>M3 Develop code that implements a design pattern for a given purpose.</td></tr>
<tr><td colspan="2">L04 Investigate scenarios with respect to design patterns</td><td rowspan="2">D4 Critically evaluate a range design patterns against the range of given scenarios with justification of your choices.</td></tr>
<tr><td>P4 Discuss a range of design patterns with relevant examples of creational, structural and behavioural pattern types.</td><td>M4 Reconcile the most appropriate design pattern from a range with a series of given scenarios.</td></tr>
</table>