

**Laboratory MANUAL**

**Semester: VI**

|  |
| --- |
| **ISL67 Object ORIENTED ANALYSIS AND DESIGN PATTERNS LABORATORY** |

Prepared by:

Mrs. Evangeline D

# Academic Year : 2021 - 2022

**TABLE OF CONTENTS**

***Sl.No. Contents PageNo.***

|  |  |  |
| --- | --- | --- |
| ***1.*** | Vision, Mission, PEO, PSO, PO | **1** |
| ***2.*** | Introduction and scope of the course | **4** |
| ***3.*** | Syllabus | **5** |
| ***4.*** | Rubrics for Evaluation | **9** |
| ***5.*** | Design of solutions | **10** |
| ***6.*** | References | **29** |

**Vision and Mission of the Department**

**Vision**

To evolve as an outstanding education and research centre of information technology to create high quality engineering professionals for the betterment of society

**Mission**

**Department of Information Science and Engineering shall create high quality IT Engineering Professionals for the betterment of society by**

* Providing education through an ever improving curriculum and effective pedagogy techniques.
* Department of Information Science and Engineering shall create high quality IT Engineering Professionals for the betterment of society by encouraging extra and cocurricular activities to develop their overall personality along with technical skills.
* Department of Information Science and Engineering shall create high quality IT Engineering Professionals for the betterment of society by collaborating with industry and academia for strengthening research, innovation and entrepreneurship ecosystem.

**Program Educational Objectives (PEOs)**

|  |  |
| --- | --- |
| **PEO No.** | **Program Educational Objectives Statements** |
| PEO1 | Become competent information technology professionals with continuous progress in career or learning |
| PEO2 | Productively engage with society by practicing research or entrepreneurship. |
| PEO3 | Function effectively as professionals in a team environment or individually |

**Program Outcomes**

|  |  |
| --- | --- |
| PO- 1 | Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems. |
| PO- 2 | Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences |
| PO- 3 | Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations. |
| PO- 4 | Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions. |
| PO- 5 | Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations |
| PO- 6 | : The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice. |
| PO- 7 | Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development. |
| PO- 8 | Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice. |
| PO- 9 | Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings. |
| PO- 10 | Communication: Communicate effectively on complex engineering 7 activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions. |
| PO- 11 | Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments. |
| PO- 12 | Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change. |

**Program Specific Objectives (PSOs)**

|  |  |
| --- | --- |
| **PSO1** | Apply Mathematical models, programming paradigms and software development practices to solve real world problems |
| **PSO2** | Adopt computing and communication models for developing IT solutions. |
| **PSO3** | Acquire data engineering skills to develop intelligent systems in a multidisciplinary environment. |

**INTRODUCTION AND SCOPE OF THE COURSE**

Design Patterns is a solution to commonly recurring problem in software design. All the design patterns are based on the strategies given by Gang of Four (GoF).

The strategies are as follows

1. Design to Interfaces
2. Favor aggregation or composition over inheritance
3. Find what varies and encapsulate it.

Design Patterns will enable the development of sustainable solutions as it follows Open Closed Principle. This principle insists that any design is open for extension and closed for modification.

Cohesion and Coupling are two important parameters in software design. While cohesion is the strength of functional relation between the elements in a module, coupling is the degree of interdependence between modules. A good software design requires loose coupling and strong cohesion.

Design Patterns are broadly categorized into three classes.

1. Creational Design Pattern: It is used to create or instantiate objects. Abstract Factory, Singleton and Factory Method are creational design patterns.
2. Structural Design Pattern: It is used to bring together existing objects. Façade, Adapter, Bridge and Decorator are structural design patterns.
3. Behavioral Design Pattern: It gives a way to manifest flexible or varying behavior. Strategy is a behavioral design pattern.

**SYLLABUS**

**PART A**

***Case-study to understand the limitation of traditional Object Oriented Design and appreciate need for DesignPatterns. Use UML Notations to design.***

You are a fresh analyst deputed to design the software for Decathlon Chain of Stores in Karnataka. You are informed about the Business Logic of Point of Sales criteria by Ms.Veronica Lodge, a dynamic business tycoon operating out of Decathlon Mumbai. She informs you that there are different types of Customers of Decathlon namely, Regular Customers, Senior Citizens and First Time Customers. Regular Customers are given a discount of 12%, Senior Citizens 10% and First Time Customers 15%. Apart from this, based on the sales-index of previous day, a Store-level discount is determined every day. This is dynamic. **E.g**.Rs.100 off for every purchase above Rs.2000. Using the Object Oriented Principles of Encapsulation, Abstraction, Inheritance, Composition and Aggregation that you have studied until this semester, give at least two ways to design this system.

**PART B**

**Common Case Study for Q#1 to Q#9**

‘Decathlon’ is a Sports retail-store started in France. Today it spreads across 22 countries & has 900 outlets in these countries. It has a ‘Point of Sale’ software system called ‘Decathlon POS’, which uses various kinds of 3rd-party software sourced locally from the various countries they are established. You are a software consultant for Decathlon, in Bangalore, with a team of consultants reporting to you. When you analyze your answer for choosing a pattern, explain wherever applicable, keeping in mind the following four design principles:

* Separation of concerns
* Program to an interface, not a concrete implementation
* Prefer composition over inheritance
* Open-Close principle (Open for extension, Closed for modification)

1. **Adaptor (Structural):** To establish the 1st Decathlon store in Mauritius, you go along with Mr. SatyaNadella,an expert in finding 3rd-party partners. For e.g. a 3rd-party Tax-Calculator system to cater to the specifics of Salesand VAT (Value-added services Tax) tax calculations in different countries. He finds a 3rd-party Tax-Calculatorsystem called ‘MauriTax’ in Port Louis. The problem is, the APIs used by ‘MauriTax’ for tax-calculation is fixed& cannot be changed. *The ‘MauriTax’ APIs are incompatible with ‘Decathlon POS’*.

How will you use theAdaptor Pattern to design & implement?

1. **Strategy (Behavioural):** How will you use the Strategy Pattern to tackle the limitations of traditional ObjectOriented Design highlighted in PART A? *The design must handle varying price-schemes having different pricingalgorithms.* Design & implement.
2. **Factory Method (Creational):** The ‘Decathlon POS’ software system classifies its customers as senior-citizens(60 and above), First-Time customers, Regular Customers. There is a very high possibility that the CustomerType hierarchy will vary, depending upon the sales-pattern. **For e.g.** there could be the need to introduce newcategories based on the customer gender, different age groups for kids (0-5, 6-12), teenagers (13-19) and agegroupsbetween 20 to 60(Twenties, 30s, 40s and 50s).

You are advised by Mr.SundarPichai, the technical architect of your team, whom you trust, to use Factory MethodPattern in order to instantiate the above Customer Type hierarchy of concrete implementation of objects. Designand implement using this.

1. **Bridge (Structural):** You get a call from Ms.Masaba Gupta of Bangalore Decathlon office that there is a policy decision made globally to introduce discount slabs for a whole month twice in a year. The discount month will be in January and July after reviewing the sales made from Feb to June (first five months) and Aug to December (last five months) respectively. It is decided to provide four slabs of discounts in 2017, namely, 30%, 25%, 20% and 15%, based on the sports item purchased. **For e.g.** all tennis rackets could have a 20% discount while cricket bats could only have a 15% discount. All exercise tread-mills could be given a 30% discount while boxing-gloves could have a 25% discount. Point to be noted here is that, the slabs of discount may not remain the same in 2018. It is likely to vary year after year. The ‘Decathlon POS’ software system classifies its customers as Senior-Citizens (60 and above), First-Time Customers, Regular Customers as of now. There is a very high possibility that the Customer Type hierarchy will vary, depending upon the sales-pattern. **For e.g.** there could be the need to introduce new categories based on the customer gender.

Use the Bridge Pattern to design & implement, *so that both the Customer Type hierarchy of classes as well as the Discount Percentage hierarchy of classes can both vary independently?* That is, they are not tied to each other.

1. **Observer (Behavioural):** There will be different discounts being offered for the sports items in Decathlon Stores across the globe for different festivals being celebrated in the various countries these stores are established. Assume that the Decathlon Chain of Stores fixes a particular discount slab for its items for a festival of a country.

Use the Observer Pattern to design and implement a system to notify the customers of the Decathlon stores of that country about the various festival / seasonal discount rates as and when they are announced.

1. **Façade (Structural):** You get a call from Ms.Betty Cooper of Bangalore Decathlon office that there is a policy decision made globally to incorporate some new rules for ‘Process Sale Use-Case’. **For e.g.** if payment is made via gift-certificate, the customer can buy only one item for the amount in the certificate. No other items can be bought with that gift-certificate. There must be no cash-back to the customer if the item costs less than amount specified in the gift-certificate. If the item costs more, the excess payment can be accepted via cash only & not credit / debit cards. When a new sale is created, these rules must become effective. You come to know from Mr. SatyaNadella, an expert in finding 3rd-party partners, that the Italian Competitor for Decathlon called ‘Sport 2000’ has a ready-made ‘rule-engine’ sub-system for this, whose specific implementation details is not known yet, as the business heads of Decathlon & Sport 2000 are chalking out the software purchase terms. This Sport 2000 rule-engine will be responsible for evaluating a set of rules against an operation & indicating if any of the rules invalidated the operation (e.g. ‘makeNewSale’ operation).

How will you use the Façade pattern to provide a common unified interface to a dissimilar set of implementations, developed by a 3rd-party vendor, the implementation details are not known to you?

1. **Abstract Factory (Creational):** As an analyst in charge of designing the Decathlon POS Software, you realize the need to streamline the creation of objects belonging to different products in the Decathlon store. There are two major categories of products:

**a)** For differently abled sports enthusiasts

**b)** For able-bodied sports enthusiasts

In each of the above categories there are products for outdoor adventure sports (e.g. trekking, para-gliding, bungee-jumping etc.), outdoor regular games (cricket, football, baseball etc.) indoor regular games (table tennis, squash etc.). There is a possibility of further class/object instantiation explosion with categories such as male & female sports enthusiasts and different equipment for them. *Objects need to be instantiated based on these categories*. Design & implement using Abstract Factory.

1. **Decorator(Behavioural – Structural according to GoF):** There is an existing interface method in the Decathlon POS software system called ‘getCurrentStock’ which is implemented by two concrete classes ‘IndoorSports’ and ‘OutdoorSports’, to get the number of stocks for the sports items belonging to these respective categories. On studying the Decathlon POS system, you as an analyst realize the need to get sports stock update of various items within:

IndoorSports - ‘GamesOnTable’ (e.g. Table Tennis, Billiards, Snooker etc.)

‘BoardGames’ (e.g. Carom, Chess etc.) ‘CourtGames’ (e.g. Basketball, Badminton, Kabaddi etc.)

OutdoorSports – ‘AdventureGames’ (e.g. trekking, para-gliding, bungee-jumping etc.)

‘StadiumGames’ (e.g. cricket, football, baseball etc.) ‘Athletics’ (e.g. different distances for running, high jump etc.)

Use the Decorator pattern, decorating the ‘getCurrentStock’ method to design and implement this scenario.

1. **Template Method (Behavioural):** To keep up with the customer convenience of online ordering DecathlonChain of stores decides to have two modes of order-processing, namely ‘online’ and ‘offline’. Both modes havethe same processing steps for order-processing, namely ‘selectItem’, ‘doPayment’ and ‘doDelivery’. But, the waythese steps are done varies between the two modes.

selectItem – online – gives tabular depiction of price comparison of the item chosen.Offline – allows trying out of the items in the store

doPayment – online – net-banking payment; offline – pays through cash / swipe-carddoDelivery – online – needs to pay the charges for shipping & delivery address; offline – collect at the counter.

Show how you as the analyst will use the Template Method pattern to design and implement this.

1. **Singleton (Creational):** A Browser’s history has data of all the visited URLs across all tabs and windows of a browser. The history is saved such that the data persists even after closing the browser. How would you useSingleton Pattern to implement Browser History such that on visiting a URL on any open tab of a browser theURL gets added to the existing history?

**Course Outcomes**

At the end of this course, the student will be able to:

|  |  |
| --- | --- |
| **CO#** | **Course Outcome Description** |
| **CO1** | Design the given problem using standard UML (**PO 2, PO 5) (PSO 1, 2, 3)** |
| **CO2** | Identify maintenance issues in the existing solutions **(PO 2, PO 6) (PSO 1, 2, 3)** |
| **CO3** | Explain the use of design patterns **(PO 10) (PSO 1, 2, 3)** |
| **CO4** | Apply Design patterns to the problems**(PO 2, PO 4, PO 11) (PSO 1, 2, 3)** |

**RUBRICS FOR EVALUATION – CONTINUOUS ASSESSMENT [10 Marks]**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Performance indicators** | **Excellent(E)** | **Good(G)** | **Fair(F)** | **PO’s** |
| **Understanding of Design Patterns**   1. **Marks)** | (1) Detailed and extensive explanation of the design patterns and its key features.  (2)Excellent understanding of the applicability of the design pattern. | (1) Good explanation of the design patterns and its key features.  (2)Good understanding of the applicability of the design pattern. | 1. Moderate explanation of the design patterns and its key features.   (2)Moderate understanding of the applicability of the design pattern. | PO1, PO6,  PO8,  PO9, PO10,PO12 |
| **UML Diagram & Design of solutions**  **(2 Marks)** | (1) Application of generic structure of the design pattern to the problem with accuracy and precision.  (2) Deep understanding of the problem at hand and usage of appropriate OO principles | (1) Application of generic structure of the design pattern to the problem with acceptable accuracy and precision.  (2) Good understanding of the problem at hand and usage of appropriate OO principles | (1) Application of generic structure of the design pattern to the problem with moderate accuracy and precision.  (2) Moderate understanding of the problem at hand and usage of appropriate OO principles | PO1, PO6,  PO8,  PO9, PO10,PO12 |
| **Implementation and Verification of Results**  **(5 Marks)** | (1) Excellent coding skills  (2) Great ability to perceive and analyse the results | (1) Appreciable coding skills  (2)Good ability to perceive and analyse the results | (1) Excellent coding skills  (2)Moderate ability to perceive and analyse the results | PO1, PO6,  PO8,  PO9, PO10,PO12 |
| **Documentation**  **(10 Marks)** | Students have followed the guidelines in writing the lab records and the results obtained for various combinations of input. | Students has partially followed the guidelines in writing the lab records and the results obtained for various combinations of input. | Students have to some extent only followed the guidelines in writing the lab records and the results obtained for various combinations of input. | PO9, PO10  PO12 |
| **Viva**  **(10 Marks)** | Students were able to answer the significance and application of the design patterns and were able to showcase the principles of design patterns with much clarity. | Students were able to answer the significance and application of the design patterns and were able to showcase the principles of design patterns to some extent. | Students were able to answer the significance and application of the design patterns. | PO9, PO10  PO12 |

**DESIGN OF SOLUTIONS**

**PART-A**

***Case-study to understand the limitation of traditional Object Oriented Design and appreciate need for DesignPatterns. Use UML Notations to design.***

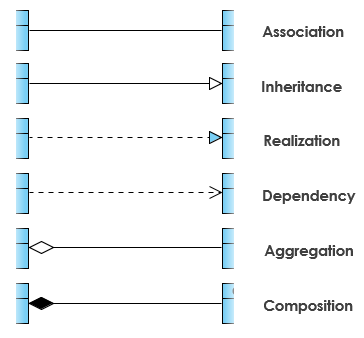
You are a fresh analyst deputed to design the software for Decathlon Chain of Stores in Karnataka. You are informed about the Business Logic of Point of Sales criteria by Ms.Veronica Lodge, a dynamic business tycoon operating out of Decathlon Mumbai. She informs you that there are different types of Customers of Decathlon namely, Regular Customers, Senior Citizens and First Time Customers. Regular Customers are given a discount of 12%, Senior Citizens 10% and First Time Customers 15%. Apart from this, based on the sales-index of previous day, a Store-level discount is determined every day. This is dynamic. **E.g**.Rs.100 off for every purchase above Rs.2000. Using the Object Oriented Principles of Encapsulation, Abstraction, Inheritance, Composition and Aggregation that you have studied until this semester, give at least two ways to design this system.

Solution:

Class is represented by a rectangle with three compartments. The first compartment contains class name.

Second compartment contains attributes and the third compartment contains methods.

Private, public and protected data members are marked as -, + and #. The data types of the members are given by representing it as “: datatype” following the name of the data member.



**PART-B**

**Common Case Study for Q#1 to Q#9**

‘Decathlon’ is a Sports retail-store started in France. Today it spreads across 22 countries & has 900 outlets in these countries. It has a ‘Point of Sale’ software system called ‘Decathlon POS’, which uses various kinds of 3rd-party software sourced locally from the various countries they are established. You are a software consultant for Decathlon, in Bangalore, with a team of consultants reporting to you. When you analyze your answer for choosing a pattern, explain wherever applicable, keeping in mind the following four design principles:

* Separation of concerns
* Program to an interface, not a concrete implementation
* Prefer composition over inheritance
* Open-Close principle (Open for extension, Closed for modification)

1. **Adaptor (Structural):** To establish the 1st Decathlon store in Mauritius, you go along with Mr. SatyaNadella,an expert in finding 3rd-party partners. For e.g. a 3rd-party Tax-Calculator system to cater to the specifics of Salesand VAT (Value-added services Tax) tax calculations in different countries. He finds a 3rd-party Tax-Calculatorsystem called ‘MauriTax’ in Port Louis. The problem is, the APIs used by ‘MauriTax’ for tax-calculation is fixed& cannot be changed. *The ‘MauriTax’ APIs are incompatible with ‘Decathlon POS’*.

How will you use theAdaptor Pattern to design & implement?

**Key Features of Adaptor Design Pattern**

Intent: Match an existing object beyond your control to a particular interface.

Problem: A system has the right data and behavior but the wrong interface. Typicallyused when you have to make something a derivative of an abstractclass we are defining or already have.

Solution The Adapter provides a wrapper with the desired interface.

Participants and Collaborators: The Adapter adapts the interface of an Adaptee to match that of the

Adapter's Target (the class it derives from). This allows the Clientto use the Adaptee as if it were a type of Target.

Consequences: The Adapter pattern allows for preexisting objects to fit into new classstructures without being limited by their interfaces.

Implementation: Contain the existing class in another class. Have the containing classmatch the required interface and call the methods of the contained class. The generic structure of the Adapter design pattern is given in Figure B.1.1

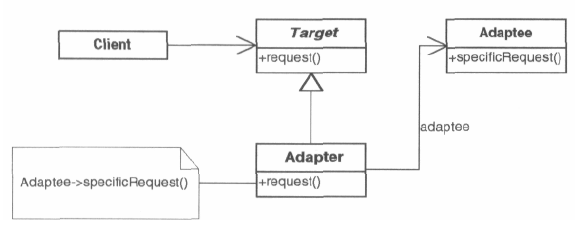


Figure B.1.1 Generic Structure of Adapter Design Pattern

The solution is designed as given in Figure B.1.2.

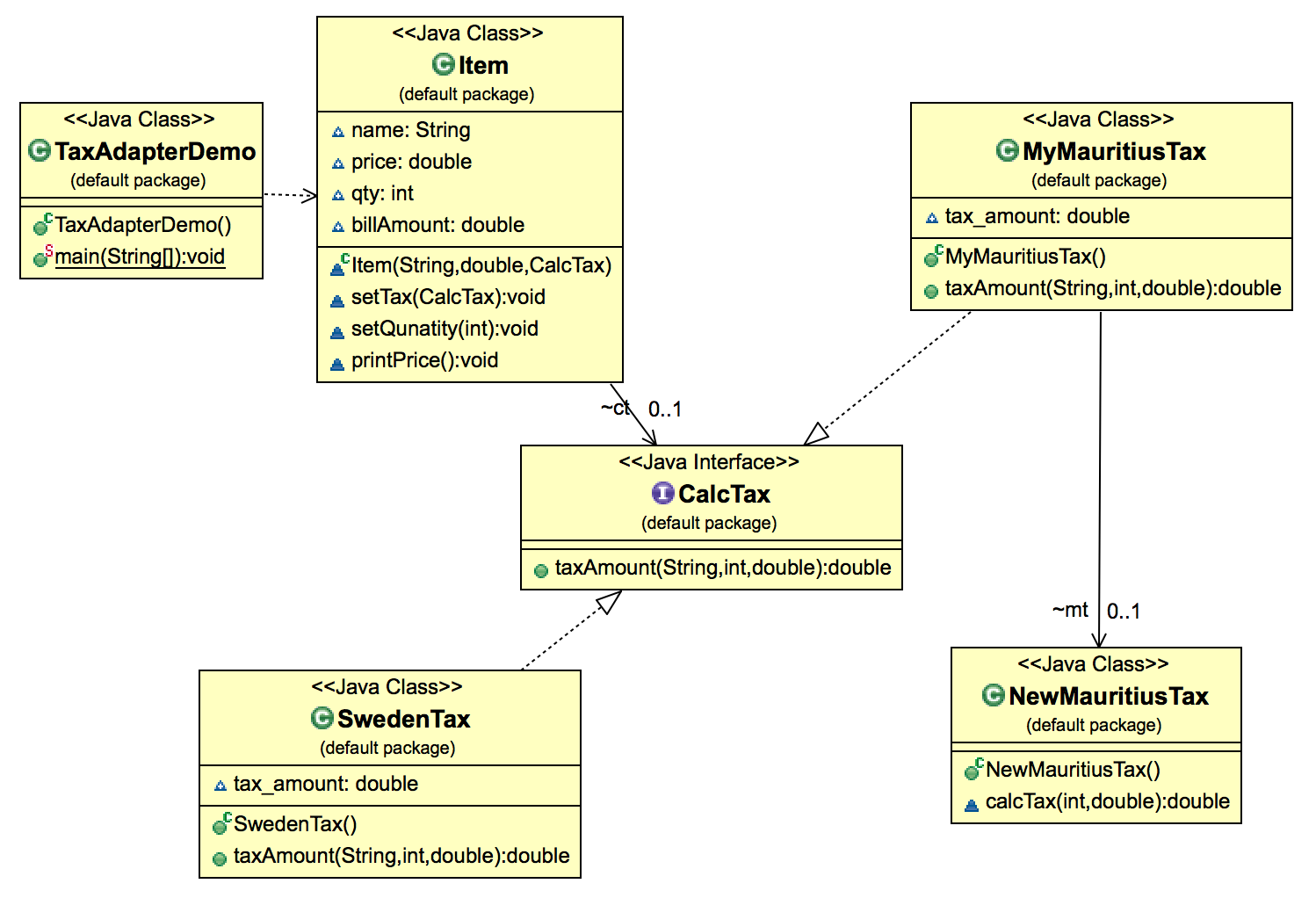


Figure B.1.2 Class Diagram Solution of Adapter Design Pattern Question

1. **Strategy (Behavioural):** How will you use the Strategy Pattern to tackle the limitations of traditional ObjectOriented Design highlighted in PART A? *The design must handle varying price-schemes having different pricingalgorithms.* Design & implement.

**Key Features of Strategy Design Pattern**

Intent: Allows you to use different business rules or algorithms depending upon the context in which they occur.

Problem: The selection of an algorithm that needs to be applied depends upon the client making the request or the data being acted upon.

Solution: Separates the selection of algorithm from the implementation of the algorithm. Allows for the selection to be made based upon context.

Participants and Collaborators: The strategy specifies how the different algorithms are used. The concreteStrategies implement these different algorithms.The Context forwards requests from its Client to the Strategy.

Consequences: The Strategy pattern defines a family of algorithms.Switches and/or conditionals can be eliminated.The interaction between the Concrete Strategies and the Context may require the addition of getstatetype methods to the Context.

Implementation: Have the class that uses the algorithm (the Context) contains an abstract class (the strategy) that has an abstract method specifying how to call the algorithm.

The generic structure of the Strategy design pattern is given in Figure B.2.1

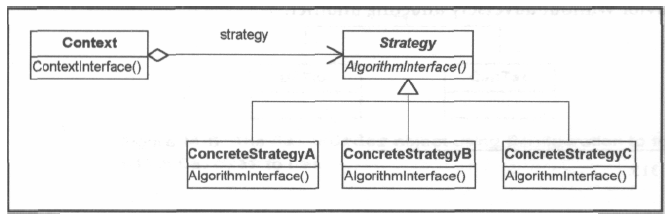


Figure B.2.1 Generic Structure of Strategy Design Pattern

The solution is designed as given in Figure B.2.2.

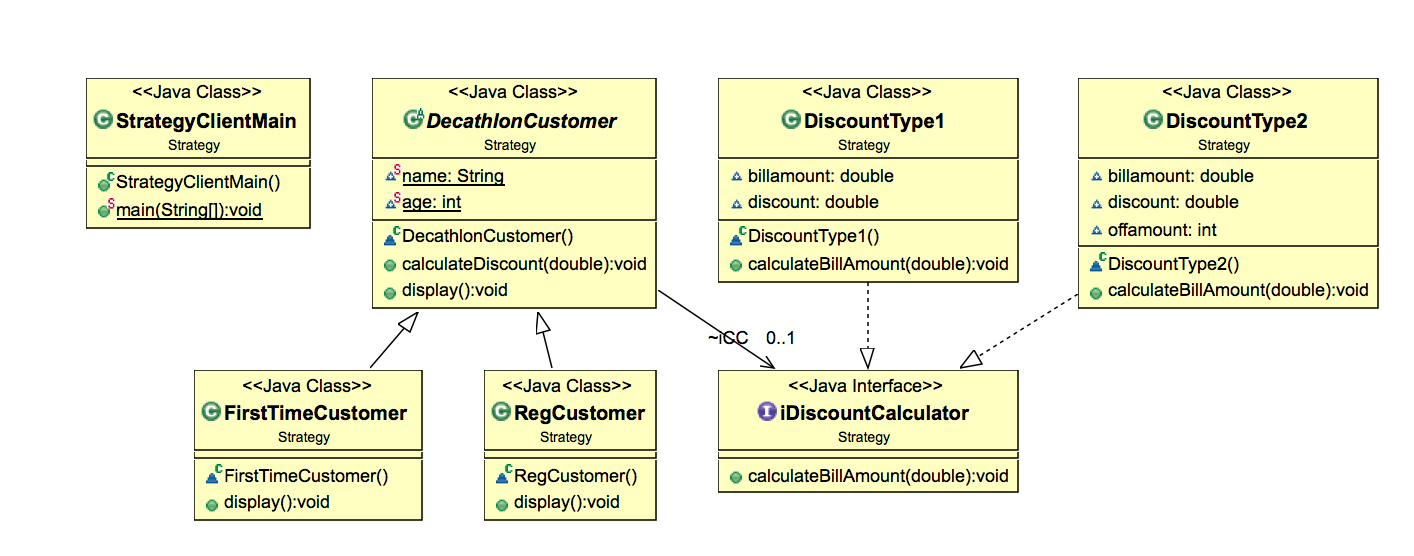


Figure B.2.2 Class Diagram Solution of Strategy Design Pattern Question

1. **Factory Method (Creational):** The ‘Decathlon POS’ software system classifies its customers as senior-citizens(60 and above), First-Time customers, Regular Customers. There is a very high possibility that the CustomerType hierarchy will vary, depending upon the sales-pattern. **For e.g.** there could be the need to introduce newcategories based on the customer gender, different age groups for kids (0-5, 6-12), teenagers (13-19) and agegroupsbetween 20 to 60(Twenties, 30s, 40s and 50s).

You are advised by Mr.SundarPichai, the technical architect of your team, whom you trust, to use Factory MethodPattern in order to instantiate the above Customer Type hierarchy of concrete implementation of objects. Designand implement using this.

**Key Features of Factory Method Design Pattern**

Intent: Define an interface for creating an object, but let subclasses decidewhich class to instantiate. Defer instantiation to subclasses.

Problem: A class needs to instantiate a derivation of another class, but doesn'tknow which one. Factory Method allows a derived class to make thisdecision.

Solution: A derived class makes the decision on which class to instantiate and

how to instantiate it.

Participants and Collaborators: Product is the interface for the type of object that the Factory Methodcreates. Creator is the interface that defines the Factory Method.

Consequences: Clients will need to subclass the Creator class to make a particular

ConcreteProduct.

Implementation: Use a method in the abstract class that is abstract. The abstract class' code refers to this method when it needs toinstantiate a contained object but does not know which particularobject it needs.

The generic structure of the Strategy design pattern is given in Figure B.3.1

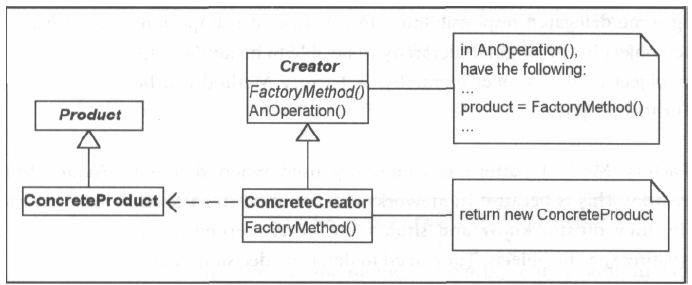


Figure B.3.1 Generic Structure of Factory Method Design Pattern

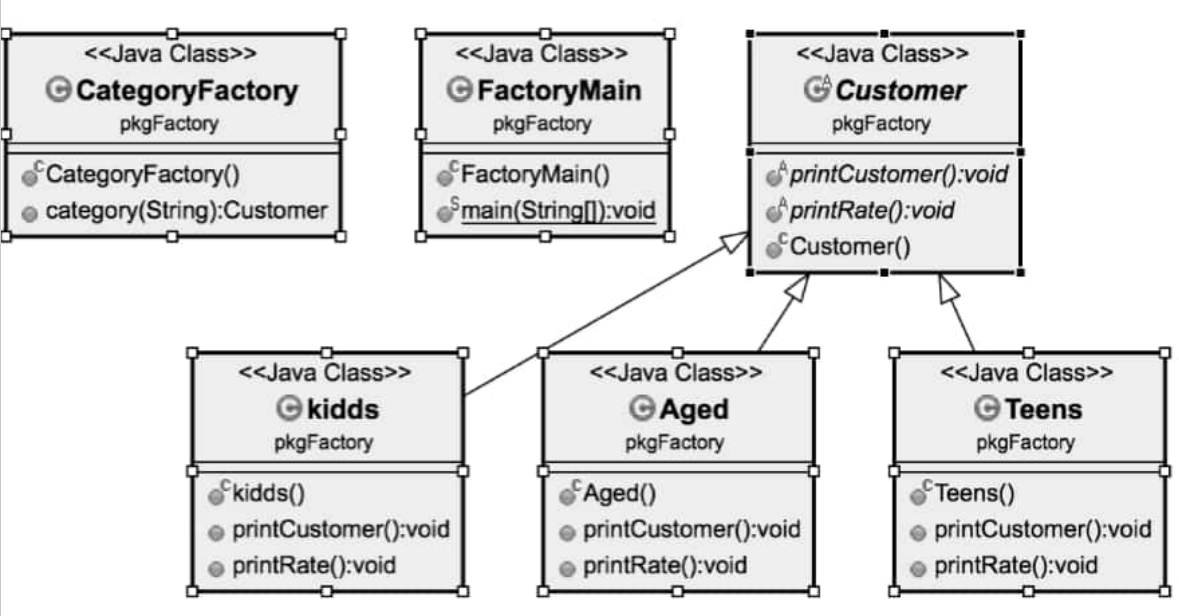
The solution is designed as given in Figure B.3.2.

Figure B.3.2 Class Diagram Solution of Factory Method Design Pattern Question

1. **Bridge (Structural):** You get a call from Ms.Masaba Gupta of Bangalore Decathlon office that there is a policy decision made globally to introduce discount slabs for a whole month twice in a year. The discount month will be in January and July after reviewing the sales made from Feb to June (first five months) and Aug to December (last five months) respectively. It is decided to provide four slabs of discounts in 2017, namely, 30%, 25%, 20% and 15%, based on the sports item purchased. **For e.g.** all tennis rackets could have a 20% discount while cricket bats could only have a 15% discount. All exercise tread-mills could be given a 30% discount while boxing-gloves could have a 25% discount. Point to be noted here is that, the slabs of discount may not remain the same in 2018. It is likely to vary year after year. The ‘Decathlon POS’ software system classifies its customers as Senior-Citizens (60 and above), First-Time Customers, Regular Customers as of now. There is a very high possibility that the Customer Type hierarchy will vary, depending upon the sales-pattern. **For e.g.** there could be the need to introduce new categories based on the customer gender.

Use the Bridge Pattern to design & implement, *so that both the Customer Type hierarchy of classes as well as the Discount Percentage hierarchy of classes can both vary independently?* That is, they are not tied to each other.

**Key Features of Bridge Pattern**

Intent: Decouple a set of implementations from the set of objects using them.

Problem: The derivations of an abstract class must use multiple implementations without causing an explosion in the number of classes.

Solution: Define an interface for all implementations to use and have the derivations of the abstract class use that.

Participants and Collaborators: The Abstraction defines the interface for the objects being implemented. The Implementor defines the interface for the specific implementation classes. Classes derived from the Abstraction use classes derived from the Implementor without knowing which particular ConcreteImplementor is in use.

Consequences: The decoupling of the implementations from the objects that use them increases extensibility. Client objects are not aware of implementation issues.

Implementation: Encapsulate the implementations in an abstract class. Contain a handle to it in the base class of the abstraction being implemented.

The generic structure of the Strategy design pattern is given in Figure B.4.1

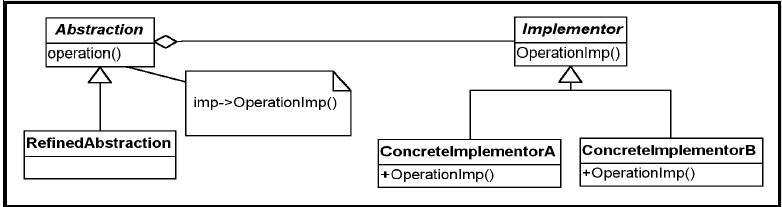


Figure B.4.1 Generic Structure of Bridge Design Pattern

The solution is designed as given in Figure B.4.2

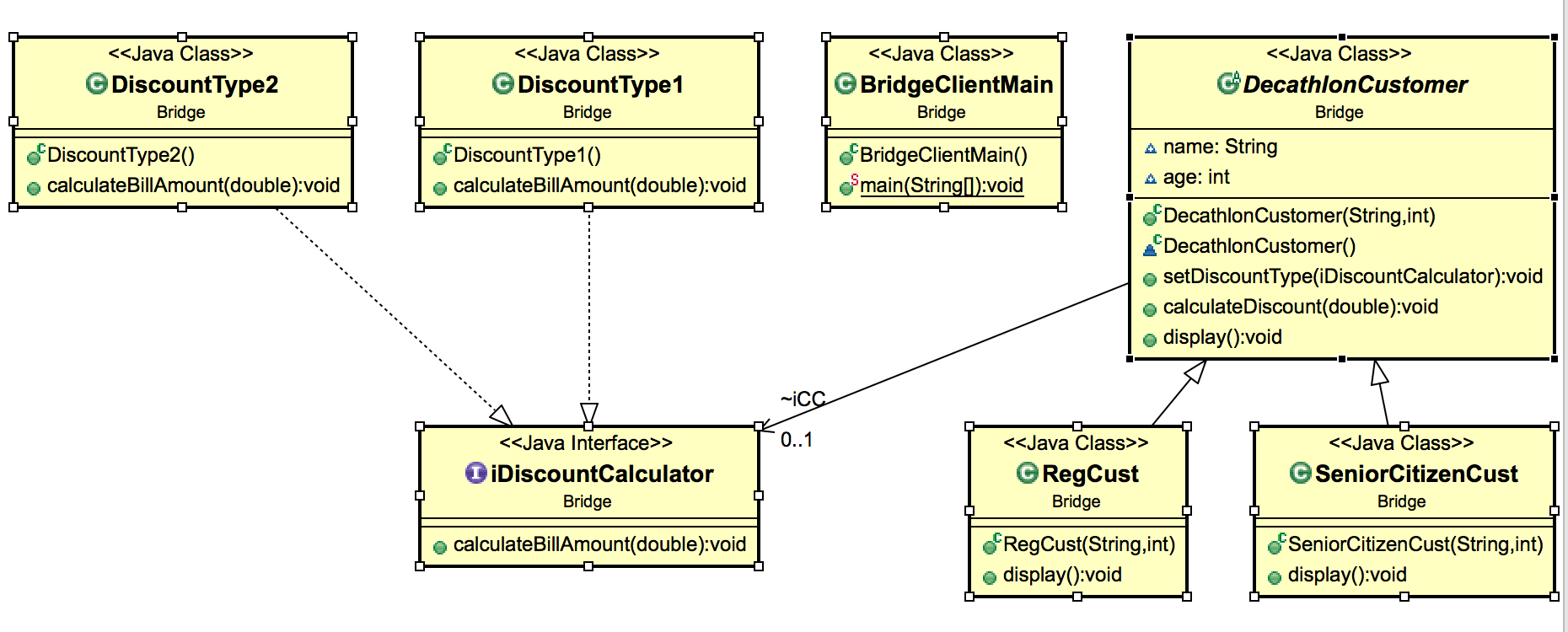


Figure B.4.2 Class Diagram Solution of Bridge Design Pattern Question

1. **Observer (Behavioural):** There will be different discounts being offered for the sports items in Decathlon Stores across the globe for different festivals being celebrated in the various countries these stores are established. Assume that the Decathlon Chain of Stores fixes a particular discount slab for its items for a festival of a country.

Use the Observer Pattern to design and implement a system to notify the customers of the Decathlon stores of that country about the various festival / seasonal discount rates as and when they are announced.

**Key Features of Observer Design Pattern**

Intent: Define a one-to-many dependency between objects so that when oneobject changes state, all its dependents are notified and updated automatically.

Problem :You need to notify a varying list of objects that an event has occurred.

Solution: Observers delegate the responsibility for monitoring for an event to a centralobject: the Subject.

Participants and Collaborators: The Subject knows its Observers because the Observers registerwith it. The Subject must notify the Observers when the event in questionoccurs. The Observers are responsible both for registering with theSubject and for getting the information from the Subject when notified.

Consequences: Subjects may tell Observers about events they do not need to know if

some Observers are interested in only a subset of events. Extra communicationmay be required if Subjects notify Observers which then go back andrequest additional information.

Implementation: Have objects (Observers) that want to know when an event happensattach themselves to another object (Subject) that is watching for theevent to occur or that triggers the event itself.When the event occurs, the Subject tells the Observers that it hasoccurred.The Adapter pattern is sometimes needed to be able to implement theObserver interface for all of the Observer-type objects.

The generic structure of the Observer design pattern is given in Figure B.5.1

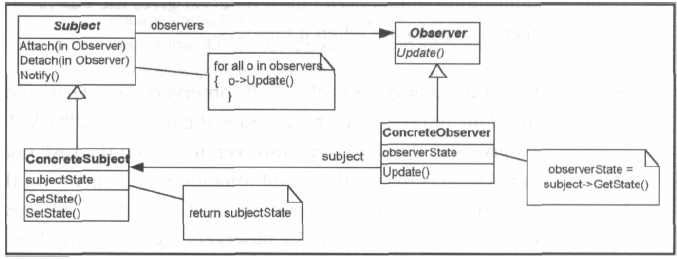


Figure B.5.1 Generic Structure of Observer Design Pattern

The solution is designed as given in Figure B.5.2

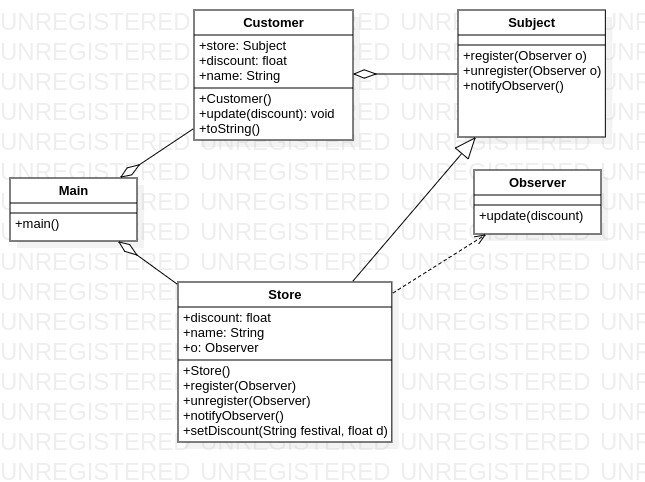


Figure B.5.2 Class Diagram Solution of Observer Design Pattern Question

1. **Façade (Structural):** You get a call from Ms.Betty Cooper of Bangalore Decathlon office that there is a policy decision made globally to incorporate some new rules for ‘Process Sale Use-Case’. **For e.g.** if payment is made via gift-certificate, the customer can buy only one item for the amount in the certificate. No other items can be bought with that gift-certificate. There must be no cash-back to the customer if the item costs less than amount specified in the gift-certificate. If the item costs more, the excess payment can be accepted via cash only & not credit / debit cards. When a new sale is created, these rules must become effective. You come to know from Mr. SatyaNadella, an expert in finding 3rd-party partners, that the Italian Competitor for Decathlon called ‘Sport 2000’ has a ready-made ‘rule-engine’ sub-system for this, whose specific implementation details is not known yet, as the business heads of Decathlon & Sport 2000 are chalking out the software purchase terms. This Sport 2000 rule-engine will be responsible for evaluating a set of rules against an operation & indicating if any of the rules invalidated the operation (e.g. ‘makeNewSale’ operation).

How will you use the Façade pattern to provide a common unified interface to a dissimilar set of implementations, developed by a 3rd-party vendor, the implementation details are not known to you?

**Key Features of Façade Pattern**

Intent:You want to simplify how to use an existing system. You need to define your own interface.

Problem:You need to use only a subset of a complex system. Or you need to interact with the system in a particular way.

Solution: The Facade presents a new interface for the client of the existing system to use.

Participants and Collaborators: It presents a specialized interface to the client that makes it easier to use.

Consequences: The Facade simplifies the use of the required subsystem. However, since the Facade is not complete, certain functionality may be unavailable to the client.

Implementation Define a new class (or classes) that has the required interface. Have this new class use the existing system.

The generic structure of the Facade design pattern is given in Figure B.6.1

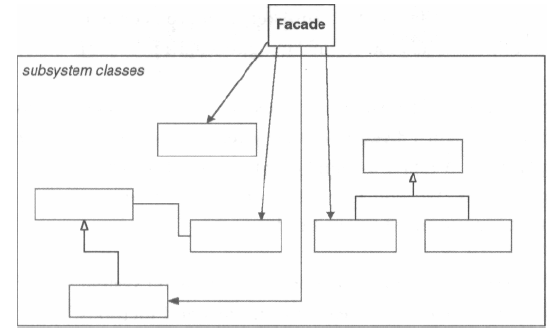


Figure B.6.1 Generic Structure of Facade Design Pattern

The solution is designed as given in Figure B.6.2

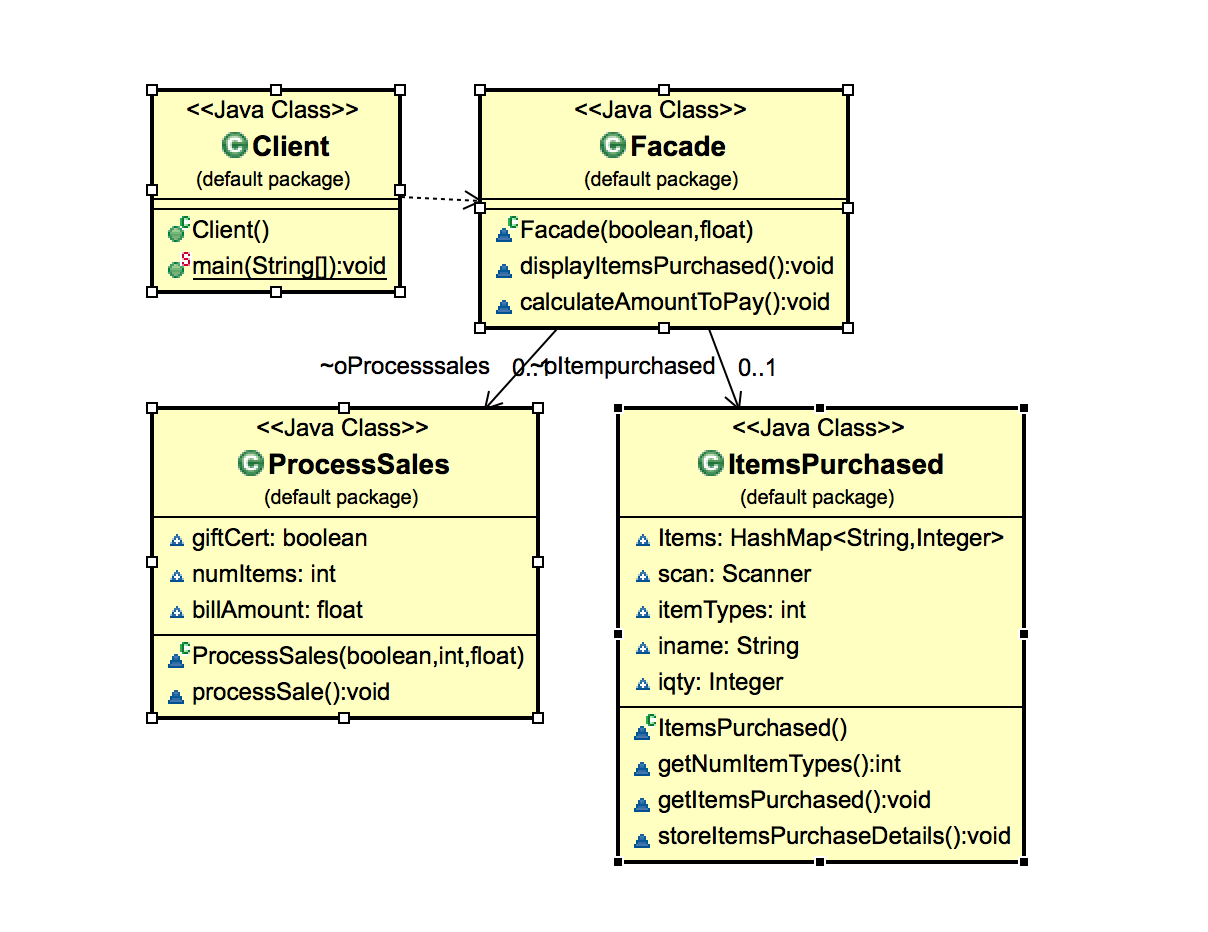


Figure B.6.2 Class Diagram Solution of Facade Design Pattern Question

1. **Abstract Factory (Creational):** As an analyst in charge of designing the Decathlon POS Software, you realize the need to streamline the creation of objects belonging to different products in the Decathlon store. There are two major categories of products:

**a)** For differently abled sports enthusiasts

**b)** For able-bodied sports enthusiasts

In each of the above categories there are products for outdoor adventure sports (e.g. trekking, para-gliding, bungee-jumping etc.), outdoor regular games (cricket, football, baseball etc.) indoor regular games (table tennis, squash etc.). There is a possibility of further class/object instantiation explosion with categories such as male & female sports enthusiasts and different equipment for them. *Objects need to be instantiated based on these categories*.

Design & implement using Abstract Factory.

**Key Features of Abstract Factory Design Pattern**

Intent: You want to have families or sets of objects for particular clients (or cases).

Problem: Families of related objects need to be instantiated.

Solution: Coordinates the creation of families of objects. Gives a way to take the rules of how to perform the instantiation out of the client object that is using these created objects.

Participants and Collaborators: The AbstractFactory defines the interface for how to create each member of the family of objects required. Typically, each family is created by having its own unique ConcreteFactory.

Consequences: The pattern isolates the rules of which objects to use from the logic of how to use these objects.

Implementation: Define an abstract class that specifies which objects are to be made. Then implement one concrete class for each family. Tables or files can also be used to accomplish the same thing.

The generic structure of the Abstract Factory design pattern is given in Figure B.7.1

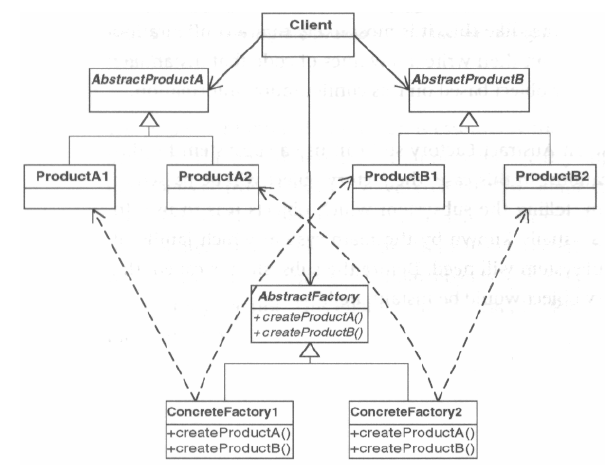


Figure B.7.1 Generic Structure of Abstract Factory Design Pattern

The solution is designed as given in Figure B.7.2

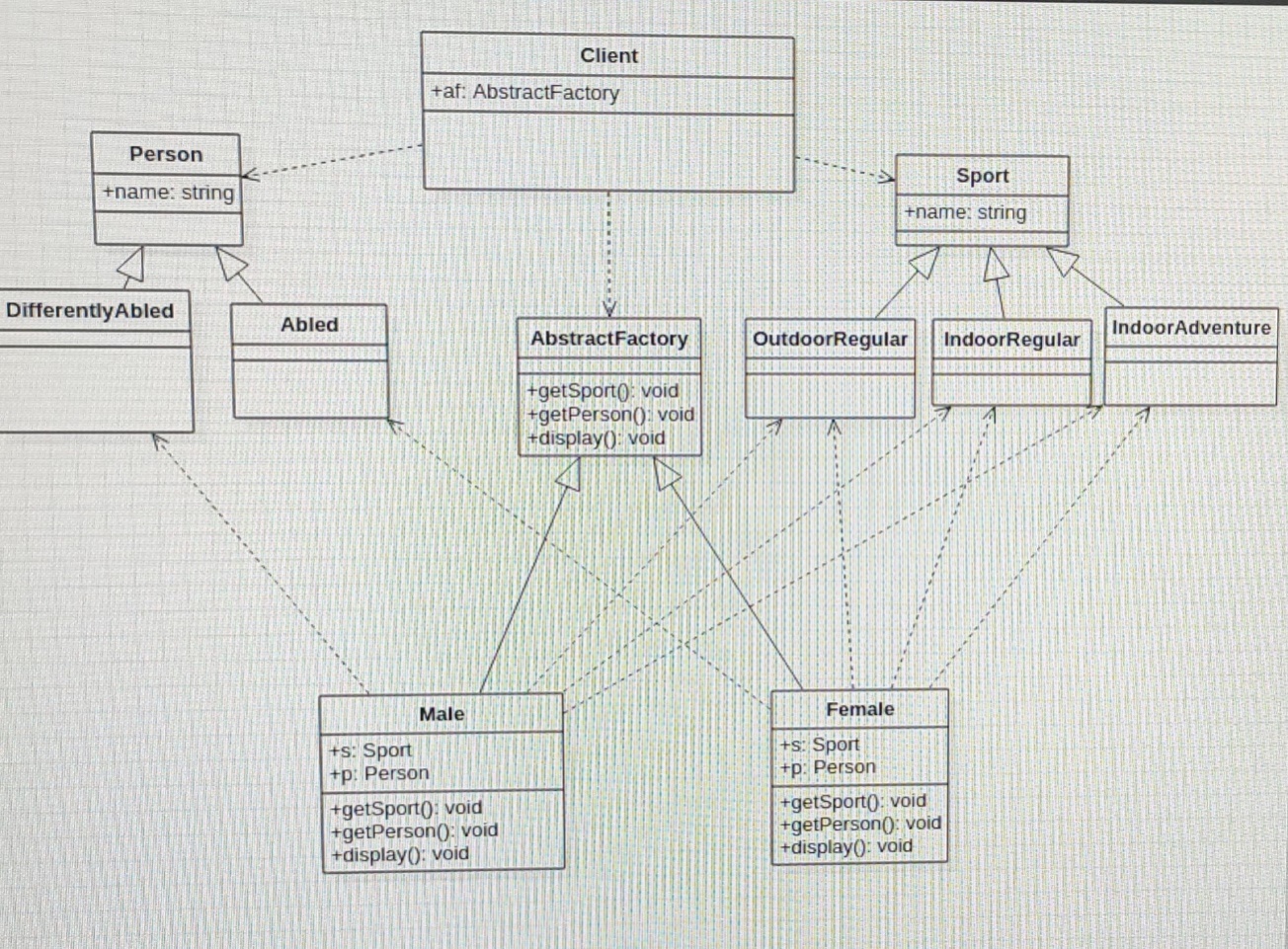


Figure B.7.2 Class Diagram Solution of Facade Design Pattern Question

1. **Decorator(Behavioural – Structural according to GoF):** There is an existing interface method in the Decathlon POS software system called ‘getCurrentStock’ which is implemented by two concrete classes ‘IndoorSports’ and ‘OutdoorSports’, to get the number of stocks for the sports items belonging to these respective categories. On studying the Decathlon POS system, you as an analyst realize the need to get sports stock update of various items within:

IndoorSports - ‘GamesOnTable’ (e.g. Table Tennis, Billiards, Snooker etc.)

‘BoardGames’ (e.g. Carom, Chess etc.) ‘CourtGames’ (e.g. Basketball, Badminton, Kabaddi etc.)

OutdoorSports – ‘AdventureGames’ (e.g. trekking, para-gliding, bungee-jumping etc.)

‘StadiumGames’ (e.g. cricket, football, baseball etc.) ‘Athletics’ (e.g. different distances for running, high jump etc.)

Use the Decorator pattern, decorating the ‘getCurrentStock’ method to design and implement this scenario.

**Key Features of Decorator Pattern**

Intent: Attach additional responsibilities to an object dynamically.

Problem: The object that you want to use does the basic functions you require.However, you may need to add some additional functionality to the object,occurring before or after the object's base functionality. Note that the Javafoundation classes use the Decorator pattern extensively for I/O handling.

Solution: Allows for extending the functionality of an object without resorting to subclassing.

Participants and Collaborators: The ConcreteComponent is the class having function added to it by the

Decorators. Sometimes classes derived from ConcreteComponentare used to provide the core functionality, in which caseConcrete-Component is no longer concrete, but rather abstract. The

Component defines the interface for all of these classes to use.

Consequences: Functionality that is to be added resides in small objects. The advantage isthe ability to dynamically add this function before or after the functionality inthe ConcreteComponent. *Note:* While a decorator may add itsfunctionality before or after that which it decorates, the chain of instantiation

always ends with the ConcreteComponent.

Implementation: Create an abstract class that represents both the original class and the new functions to be added to the class. In the decorators, place the new function calls before or after the trailing calls to get the correct order.

The generic structure of the Decorator design pattern is given in Figure B.8.1

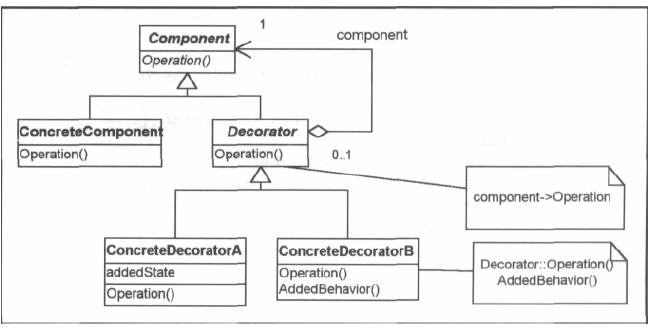


Figure B.8.1 Generic Structure of Decorator Design Pattern

The solution is designed as given in Figure B.8.2

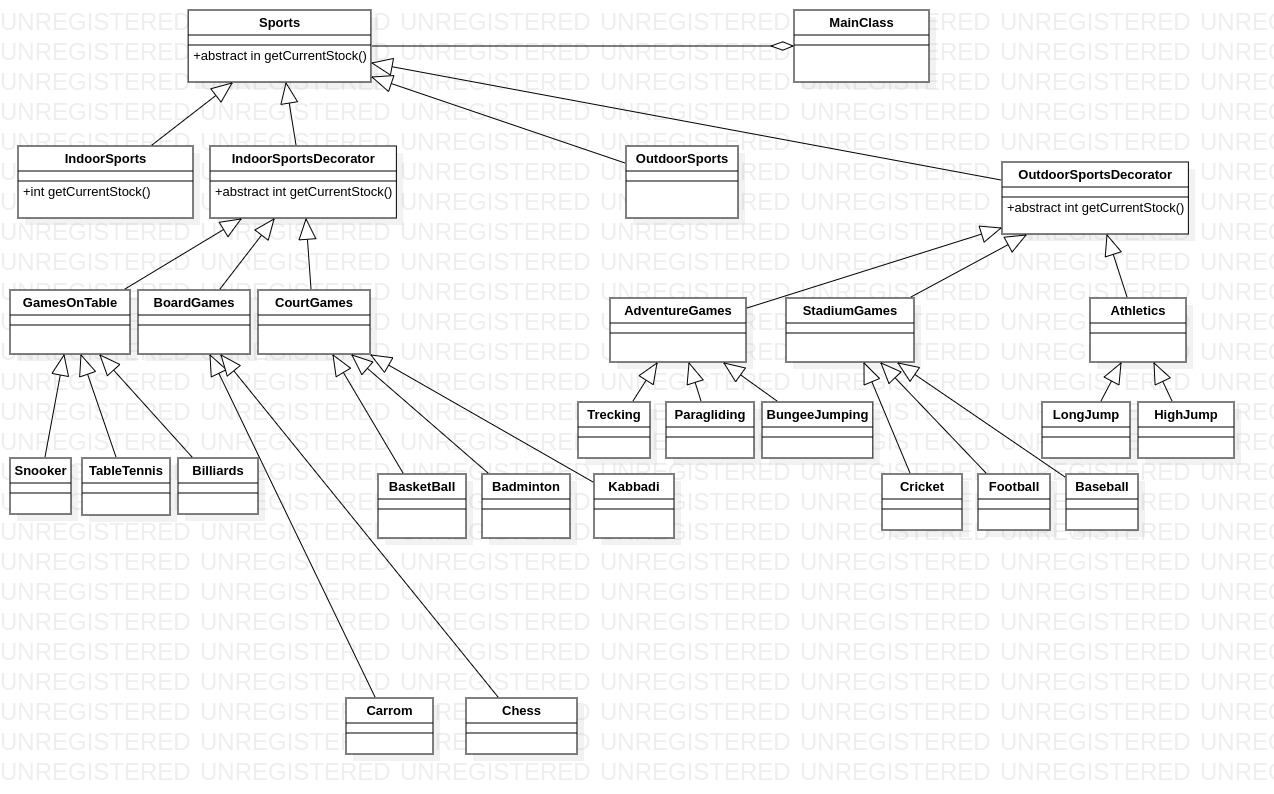


Figure B.8.2 Class Diagram Solution of Decorator Design Pattern Question

1. **Template Method (Behavioural):** To keep up with the customer convenience of online ordering DecathlonChain of stores decides to have two modes of order-processing, namely ‘online’ and ‘offline’. Both modes havethe same processing steps for order-processing, namely ‘selectItem’, ‘doPayment’ and ‘doDelivery’. But, the waythese steps are done varies between the two modes.

selectItem – online – gives tabular depiction of price comparison of the item chosen.Offline – allows trying out of the items in the store

doPayment – online – net-banking payment; offline – pays through cash / swipe-carddoDelivery – online – needs to pay the charges for shipping & delivery address; offline – collect at the counter.

Show how you as the analyst will use the Template Method pattern to design and implement this

**Key Features of Template Method Pattern**

Intent: Define the skeleton of an algorithm in an operation, deferring some steps to subclasses. Redefine the steps in an algorithm without changing the algorithm's structure.

Problem: There is a procedure or set of steps to follow that is consistent at onelevel of detail, but individual steps may have different implementations ata lower level of detail.

Solution: Allows for definition of substeps that vary while maintaining a consistentbasic process.

Participants and Collaborators: The Template Method consists of an abstract class that defines the basicTemplateMethod classes that need to be overridden.Each concrete class derived from the abstract class implements anew method for the Template.

Consequences: Templates provide a good platform for code reuse. They also are helpful in ensuring the required steps are implemented. They bind the overridden steps together for each Concrete class, and so should only beused when these variations always and only occur together.

Implementation: Create an abstract class that implements a procedure using abstract methods. These abstract methods must be implemented in subclasses to perform each step of the procedure. If the steps vary independently, each step may be implemented with a Strategy pattern.

The generic structure of the Template Method design pattern is given in Figure B.9.1.

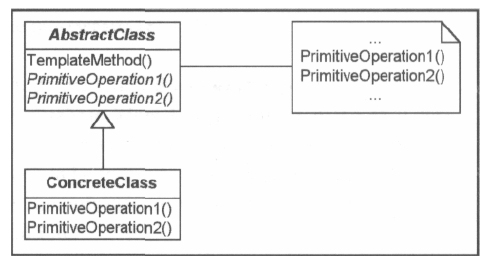


Figure B.9.1 Generic Structure of Template Method Design Pattern

The solution is designed as given in Figure B.9.2.

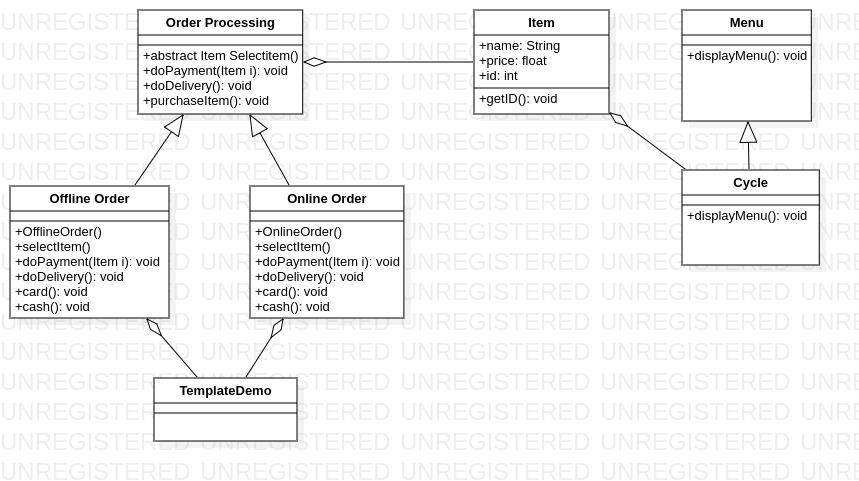


Figure B.9.2 Class Diagram Solution of Template Method Design Pattern Question

1. **Singleton (Creational):** A Browser’s history has data of all the visited URLs across all tabs and windows of a browser. The history is saved such that the data persists even after closing the browser. How would you useSingleton Pattern to implement Browser History such that on visiting a URL on any open tab of a browser theURL gets added to the existing history?

Key Features of Singleton Design Pattern

Intent: You want to have only *one* of an object but there is no global object that controls the instantiation of this object.

Problem: Several different client objects need to refer to the same thing and you want to ensure that you do not have more than one of them.

Solution: Guarantees one instance.

Participants and Collaborators: Clients create an get Instance of the Singleton solely through the

instance method.

Consequences: Clients need not concern themselves whether an instance of the Singleton exists. This can be controlled from within the Singleton.

Implementation: Add a private static member of the class that refers to the desired object (initially, it is NULL). Add a public static method that instantiates this class if this member is NULL (and sets this member's value) and then returns the value of this member. •Set the constructor's status to protected or private so that no one candirectly instantiate this class and bypass the static constructor mechanism.

The generic structure of Singleton design pattern is given in Figure B.10.1.

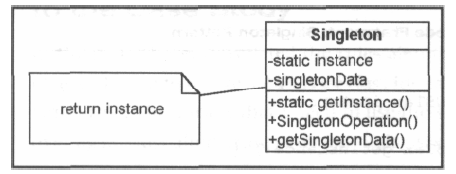


Figure B.10.1 Generic Structure of Singleton Pattern

The solution is designed as given in Figure B.10.2.

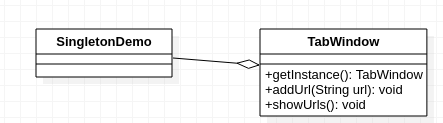


Figure B.9.2 Class Diagram Solution of Singleton Design Pattern Question

**REFERENCES**

1. [Alan Shalloway,](https://www.amazon.com/Alan-Shalloway/e/B001IXPWYW?ref=dbs_a_def_rwt_hsch_vu00_tkin_p1_i0)[James R. Trott](https://www.amazon.com/s?ie=UTF8&field-author=James+R.+Trott&text=James+R.+Trott&sort=relevancerank&search-alias=digital-text&ref=dbs_a_def_rwt_hsch_vu00_tkin_p1_i0), “Design Patterns Explained: A New Perspective on Object-Oriented Design (Software Patterns)”, Pearson Education, Second Edition, 2010.
2. Eric Freeman, Elisabeth Freeman, “Head First Design Patterns”, O’reilly Publications, October 2004, 1st Edition.
3. Satzinger, Jackson, Burd, “Object Oriented Analysis and Design with Unified Process”, Thomson Learning, 1st Indian Reprint 2007.
4. Herbert Schildt, “Java: The Complete Reference”, 9th Edition, McGraw Hill.
5. https://docs.staruml.io/