Book Vending Machine

By: Priyanka Rao(W1166161)
Venkata Ramana Madhuri Kambalapalli(W1166617)

Table of Contents

1. Project description

Our project is about developing a Smart Vending Machine that allows users to exchange used books for no cost. It is a desktop application built in java platform. We have two types of vending machines:

1. School Vending Machine which can be installed in schools or colleges. This machine has used books relevant to the place of installation. Students can use this machine to exchange books for no cost.
2. Airport Vending Machine can be installed at various airports. This machine has used/old books belonging to different category. This machine allows travellers to rent books for minimum amount. They can rent the books at the beginning of their travel at source airport and return the book at destination airport.

Each vending machine has the option to get vending card. Vending card can be used to perform transaction for example to pay the book rent at airport vending machine. We have administrator to manage vending machines. He performs two important tasks :

1. Vending machine management: He is responsible for setting up and deletion of vending machine. His decision is based on activities/transaction at each vending machine. He removes vending machine that has less activity.
2. Update inventory of Vending Machine: He can add or remove items from vending machine. He updates stock of each vending machine upon receiving notification about stock status.

1.1 Assumption

Each school vending machine allows user/students to exchange books where the number of books he can withdraw from the machine is equal to the number of books he adds to the machine.

1.2 Functional Requirement

School Vending machine has following functionality

Donate Book:

1. User first enters no of books he wants to donate. Based on the number textbox fields will pop up.
2. User enters book name and selects category(grad,undergrad or generic)
3. Books are added to the catalog of that machine.

Exchange Book:

1. To exchange books student first need to add/donate books as explained above. The added book count is passed to exchange view where drop down list equal to books added will pop up.
2. Student then need to select book from the dropdown list and then click on getbook button.

Airport Vending Machine has following functionality

Rent Book:

1. Select number of books to rent
2. Select book type from dropdown list
3. Click on rent button
4. Rent is calculated based on rent amount and number of books
5. Payment option window opens which has two option cash or vending card
6. Upon selection, particular window opens where rent amount is to be paid.

Payment Option:

1. If card option is selected, Card Login window pops up. User can login using card no and pin number, after authentication amount is deducted from card.
2. If cash option mode is selected , user inserts cash and then change amount will calculated and returned.

Both Vending machines provide options to get new vending card, view vending card balance and recharge vending card.

Get a Vending machine card:

1. select generate card option from main menu
2. create vending card page opens
3. Enter name, location, address, contact number and set pin number.
4. click on submit button
5. vending card number is generated and displayed to the user

View card balance and recharge:

1. Login using card number and pin number
2. After validation & authentication, balance is displayed
3. If user thinks balance is low he can click on recharge button.
4. window opens with denomination, select amount to be recharged and card is recharged.

1.3 Technologies used

Java, swings, frames and MYSQL for backend and Visual Paradigm for UML diagrams.

2. Design Methodology

1. **Object Oriented Principle**

   Following OO principle we addressed while developing our project:

1. Each class has single responsibility so that there is high cohesion and low coupling. example:

   Vending card: This class includes functionality to generate new card.

   Book : This class in one of the Item type. Each vending machine has book as an item. This class has getter and setter methods to set data members likes book name, book category, price.

   SchoolVM: School Vending machine class has functionalities like create vending machine,display items in vending machine etc.

2. Implementing to Interface rather than concrete class.

   example 1: Vending Machine has a Interface,which makes our code extensible and any new type of vending machine can be easily subclassed from this interface.

   example 2: We have abstracted the item each vending machines contains , so that our vending machine can easily support any new type of item in future. Current implementation support Book

3. Using composition over inheritance: We have given prominence to has-a relation over is-a relationship. So our Vending Machine class has item, vending card and VendingCard class has payment strategy.

2. Book Vending Machine UML Diagram

Consolidated UML diagram of Book Vending machine. This diagram depicts relationship between several classes used in the project.

Figure 1: UML diagram of Book Vending Machine

1. **Bridge Pattern:** It is a structural pattern. It is used to decouple abstraction from its actual implementation, so that both can vary independently.

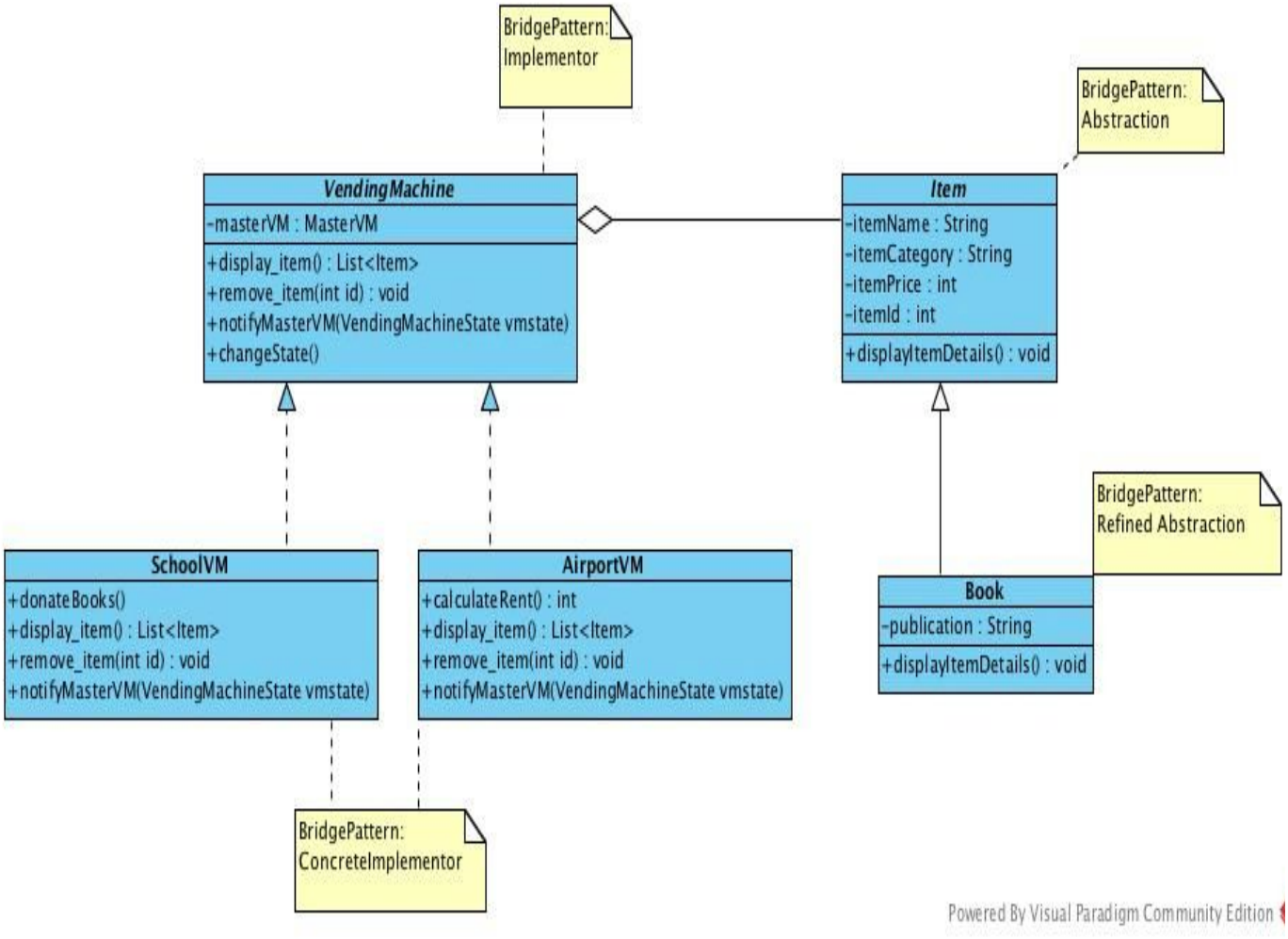

Figure 2: Bridge pattern class diagram

Usage: In our project we have used bridge pattern to make vending machine independent of its functionality and the items it holds. This pattern separates different kinds of item eg: book, stationery, etc from implementation aspects of vending machine like exchange of used items only, renting items, selling new items.

We have two implementations of vending machine:

1. SchooVM: allows students to exchange used books
2. AirportVM: allows passengers to rent book in the airport

To support this two different implementation we have an interface VendingMachine and both SchoolVM and AirportVm are subclassed from it. Each vending machine can have variety of items, so we have an interface Item, and current implementation support only book as one of the item type.

Advantage: This pattern makes our code flexible and extensible to support future requirement. for example: if we have another implementation of vending machine in future which sells multiple items like book & stationery; to support this we just need to subclass from vending machine class thereby preserving OCP(Open to extension Closed to Modification). All the implementation details are hidden from client.

2. **Factory Method:** It is a creational pattern. It defines an interface for creating objects and lets subclass deal with instantiation.

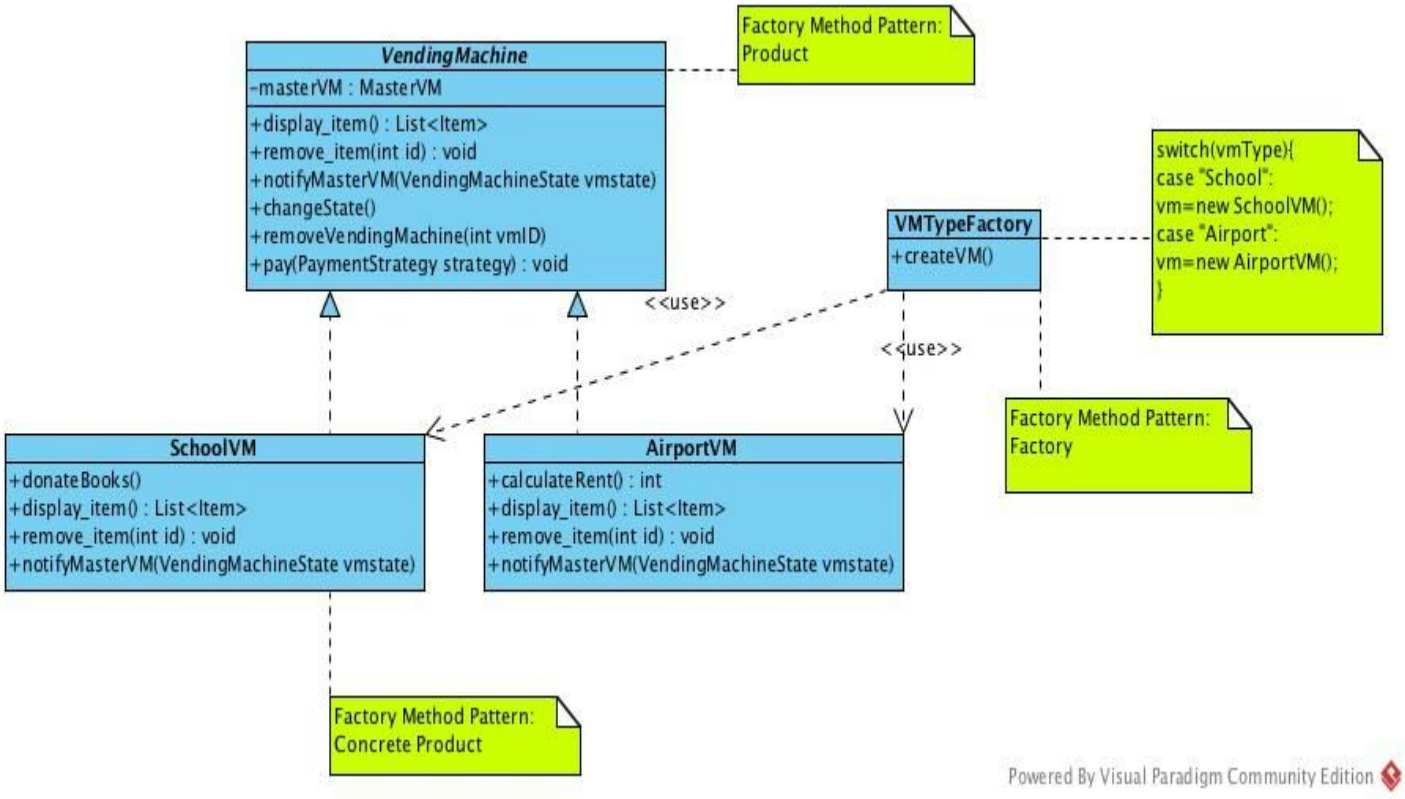

Figure 3: Factory method pattern class diagram

Usage: We have two types of vending machines School VM and Airport VM. So we create an Interface VmFactoryType class and let subclass to instantiate School VM and Airport VM and new Vending Machine type can be supported by just subclassing.

Advantage: Instead of distributing object creation in different classes we use Factory method to encapsulate object creation at one place. Easy to maintain and modify. Provides an application-independent object with an application-specific object to which it can delegate the creation of other application-specific objects.

3. **Observer Pattern:** It is a Behavioral Pattern. It is used to define one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically.

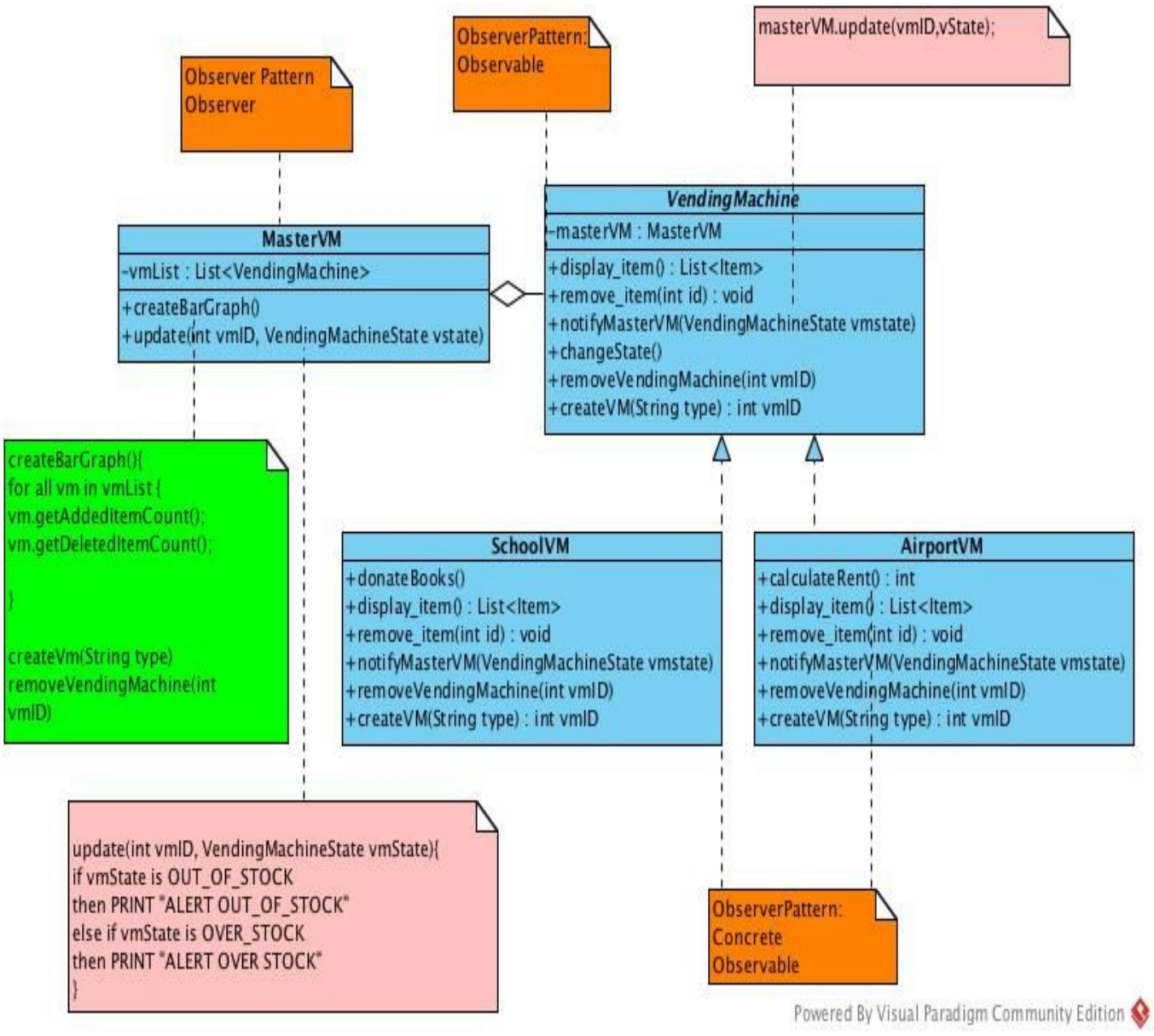

Figure 4: Observer Pattern class diagram

Usage: We have used both Observer Push model and Observer pull Model. In our project we have many to one dependency(many subjects - one observer)

Observer Push model: each subject(SchoolVM or Airport VM) first register themselves with the Observer (MasterVM). When user/client withdraws or donates book the subjects(SchoolVm or Airport VM) might go OUT_OF_STOCK state or OVERSTOCK state. In this state subject will notify the observer and observer will take appropriate actions like removing the extra books from Vending machine or Updating new Books to Vending Machine.

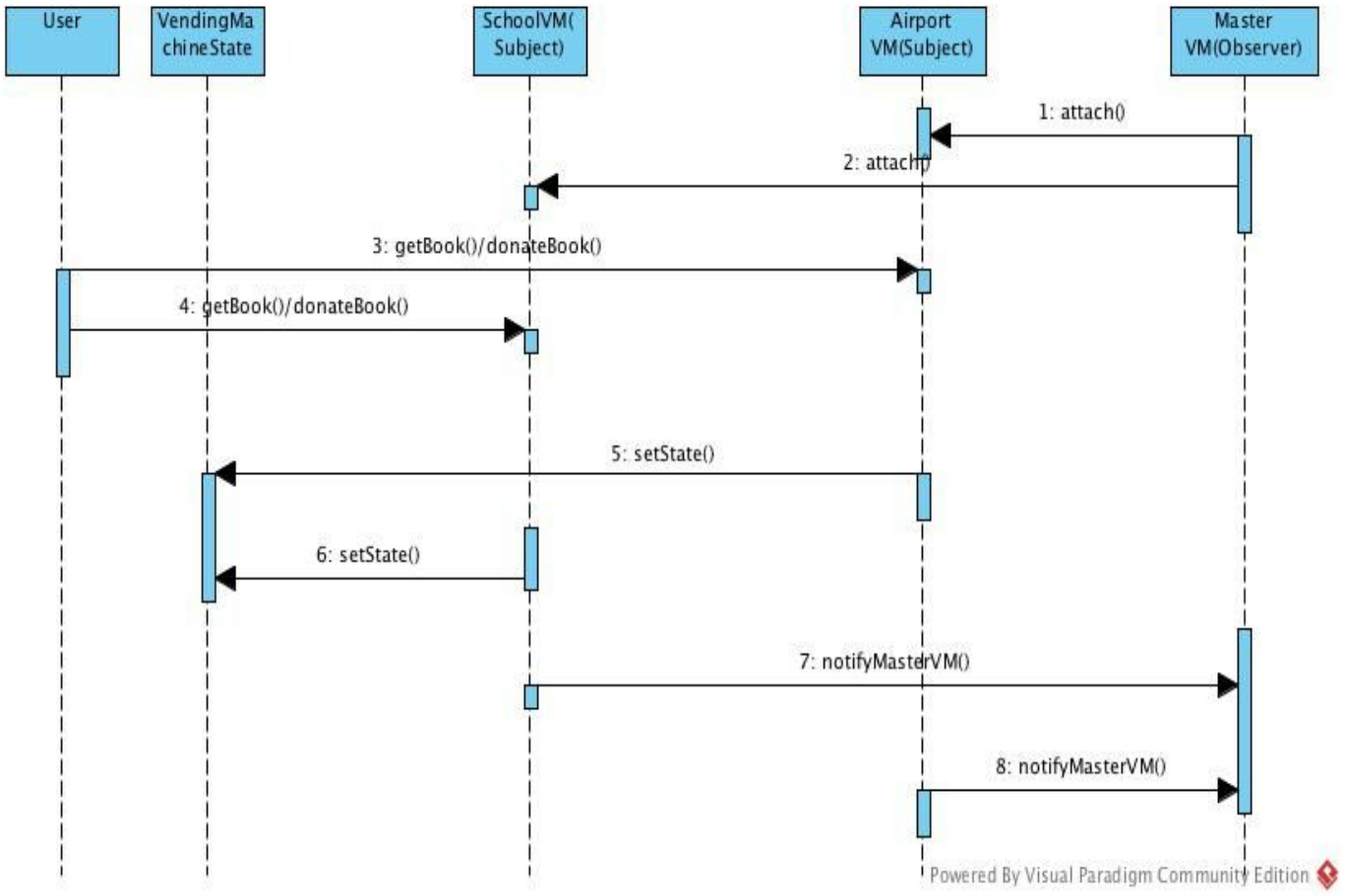

Figure 5: Observer push model sequence diagram

Observer Pull Model: MasterVM(Observer) periodically gets total number of books added and deleted from individual machine and this count is represented in the form of bar graph to the admin.This bar

graph depicts the transactions at each machine. Admin can make intelligent decision based on this graph. For example he can remove vending machine with low transactions.

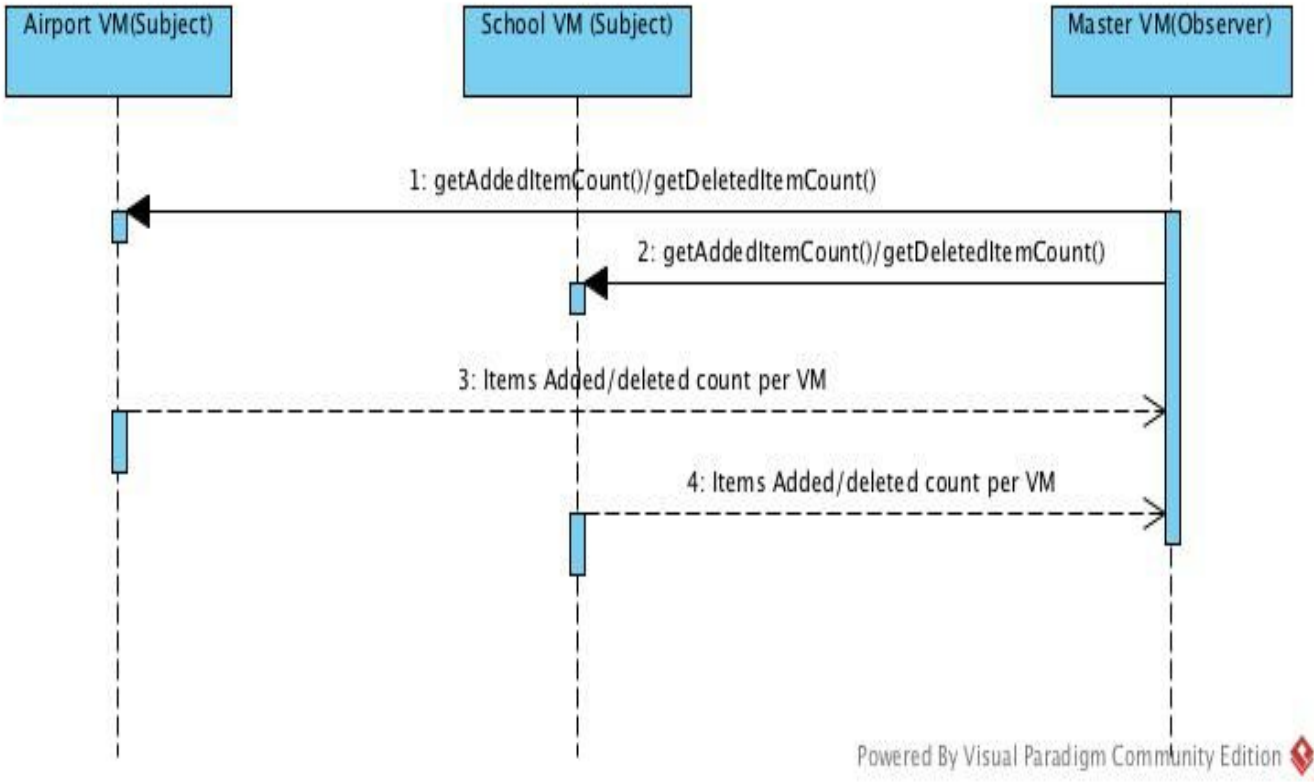

Figure 6: Observer Pull model sequence diagram

Advantage of using Observer: The Observer pattern lets you vary subjects and observers independently. It lets you add observers without modifying the subject or other observers.

4. **State Pattern:** It is a behavioral pattern. It allows an object to alter its behavior when its internal state changes. The object will appear to change its class. In State pattern, we create objects which represent various states and a context object whose behavior varies as its state object changes..

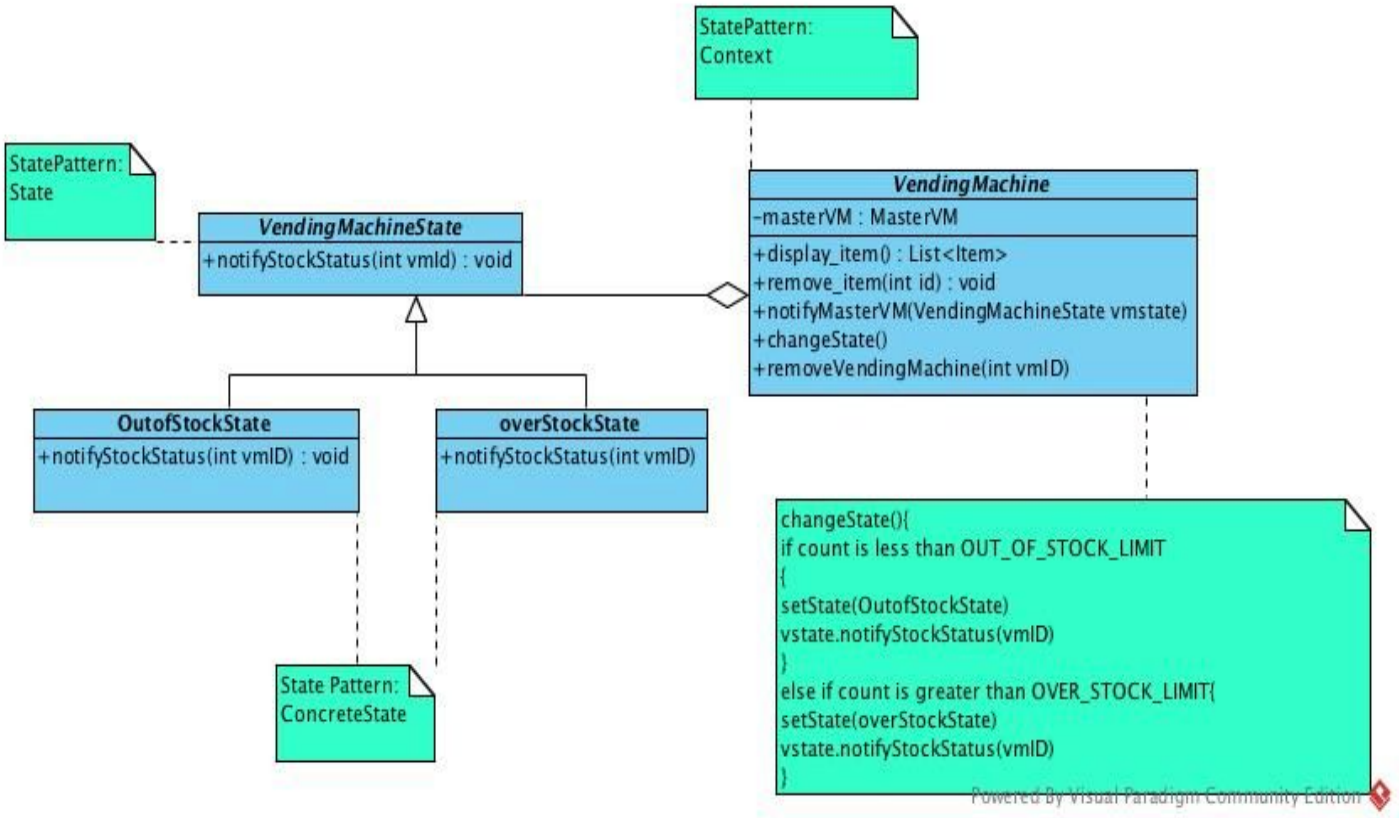

Figure 7: State pattern class diagram

Usage: Book Vending Machine has two states OutOfStockState and OverStockState. While renting or exchanging books, if book counts fall below specified limit i.e 2 in our application, vending machine will go to OutOfStockState. When user donates or returns books, if count of books on vending machine reaches above specific limit, vending machine will be in OverStockState

Advantages: Reduces duplication by eliminating repeated if/then or switch statements If State objects have no instance variables, they are essentially flyweights with no intrinsic state.

5. **Strategy Pattern:** Define a family of algorithms, encapsulate each one and make them interchangeable. Strategy lets the algorithm vary independently from the clients that use it.

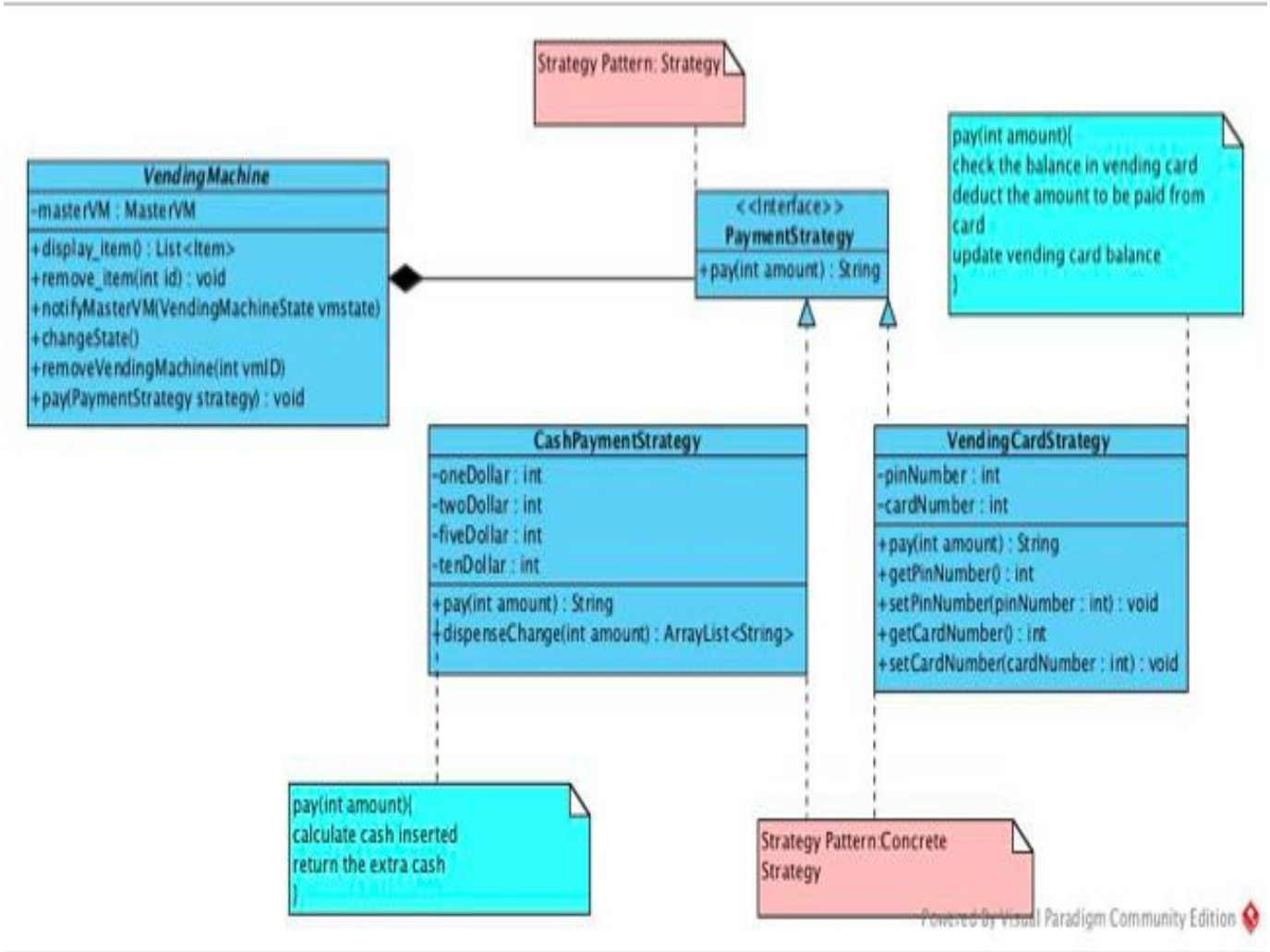

Figure 8: Strategy Pattern class diagram

Usage: When a user selects books for renting, total price to be paid will be calculated and displayed. User is given a choice to pick different modes of payment like cash or using vending card. Each payment method has different logic and can be interchangeable i.e. customer can use any strategy to make the payment. Cash payment module will take different denominations of cash as input from user and calculates the return change and gives it to user. Vending card mode of payment will take card number and pin number from user, validates card, checks balance, deducts rent and updates vending card balance.

Advantages: In future, new methods of payment like debit card or credit card can be added easily by extending the interface without violating OCP principle. So the application can use any strategy at later period of time to make payment for books rented.

6. **Chain of Responsibility:** creates a chain of receiver objects for a request. This pattern decouples sender and receiver of a request based on type of request. This pattern comes under behavioral patterns.

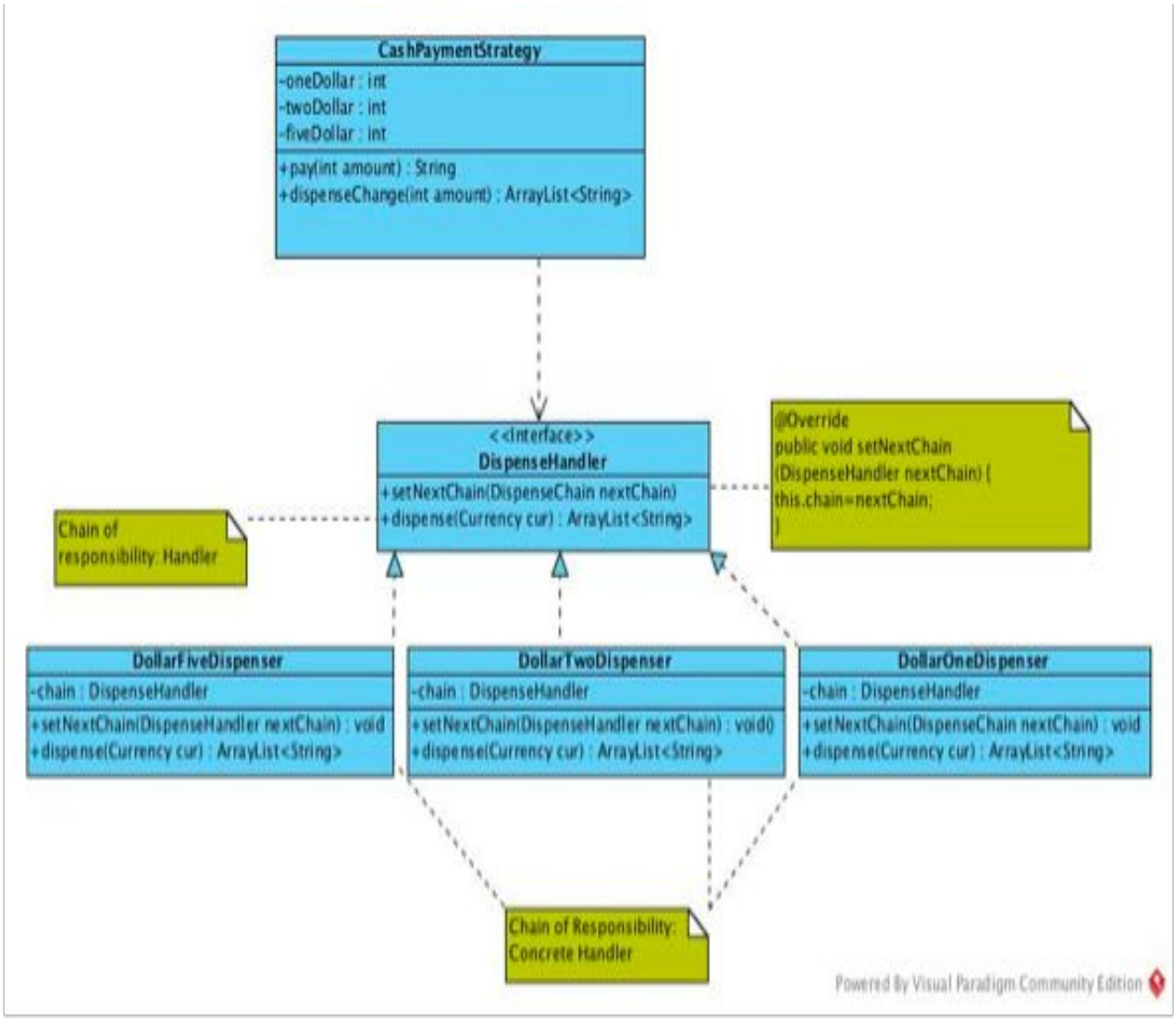

Figure 9: Chain of responsibility pattern class diagram

Usage: In cash payment module, when user inserts extra money, change has to be returned to user based on exact denomination. For example, if user has to pay $7 and inserted $20, $13 has to be returned in denominations of two notes of $5, one note of $2 and one note of $1. This problem of dispensing exact denominations is handled by different objects in chain. First object in the chain will divide amount by 5 and decides how many $5 notes to be dispensed. The remainder amount will be sent to next object in the chain to decide on count of $2 notes and so on.

Advantages: Chain of responsibility pattern for dispensing change or cash makes application flexible to add new code later for different denominations by just extending the handler. Without this pattern if new denominations logic has to be added, one has to open the code and modify it.

7. **Singleton Pattern:** is used to enforce that only one instance exists at all times.

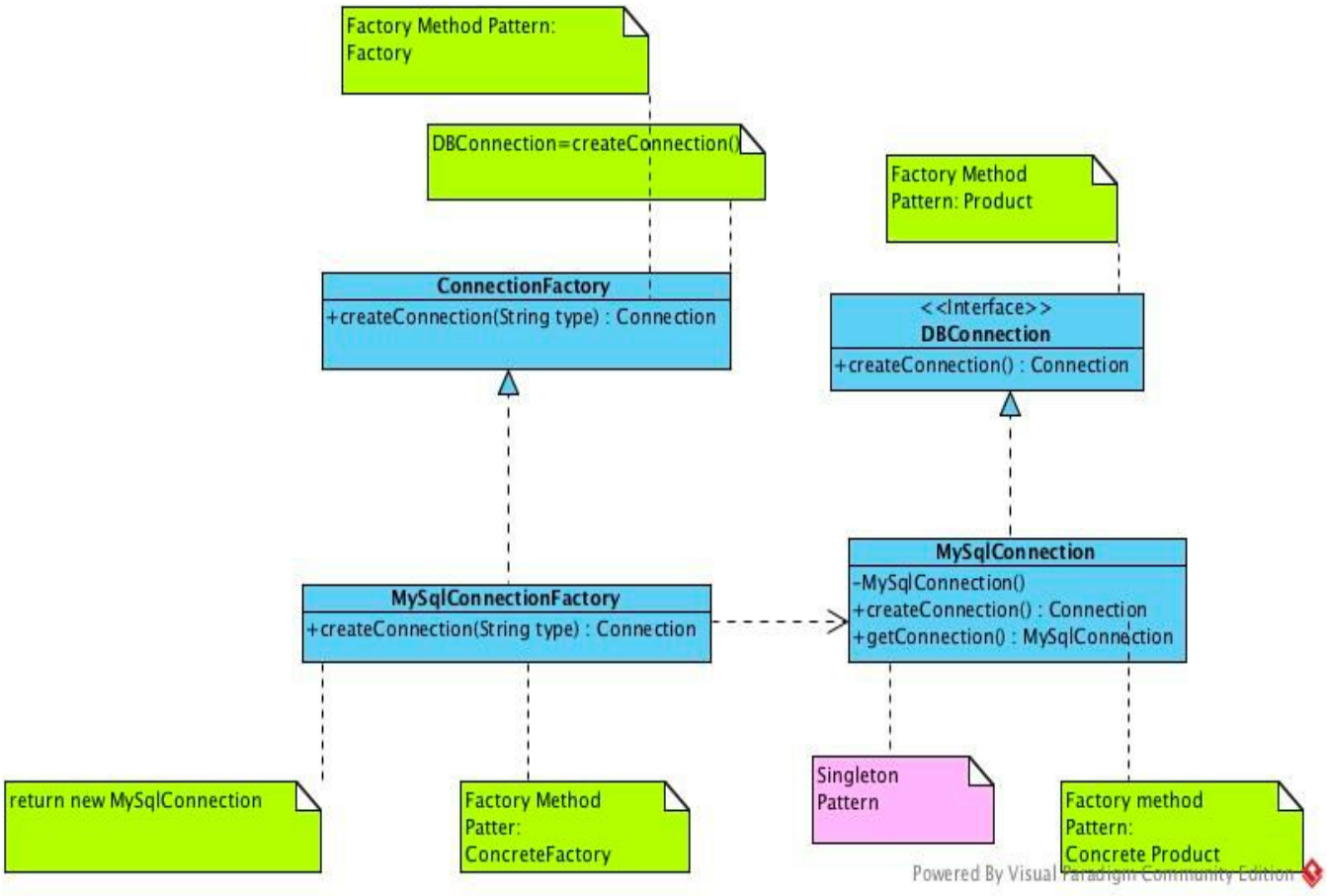

Figure 10: SIngleton Pattern class diagram

Usage: We used this pattern to set up database connection, so that only one database connection exists at all time. We created database singleton connection using factory method pattern to make application compatible for different jdbc connections. We

implemented application using MySql database and used MySqlConnectionFactory which is subclass of ConnectionFactory.

Advantages: New type of connections can be created by implementing Connection Factory class. MySqlConnection Singleton class ensures only one connection is created any time. This enforces consistent database transactions.

8. **Data Access Pattern:** The Data Accessor pattern decouples application logic from the physical data access implementation.The Data Accessor implementation handles all the physical data access details on behalf of the application code.

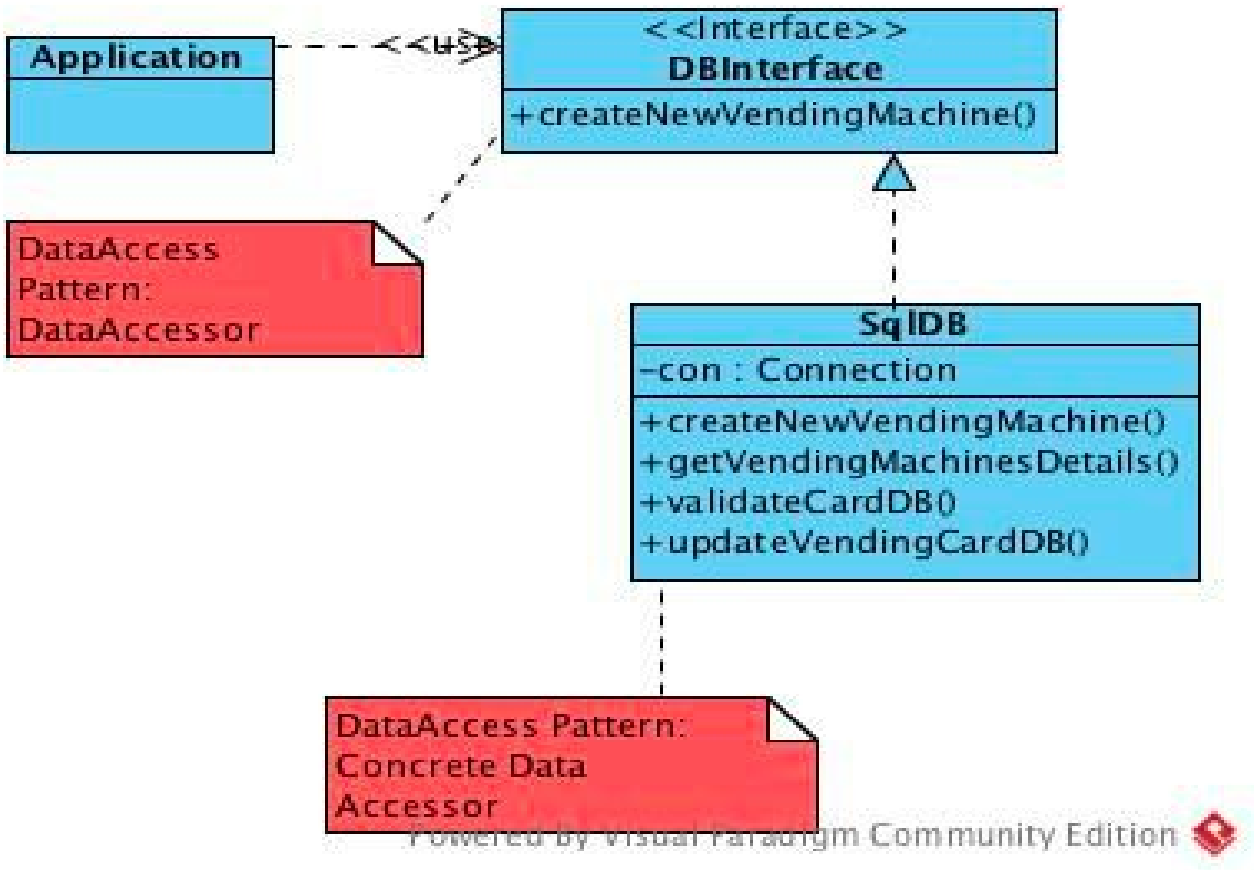

Figure 11: Data access pattern class diagram

Usage: Instead of sprinkling database access code throughout the application logic, we have separated database related code like update,delete, insert into a single file named SqlDB which implements DBInterface. This interface is the only single point of access to all database related operations.

Advantages: If application has to be moved to work on different database structures like file, xml etc, new code can be written by extending this DBInterface without affecting existing application. In this way we built an abstraction that hides low-level data access details from the rest of the application code.

9. MVC Pattern

MVC Pattern stands for Model-View-Controller Pattern. This pattern is used to separate application's concerns.

- **Model** - Model represents an object carrying data. It can also have logic to update controller if its data changes.
- **View** - View represents the visualization of the data that model contains.
- **Controller** - Controller acts on both model and view. It controls the data flow into model object and updates the view whenever data changes. It keeps view and model separate.

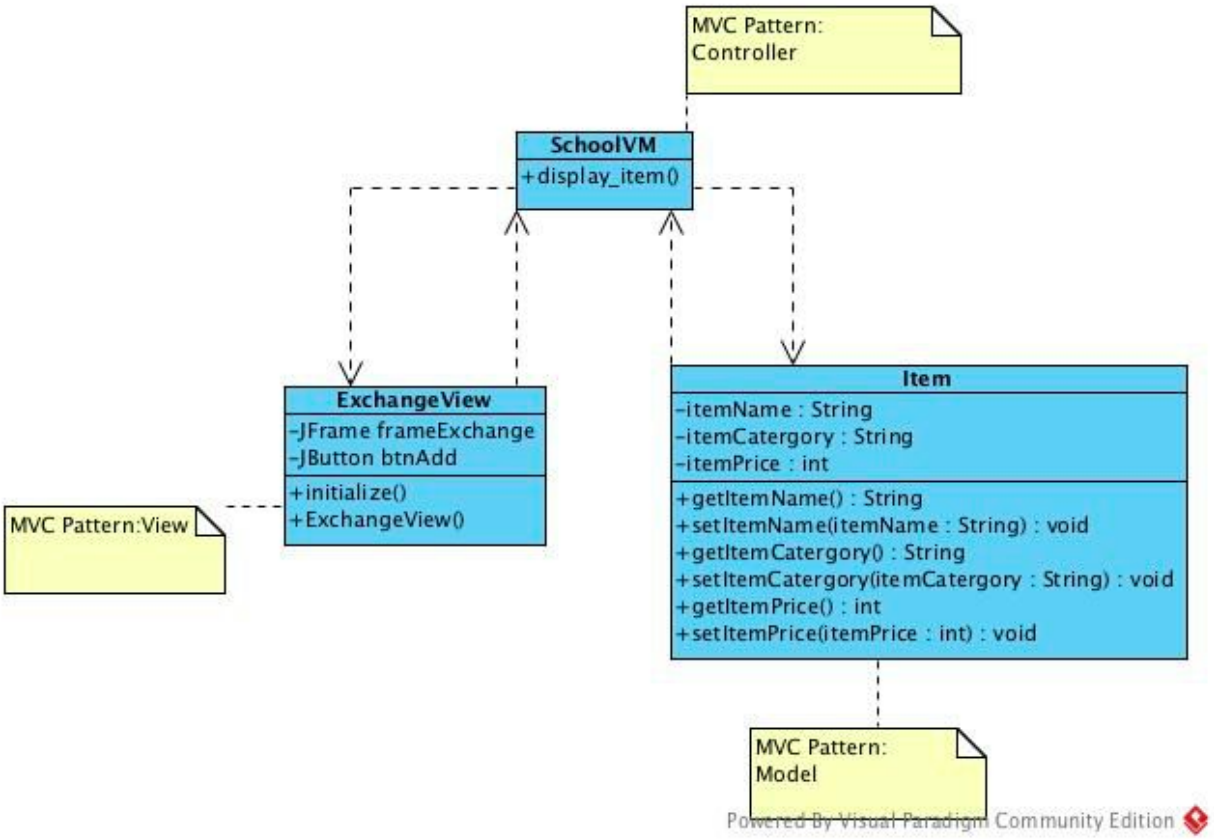

Figure 12: MVC Pattern class diagram

Usage: Book Vending Machine project code is divided into three modules, View package has all classes related to Java Swing UI view, Controller package has classes that controls data and view and data models are available in Model package. Above diagram describes one case of model view control where exchange view interacts with SchoolVM which is a controller and gets Item information and displays on the view.

Advantages: Whenever new code has to be added to the project, one can easily understand which classes can be used and where to insert. Swing View can be replaced with different kinds of views like jsp or html easily as all logic to get or display information is handled by controller.

3. Database tables

MasterVM Table

This table has list of vending machine created. Each row in table represent one vending machine. The type field identifies vending machine type. Current implementation has only two types School and Airport.

idVM	Location	Type	Name
12	Santa Clara	School	SCU
13	San jose	Airport	SJC

VENDINGMACHINE12

Name do the table is concatenation of Vending Machine and Id of vending machine generated when its created.This table keeps list of all items in the vending machine. Current implementation has one item Book. When book is drawn from vending machine its flag is set to delete.

idBook	flag	title	category	publication	price
1	add	Design patterns	Graduate	Addison Wesley	10$
2	delete	JAVA	Graduate	O'reilly	5$

VENDINGCARD

Vending card is used to perform transaction at vending machine. Each user has a vending card. This table keeps data relating to vending card.

idVendingCard	name	address	contactNo	pin	amount
95050001	John	Santa Clara	22222222	1234	10$

4. Future Scope

1. **Mediator Pattern**: Defines an object that encapsulates how a set of objects interact. Mediator promotes loose coupling by keeping objects from referring to each other explicitly, and it lets you vary their interaction independently.

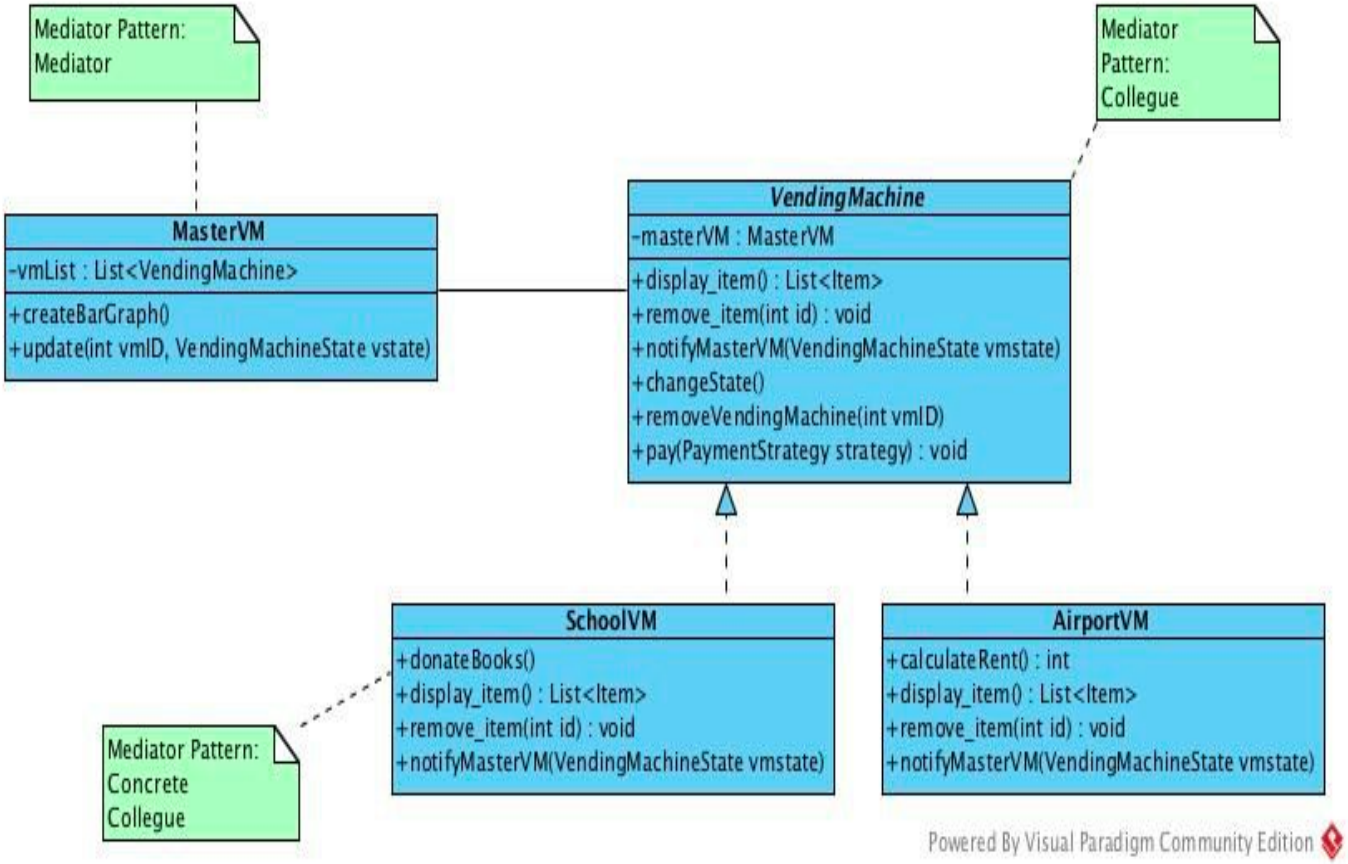

Figure: 12 Mediator Pattern class diagram

Usage: Vending machine when unable to service particular request of client , it contacts mediator (MASTERVM) about its state and MASTERVM will coordinate with other VENDING MACHINES to know which one is able to serve the request, that vending machine id is passed to first Vending machine which can now assist the client by

telling next nearest vending machine that is able to service the request.This will be very helpful in Airport VM. example when client finds no book of his interest at the vending machine , vending machine using mediator pattern locates the nearest machine which has this book , so the client can now go to that machine.

Advantage: only one class(MasterVM) needs to be aware of other vending machine and hence only this class can be modified to add new vending machine.

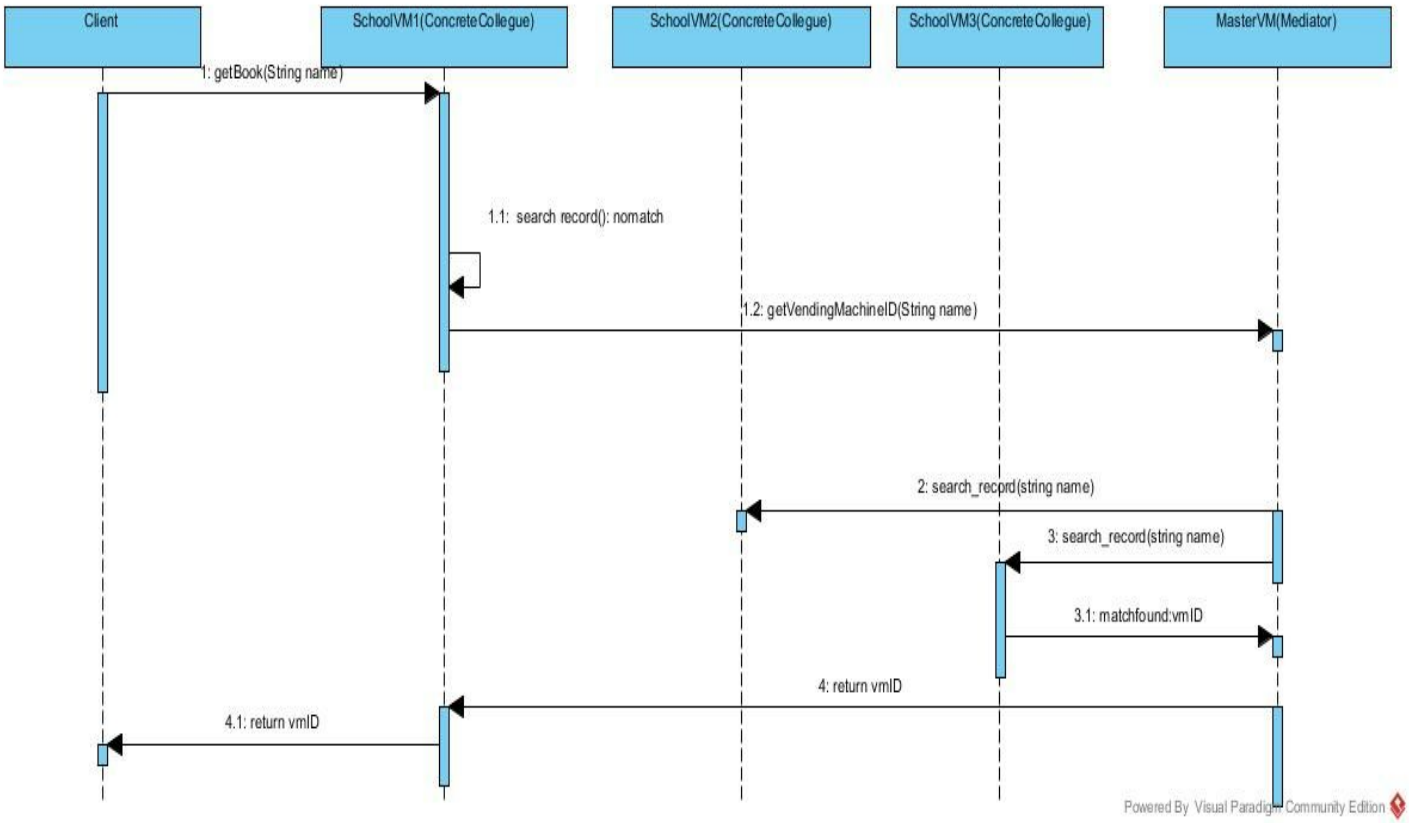

Figure: 13 Mediator Pattern Sequence diagram

2. **Strategy Pattern :** can be used with mediator pattern above to find the next nearest vending machine. We can use several algorithm/ strategy to find next nearest vending machine.