|  |  |
| --- | --- |
| **SOFTWARE DESIGN PATTERNS**  Diploma in IT  Year 2 (2024/25) Semester 4 | Week **4** |
| **2** hours |
| **Practical 4 – Observer / Iterator** | |

**OBJECTIVES**

* Implement the Observer and Iterator design patterns

**ACTIVITIES**

**Activity 1**

Recall Tutorial 4 Activity 1:

The class CompanyStock contains the following attributes:

* name: string // Name of the stock
* currentPrice: double // Current price of the stock

The class Customer contains the following attributes:

* name: string // Name of the customer

A customer can own many stocks, and a stock can be owned by many customers. Whenever the current price of the stock changes, all the customers owning this stock must be notified.

Implement the Observer design pattern such that running the following program will produce the desired output.

Program:

CompanyStock apple = new CompanyStock("Apple", 233.0);

CompanyStock tesla = new CompanyStock("Tesla", 261.63);

CompanyStock singtel = new CompanyStock("SingTel", 3.23);

Customer john = new Customer("John");

john.addStock(apple);

john.addStock(tesla);

Customer mary = new Customer("Mary");

mary.addStock(apple);

mary.addStock(singtel);

apple.CurrentPrice = 225.5;

tesla.CurrentPrice = 264.2;

john.removeStock(apple);

apple.CurrentPrice = 255.1;

Output:

John is notified that Apple price is set to $225.50.

Mary is notified that Apple price is set to $225.50.

John is notified that Tesla price is set to $264.20.

Mary is notified that Apple price is set to $255.10.

**Activity 2**

The following (incomplete) code defines a DinerMenu class that contains a list of MenuItem objects stored in an array.

public class MenuItem

{

private string name;

private float price;

// other implementations, e.g., constructor

}

public class DinerMenu

{

int numberOfItems = 0;

MenuItem[] menuItems = new MenuItem[20];

public int NumberOfItems { get { return numberOfItems; } }

public PriceIterator createIterator(float maxPrice)

{

return new PriceIterator(this, maxPrice);

}

public void addItem(MenuItem item)

{

menuItems[numberOfItems++] = item;

}

public MenuItem getItem(int position) { return menuItems[position]; }

// other implementations, e.g., constructor, addItem(), getItem() etc.

}

Write the C# code for the PriceIterator class, which is an iterator that traverses all items in the menu whose price is at most the given maxPrice parameter. Running the following program should produce the desired output.

Program:

DinerMenu mcdodo = new DinerMenu();

mcdodo.addItem(new MenuItem("Hamburger", 2.00f));

mcdodo.addItem(new MenuItem("McSpicy", 3.50f));

mcdodo.addItem(new MenuItem("Fries", 1.50f));

PriceIterator iter = mcdodo.createIterator(2.50f);

while (iter.hasNext())

{

MenuItem item = (MenuItem) iter.next();

Console.WriteLine($"{item.Name,12} ${item.Price:N2}");

}

Output:

Hamburger $2.00

Fries $1.50