|  |  |
| --- | --- |
| **SOFTWARE DESIGN PATTERNS**  Diploma in IT  Year 2 (2024/25) Semester 4 | Week **4** |
| **1** hour |
| **Tutorial 4 – Observer / Iterator** | |

**OBJECTIVES**

* Understand the Observer and Iterator design patterns

**ACTIVITIES**

**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.

1. Draw the class diagram for the above scenario using the Observer design pattern

A diagram of a company

Description automatically generated

1. Draw a sequence diagram to illustrate the function setCurrentPrice() in the class CompanyStock

A diagram of a diagram

Description automatically generated

**Activity 2**

In a video game, each Shop in the game has a collection of Items. Characters can visit shops to purchase items, but not all characters can use all items (for example, a barbarian character may not be able to use a magic wand due to their class, or a Level 5 character may not be able to use a Level 10 item as their level is too low).

The Item class has a name attribute, and the Shop class has a method createUsableItemIterator(Character) that returns an iterator for all items in the shop that the given character can use. Write C# code that lists all the usable items for a Character in a Shop:

Shop has many items

Character many to many?

Character character = new Character();

Shop shop = new Shop();

// Write your code here