|  |  |
| --- | --- |
| **SOFTWARE DESIGN PATTERNS**  Diploma in IT  Year 2 (2024/25) Semester 4 | Week **6** |
| **1** hour |
| **Tutorial 6 – Factory Method / Abstract Factory** | |

**OBJECTIVES**

* Understand the Factory Method and Abstract Factory design patterns

**ACTIVITIES**

**Activity 1**

A video role-playing game has Quests, which require the player to kill a number of creatures of a particular type within the time limit (in number of days) to get the reward (in number of gold pieces). The game has different areas, and each area generates different Quests based on factors like the kind of monsters inhabiting the area or the difficulty level of the area.

The game currently has two areas called Rivendell and Mordor. Each area generates different quests depending on whether the player is in a village or a city. The following table shows the quests generated:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Area** | **Location** | **Monster** | **Number** | **Time Limit (days)** | **Reward (GP)** |
| Rivendell | Village | Pixie | 3 | 4 | 8 |
| City | Thief | 2 | 6 | 15 |
| Mordor | Village | Giant Rat | 6 | 5 | 10 |
| City | Bandit | 4 | 7 | 20 |

The following are a set of classes for the game.

A black and white rectangular sign with black text

Description automatically generatedA black and white rectangle with black text

Description automatically generatedA black and white text

Description automatically generatedA black and white rectangle with black text

Description automatically generatedA black and white rectangle with black text

Description automatically generatedA black and white rectangle with black text

Description automatically generatedA black and white text

Description automatically generatedA black and white rectangle with black text

Description automatically generatedA black and white box with text

Description automatically generated

1. Arrange the elements into a class diagram that uses the Factory Method design pattern and show their associations (including inheritances).

A diagram of a server

Description automatically generated

1. Identify the classes that correspond to the Creator, ConcreteCreator, Product and ConcreteProduct classes in the Factory Method design pattern.

**Activity 2**

In the new version of the game, the quest generation process was updated. The number of monsters and the time limit is now determined by the area, although each location still generates a different monster. Also, each monster is given a difficulty level, and the reward is calculated using the following formula:

reward = difficulty \* number – time\_limit

The following table summarizes the values for each area:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Area** | **Location** | **Monster** | **Difficulty** | **Number** | **Time Limit (days)** |
| Rivendell | Village | Pixie | 3 | 3 | 5 |
| City | Thief | 8 |
| Mordor | Village | Giant Rat | 2 | 5 | 3 |
| City | Bandit | 5 |

The following are a set of classes for the new version of the game.

A black and white rectangular sign with black text

Description automatically generated A black rectangle with black text

Description automatically generated A black and white rectangle with black text

Description automatically generated A close up of a sign

Description automatically generated A black and white rectangle with black text

Description automatically generated A white rectangular sign with black text

Description automatically generated A black and white rectangular sign

Description automatically generated A white box with black text

Description automatically generatedA white box with black text

Description automatically generatedA black and white text

Description automatically generated A black and white text

Description automatically generated

1. Arrange the elements into a class diagram that uses the Abstract Factory design pattern and show their associations (including inheritances).
2. Identify the classes that correspond to the AbstractFactory, ConcreteFactory, AbstractProduct, ConcreteProduct and Client classes in the Abstract Factory design pattern.