|  |  |
| --- | --- |
| **Øvelse** | GameWorldV1 |
| **Projekt** | GameWorldV1 |
| **Formål** | Benytte *Factory Method* design pattern til at forbedre strukturen af en fungerende applikation. |
| **Beskrivelse** | Den givne Visual Studio solution rummer fra starten følgende klasser, der skal repræsentere en simulering af en simpel spil-verden:   * **Fighter**: Repræsenterer en spiller, som kan kæmpe mod forskel­lige modstandere. * **IOpponent + flere klasser som implementerer dette interface (i Opponents.cs)** : Repræsenterer de modstandere man kan møde i spillet. Bemærk, at alle klasser implementerer **IOpponent**. * **World**: Selve spil-verdenen, som har meget begrænset funktion­alitet. Den centrale metode er **GenerateEncounter**. * **IOpponentFactory**: Et interface for en ”opponent factory”, som implementeres af den ufærdige klasse **OpponentFactory­Stand­ard**, der ikke er taget i brug endnu. * **Program.cs**: Test af **Generate­Encounter­-**metoden. |
| **Trin** | 1. Download og unzip (VIGTIGT) filen **GameWorldV1.zip** fra Moodle. 2. Åbn **GameWorldV1** i Visual Studio. Den vigtigste del af koden er metoden **GenerateEncounter** i **World**-klassen. En stor del af koden omhandler logik­ken for at udvælge en passende modstander til spilleren. Denne udvælgelse afhænger af to faktorer: spillerens ”erfaring” (defineret af property **Experi­ence** i **Fighter**-klassen), samt om det p.t. er dag eller nat i spillets verden (defineret af property **DayOrNight** i **World**-klassen). Sørg for at du forstår hvordan denne logik er opbygget, før du fortsætter (NB: Det er ikke nødvendigt at sætte sig grundigt ind i hvordan modstander-klasserne er opbygget; det vigtige er, at de alle implementerer **IOpponent**). 3. Den nuværende opbygning er ret ufleksibel, hvis man ønsker at kunne vari­ere udvælgelsen af modstandere. Vi vil derfor gerne udskille denne logik i en factory-klasse, specifikt klassen **OpponentFactoryStandard**. **Din opgave er at implementere denne ændring.** Det vil involvere disse trin:    1. Fjern den del af koden som implementerer logikken til udvælgelse af en modstander fra **GenerateEncounter**, og flyt den i stedet til metoden **Create** i **OpponentFactoryStandard**.    2. Tilføj en ny parameter til **GenerateEncounter**, af typen **IOpponent­Fac­tory**.    3. Brug den nye parameter til at sætte den lokale variabel **theOppo­nent** til det som et kald af **Create** returnerer (husk at **Create** jo skal have både spillerens erfaring samt dag/nat-tilstanden med som argumenter, da de jo stadig bruges til at udvælge en modstander).    4. Ret koden i **Program.cs**, sådan at kaldet af **GenerateEncounter** nu får et argument af typen **OpponentFactoryStandard**. 4. Du burde nu kunne køre programmet igen, og få det samme resultat som før. Husk på, at denne øvelse ikke går ud på at få programmet til at kunne noget nyt, men om at forbedre dets struktur. 5. Hvis du har tid tilovers, kan du prøve at definere endnu en klasse som implementerer **IOpponentFactory**, og i den definere en anden logik til udvælgelse af modstandere. Prøv også at bruge denne nye klasse som argument til kaldet af **GenerateEncounter** i **Program.cs**. |