Mittuniversitetet
MID SWEDEN UNIVERSITY

Lab 4 – DP AbstractFactory

Designmönster med C++

Syfte: Tillämpning av DP AbstractFactory (och tillfredsställande av akut spelbehov).

Lab 4 - Game och Abstract Factory

I den här laborationen ska vi syssla med en kommande storsäljare på spelmarkanden... Ett textbaserat spel sätter spelaren på prov genom att presentera olika hinder som ska passeras.

Klassen `Obstacle` är en abstrakt basklass för konkreta deriverade klasser som representerar olika hinder. För varje hinder ska spelaren välja mellan ett antal olika alternativ, actions.

Den abstrakta basklassen `Action` definerar interfacet för ett antal deriverade konkreta Action-klasser. För varje Obstacle-objekt är ett av de möjliga Action-objektet det som gör att spelaren kan passera hindret. Under spelet förbrukar spelaren resurser, ammunition och sin hälsa och kan tvingas ge upp när någon resurs är slut. Om hälsoresursen når noll-läge dör spelaren.

Klassen `Player` är en basklass för två spelartyper, `OrdinaryPlayer` och `Heroplayer`. Man kan välja mellan två varianter på spelet, *'Nice game'* eller *'Nasty Game'*. Varje spelvariant bygger på en viss spelartyp, en viss uppsättning hinder och tillhörande uppsättning actions.

Uppgift

Du ska tillämpa designmönstret *Abstract Factory* för att instansiera alla de objekt som hör till en viss typ av spel. För varje variant av spelet ska ett motsvarande factory-objekt användas för att skapa de ingående objekten på ett kontrollerat sätt.

`GameFactory` är en abstrakt klass som ska implementeras av klasserna `NiceGameFactory` och `NastyGameFactory` (se GameFactory.h/cpp).

En spelomgång representeras av ett `Game`-objekt (se Game.h/cpp). Konstruktorn för Game ska använda sin GameFactory-parameter för att skapa rätt uppsättning objekt för datamedlemmarna `obstacles`, `actions`, `player` och `title`:

I testprogrammet (testLab4.cpp) väljs vilken variant man ska spela och motsvarande factory-objekt används:

```
GameFactory *gf = nullptr;
cout << "Choose your game:" << endl;
cout << "1 - Nice game" << endl;
cout << "2 - Nasty game" << endl;
cout << "3 - quit" << endl;
...
```

```
switch (choice) {
case 1:  gf = Ett factory - objekt; break;
case 2:  gf = Ett annat factory - objekt; break;
case 3: keepOn = false;
}
...
```

Den bifogade källkoden (Studentkod.zip) ska kompletteras med kod där pseudo-kod och instruktioner finns, det gäller filerna GameFactory.h/cpp, Game.cpp och testLab4.cpp. Resultatet ska naturligvis bli ett fungerande spel med två spelvarianter som kan spelas upprepade gånger.

Inga ändringar av befintlig kod är tillåten. Däremot får du gärna *utvidga* spelet genom att derivera nya Obstacle-klasser / Action-klasser. Som vanligt gäller att allt dynamiskt minne ska återlämnas före avslut.

Redovisning

Zippa ihop den kompletta lösningen och skicka in.