Java text adventure - Zuul

Tot nu heb je vooral werkende prototypes gemaakt. Je code 'doet het'. Dat je code 'het doet' is weliswaar een noodzakelijke, maar niet afdoende voorwaarde.

Dit project gaat over 'Software Engineering'; het schrijven van goed onderhoudbare, gedocumenteerde, aan versiebeheer onderhevige, efficiente, leesbare code. "Loose coupling, high cohesion" (zie onderaan deze tekst).

Doel

- Object Oriented Programming (OOP)
- C#
  - 'Moderner' dan Java
  - Ook gebruikt in Unity3D
  - 'C++ with sidewheels' (voorbereiding op C++/OpenGL)

Onderwerpen

- UML/OOP (Generalization / Composition / Aggregation)
- cohesion / coupling
- refactoring
- documentatie / doxygen
- versiebeheer / git
- debugging / gdb
- public static void main(String[] args)
- static / const final / abstract / overloading / interfaces / polymorphism

Software

Google, download en installeer:

- Of: Visual Studio/MonoDevelop/… (C#) <-- aanbevolen
- Of: Eclipse/Netbeans/… (Java) <-- mag eventueel ook
- Of: Visual Studio with C++/GCC/… (C++) <-- voor nu afgeraden
- yEd (voor het tekenen van klassediagrammen)
- doxygen (voor het genereren van html-documentatie)
- git (versiebeheer)

Aangeleverd

- textadventure.md (deze file)
- opdrachten.md
- basic source code zuul (zuul-java, zuul-cs, zuul-cpp)

Opdracht

- Integreer/verwerk op creatieve wijze bijgeleverde opdrachten (en inhoud van de workshops) in jouw textadventure.
- Houd op github (compileerbare!) versies bij van jouw textadventure.

> Let op!! Breidt de code uit door nieuwe classes te schrijven (Player, Item, Inventory, Key, Medkit, Enemy etcetera). Het is *NIET* de bedoeling om de code 'uit te breiden' door hele verhalen te schrijven verdeeld over tientallen Rooms.

Opleverspecificaties

- GDD
  - kaart van je 'level'
  - volledige gameplay
- Technisch ontwerp
  - klassediagram
  - overige technische documentatie (github link etc.)
- Doxygen documentatie in html (inclusief method parameters en return values)
- compileerbare source code van (en op) github

- de sourcecode is SCHOON en bevat dus GEEN executables/bytecode/object files.
- evaluatieverslag ('evidence'): geef voorbeelden van je code die een hoge mate van cohesion en loose coupling vertoont, waarom dat zo is, waar dat beter kan en hoe dat zou moeten.

Conceptpresentatie

- GDD + Technisch ontwerp (presenteer klassediagram)
- beoogde objectives + gameplay

Eindpresentatie

- kaart van je 'level' (uit het GDD)
- objectives + gameplay
- goed voorbereide live demo
- demo documentatie (genereer ter plekke)
- demo git/github (git log)

Betekenis 'coupling' en 'cohesion'

Coupling

```
The term 'coupling' describes the 'interconnectedness' of classes.
We strive for loose coupling in a system -- that is, a system where
each class is largely independent and communicates with other
classes via a small well-defined interface.
```

Cohesion

```
The term 'cohesion' describes how well a unit of code maps to a
logical task or entity. In a highly cohesive system, each unit of
code (method, class, or module) is responsible for a well-defined
task or entity. Good class design exhibits high degree of cohesion.
```