


Titel :	Kurs :	
Deluppgift 3: Klassdiagram och C++ kod	DA337A C++ med objektorienterad design och analys)	
Skapad av :	Enhet :	
Jonas Wahlfrid	Spel	
Granskad av :	Ämnesområde :	
	Datavetenskap	
Godkänd av :	Version :	
	Issue 1 Draft B	
Filnamn :	Datum :	
Uppgift3KlassdiagramIssue1DraftB.doc	2015-10-20	

Dokumenthistoria

Version :	Datum :	Skapad av :	Granskad av:	Godkänd av:	Kommentar :
Issue 1 Draft A	2015-10-16	Jonas Wahlfrid			
Issue 1 Draft B	2015-10-20	Jonas Wahlfrid			VG krav

1. Inledning

Denna deluppgift är den tredje av flera som har som mål att skapa en design modell av ett fyra i rad spel. Uppgiften löses av en grupp bestående av två studenter.

2. Mål

Målet med uppgiften är att ge övning i objekt orienterad design.

3. Förberedelser

Innan denna uppgift påbörjas:

1. Läs igenom detta dokument.
2. Gör klart deluppgift 1 och få ett godkännande på ert klassdiagram.

4. Deluppgiften

Målet med deluppgiften är att skapa ett klassdiagram i Visual-paradigm baserat på klassdiagrammet från deluppgift 1 samt C++ kod. Klassdiagrammet ska visa relationerna, multiplicitet och helst även associations namn och/eller roll namn. Ni väljer själva om ni vill göra forward eller reverse engineering för att skapa klassdiagrammet. ~~De grupper som önskar väl godkänt på hela designuppgiften, ska efter att designen är klar med sekvensdiagram, implementera ett fungerande fyra i rad spel med stöd för människa mot människa.~~ Kravet på VG kommer att definieras i nästa deluppgift.

C++ koden ska innehålla attribut, metoder med parametrar och retur typer. Koden ska vara stubbad så att den kompilerar, men metoderna behöver inte implementeras. Paketet General, GUI och Network eller liknande namn ska realiseras. Paket i C++ implementeras med namespace. Klassernas ansvarsområde från CRC korten ska dokumenteras i h filerna i form av kommentarer. Metoderna ska kommenteras i h filerna enligt <https://docs.unrealengine.com/latest/INT/Programming/Development/CodingStandard/index.html#exampleformatting>

Metoder som ska ärvas måste märkas med virtual. Player klassen bör vara en abstrakt bas klass.

Om arv används i C++ och pekare inte används blir det problem med "object slicing" se

<http://stackoverflow.com/questions/274626/what-is-object-slicing>

För att undvika "object slicing" ska pekare användas. För att göra det omöjligt att kopiera objekt ska copy constructor och tilldelnings operator deaktiveras, genom att deklarera dessa privat i h filen och utan implementering i cpp filen i alla klasser.

private:

```
//Disable compiler-generated copy-assignment operator
Controller(const Controller& that);
Controller& operator = (const Controller& that);
```

5. Godkännande

Uppgifternas presenteras för handledaren vid nästa handlednings tillfälle.
För godkännande krävs:

Klassdiagram

1. Klassdiagrammet ska visa relationerna, multiplicitet och helst även associations namn och/eller roll namn.
2. Koden ska uppfylla kraven med kommentarer och kunna kompileras mm.
3. Object slicing problemet ska kunna förklaras.

Den enskilde studenten ska kunna redogöra för alla detaljer. När uppgiften är godkänd av handledaren ska koden zippas och klassdiagrammet lämnas i bild eller pdf format (**inte Visual-paradigm format**) på its learning.