# Duggainformation

Duggan har två syften:

- Duggan tjänar som inträdesbiljett att få påbörja projektet. Duggan är den tekniska intervjun i rekryteringen av projektgruppsmedlemmar.
- Duggan tjänar som individuell examination av kursmål för betyg 3.

**Duggan har en uppgift och bedöms G eller U.** Vi kräver tydlig, lättfattlig och underhållsbar kod, men vi kommer *inte* vara lika petiga som i labserien. En godkänd lösning ska fungera korrekt, lösningen använder lämplig klasshierarki, den saknar minnesläckor, den undviker koduppprepning, och lösningen har någorlunda namngivning och indentering.

## Duggatid

Du har 120 minuter på dig att lösa duggan. Du med rätt till förlängd skrivtid har 160 minuters skrivtid. Rätt till förlängd tid beslutas av LiU centralt.

## Ämnesinnehåll

Fokus på duggan kommer vara att skapa en minimal klasshierarki och få den att lösa en uppgift. Detta medför att god kunskap om syntax och semantik rörande klasser behövs.

1. Grunder: typer, variabler, satser, std::vector, funktioner, parametrar, argument
2. Klass: objekt/instans, konstruktor, inkapsling, datamedlem, medlemsfunktion, public, private
3. Arv: basklass, härledd klass, anrop av basklasskonstruktorkonstruktor, protected, anrop av basklassfunktion
4. Polymorfi: slicing, virtuell destruktor, dynamisk/statisk bindning, virtual, pure virtual, override
5. Minneshantering: allokering, avallokering, minnesläcka

## Hjälpmedel

Under duggan råder standard tentamensregler. Du får ha med följande hjälpmedel:

- En bok om C++. För boken gäller följande regler:
  - Kommentarer eller noteringar som direkt rör text och exempel på sidan i fråga får finnas i sidmarginalen.
  - Egna sidflikar för att enkelt kunna hitta t.ex. de olika kapitlen är tillåtna.
  - Inga extra ark eller lappar, lösa eller fastsatta, får finnas.
  - Ingen programkod på tomma sidor, insidan av pärmarna, försättsblad, etc.
- Ett A4-ark. Valfritt innehåll på vardera sida. Går ej ersätta med två enkelsidiga ark.
- Penna för att anteckna. Du kan be tentamensvakt om kladdpapper.

Följande får INTE tas med:

- Elektroniska prylar. Till exempel miniräknare, mobiltelefon, hörlurar, tangentbord, smartklocka etc.

## Inlogging

Tentamensvakt kontrollerar anmälan vid inpassage till salen. När datorns inloggningsskärm visar "tentamensläge aktivt" kan du logga in som vanligt med ditt LiU-ID och ditt privata lösenord och börja förbereda din programmeringsmiljö. Tentamensvakten kommer under duggan gå runt för att kontrollera legitimation och hjälpmedel.

## Alias för kompilering

Under duggan finns det tre alias att använda sig av för kompilering med c++17:
`w++17` Rekommenderas!
`e++17` Kompilering med alla varningar som fel.
`g++17` Kompilering utan varningar.

## Frågor

Frågor ska ställas via studentklienten. Detta för att vi ska ha en historik av konversationen samt för att vi ska kunna ge samma hjälp till olika studenter. Räck upp handen om klienten, tangentbordet, musen eller datorn inte verkar fungera som de ska.

## C++ referens

Det finns tillgång till valda delar av `cppreference.com`. Du måste starta webbläsaren via skrivbordsikonen "Web access" för att komma åt sidan.

## Inlämning

När du har en lösning som kompilerar utan varningar, kör utan minnesläckor och löser uppgiften skickar du in den via studentklienten. Din inlämning bedöms direkt[1] och du får svar via studentklienten. Om du får betyg G är det bara att spara, avsluta, logga ut, och framåt kvällen fira! Om du får komplettering läser du noga vad som inte fungerar, fixar till det efter bästa förmåga, och skickar in igen.

## Utloggning

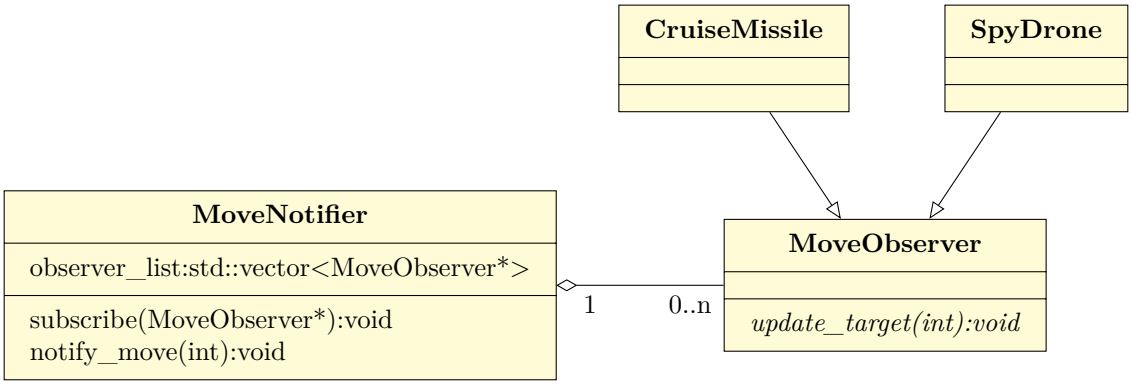
När du är nöjd med ditt betyg (som står i studentklienten) kan du avsluta duggan. Stäng alla program och spara alla filer. Sedan loggar du ut som vanligt i menyn. Lämna inte din plats innan vanliga inloggningsskärmen syns.

[1] Beroende på mängd samtidiga inskickningar kan det ta flera minuter att få svar, räck upp handen om du inte fått svar på 15 minuter.

# Uppgift

Ett vanligt problem inom programmering är att vi har något objekt (provider) med data som är av intresse för ett antal andra objekt (observers). Ett exempel är ett spelobjekt som är mål för ett antal målsökande projektiler. Målobjektet har då en position som är av intresse för projektilerna att observera. Om målobjektets position förändras behöver projektilerna få kännedom om den nya positionen.

Ett vanligt sätt att lösa detta är genom Observer-möstret. I uppgiften implementeras en applikationsanpassad variant av Observer-mönstret där vi namngivit de olika delarna efter vad de är i vårt exempel.




Du ska skapa klassen `MoveNotifier` som har en lista av alla objekt som är intresserade av målobjektets positionsförändringar. När målobjektet ändrar på sin position ber den `MoveNotifier` att skicka en notis om detta till alla dess observers. `MoveNotifier` går då igenom sin lista av alla intresserade objekt och anropar funktionen `update_target` på varje objekt lagrat i listan.

För att enkelt kunna ha fler olika typer av observers deklareras `update_target` i en gemensam basklass. Basklassen namnger vi `MoveObserver`. Den bistår med ett enhetligt gränssnitt som `MoveNotifier` kan använda och som olika typer av målsökande objekt kan ärva in.

Du ska skapa två typer av målsökande spelobjekt härledda från `MoveObserver`:

- `CruiseMissile` ska hela tiden flytta i riktning mot målobjektet. När `update_target` anropas skriver den ut “Move closer to X” där X är målposition eller “Spot on!” om den redan nått målet.
- `SpyDrone` ska hela tiden hålla sig på ett visst avstånd från målobjektet. När `update_target` anropas skriver den ut antingen “Move closer” eller “Move away”.

***Notera*** att objekt i denna uppgift endast rör sig heltalssteg i $X$-led (endimensionellt) och att vi för enkelhets skull låter bli att implementera objektens verkliga `move()`-funktion. Objekten kommer alltså stanna på samma position i denna uppgift. Istället gör `update_target()` ovan angivna utskrifter som visar på vad som bör hända när respektive objekt får reda på att målet förflyttats.

I `target.cc` finns ett givet huvudprogram. Din uppgift är att utöka denna fil med de fyra klasser som beskrivs i texten. Relationen mellan dessa klasser anges i diagrammet ovan. Du behöver själv lägga till de funktioner och datamedlemmar som behövs med ledning av beskrivningarna ovan och huvudprogrammet.

**Körexempel** på nästa sida.

Körning av programmet ska ge följande utdata i terminalen:

```
Target moved to 0
Move away
Move closer
Move closer to 0
Move closer to 0

Target moved to 2
Move away
Move closer
Move closer to 2
Move closer to 2

Target moved to 4
Move away
Move closer
Spot on!
Move closer to 4

Target moved to 6
Move away
Move away
Move closer to 6
Move closer to 6

Target moved to 8
Move closer
Move away
Move closer to 8
Move closer to 8

Target moved to 10
Move closer
Move away
Move closer to 10
Move closer to 10
```