Linköpings Universitet
Institutionen för datavetenskap (IDA)
UPP-gruppen 2022-10-06

Bonusupgift: Release v0.1 (4 poäng)

I denna bonusuppgift ska ni under projektets första två veckor ta fram en tidig version av ert projekt. Er release v0.1 ska innehålla en grundläggande struktur för ert projekt som inkluderar tomma klasser med stubbar (se sida 2) för alla klasser i er OOA samt en huvudloop som gör att något händer när programmet körs.

Syftet med bonusuppgiften är att ge er en bra grund för att kunna fortsätta koda effektivt. Genom att tidigt i utvecklingen ha en delmängd fungerande krav har ni alltid ett givet sätt att testa koden ni skriver, samt ett sätt att testa interaktioner mellan klasser även om koden för klasserna inte är helt komplett än. Vår rekommendation är att sedan fortsätta utvecklingen steg för steg, dvs bestäm vad er release v0.2 ska uppfylla och implementera det innan ni går vidare med release v0.3 osv.

Krav

Följande krav måste vara uppfyllda för att ni ska bli godkända på bonusuppgiften. Ni ska ha implementerat:

- Tomma klasser med stubbar utgående från er OOA.
- Kod som ger en grundstruktur att enkelt bygga vidare på. Detta är i huvudsak en fungerande huvudloop som delegerar allt spelspecifikt till någon av era klasser med hjälp av polymorfi.
- Kod som visar ett visst objekt på skärmen i flera instanser (t.ex. flera färgade kvadrater).
- Kod som visar spelaren (eller något annat objekt) på skärmen. Det behöver inte vara färdig grafik, utan det räcker med t.ex. en färgad cirkel.
- En enkel interaktion som implementerar ett användningsfall, dvs uppfyller (en del av) något projektkrav. Något objekt ska reagera på en användarinmatning (t.ex. spelaren) och något ska reagera på förfluten tid (t.ex. ett objekt). Vad som händer beror på ert spel. Kanske rör sig något på skärmen, eller kanske byter objekt färg (början på en animering). Koden för vad som händer ska finnas i respektive klass.

Redovisning: Denna uppgift är poänggivande för hela gruppen eftersom ni behöver ta ett gemensamt beslut att jobba mot en v0.1 release. Ni ska publicera er release v0.1 på Git inom projektets första två veckor (v45 – v46) och informera er assistent när detta är gjort.

Stubbar

En utmaning när ett projekt ska genomföras är att få till en struktur som håller hela vägen. Tyvärr krävs ofta att alla delar är på plats för att kunna se alla krav som ställs på strukturen. När vi arbetar är det tyvärr också rättfram t att börja med det som är enkelt och snabbt medan vi tenderar skjuta upp svårigheter och besvärliga beslut så länge som möjligt. Det är alltså lätt att sista veckan i projektet upptäcka att strukturen inte fungerar för de besvärligaste kraven. Då är det som regel för sent att lösa någotdera.

Lösningen är att börja med det som är svårast. I regel det ni inte har en aning om hur ni ska få till. Det är ju det som riskerar att ta längst tid. Men för att kunna börja med det svåre, och för att kunna se hela strukturen behövs ofta "resten" av projektet först. Det är nu stubbar kommer in. Att göra stubbar är egentligen konsten att med så liten insats som möjligt skriva ett komplett kompilerande ramverk. Givetvis behöver ni ta en del genvägar för att komma undan med så liten insats som möjligt. Den mest framträdanden genvägen är att varje del i ramverket bara fungerar för någon detalj. Funktionalitet går alltid lägga till senare när strukturen är på plats och funktionaliteten verkligen behövs.

När ni går igenom kraven för ert projekt infinner sig troligen antingen känslan att "detta kommer vi lösa" eller känslan "vi vet inte hur vi ska göra detta". För kraven "detta kommer vi lösa" gör ni stub och väntar med detaljerna. För kraven "vi vet inte hur vi ska göra" använder ni det ramverk era stubbar bildar och experimenterar med olika angreppssätt.

Ur ett mer tekniskt perspektiv är en "stub" en klass eller funktion som är korrekt deklarerad och som har en definition, men inte blivit fullständigt implementerad ännu. Följande är ett exempel med en funktion som ska returnera en sträng, ”am” eller ”pm”, beroende på timme:

Funktionen som en stub:

```
std::string Time::am_pm_string(int hour) const
{
  return "am"; // TODO: just a stub, add functionality later...
}
```

Syftet med stubbar är att vi vet vilka funktioner som finns, vi kan använda dem i koden (även om de inte är fullt implementerade), vi kan kompilera och vi kan skapa tester med dem. Det gör att vi i arbetet med projektet inte behöver vänta på att funktioner ska bli implementerade innan vi kan använda dem. Det gör att vi kan börja med det som känns svårt tidigare. Vi får ett ramverk vi kan använda oss av och bygga ut vartefter det behövs.

När stubben tids nog vidareutvecklats till komplett funktion kanske den ser ut så här:

```
std::string Time::am_pm_string(int hour) const
{
  std::ostringstream os{};
  oss << " ";
  if ( hour < 12 )
  {
    oss << "am";
  }
  else
  {
    oss << "pm";
  }
  return oss.str();
}
```