|  |
| --- |
|  |
| Evil Corporation Calculator |
|  |
| Grupparbete |

**SYSTEMINTEGRATÖR YHSIPI17/ARBETSMETODIK FÖR UTVECKLARE**

den 9 februari 2018

Billy Andersson, Malin Albinsson, Tommy Eriksson

Evil Corporation Calculator

Grupparbete

# Sammanfattning

Detta är en enkel miniräknare byggd i Javafx som klarar att hantera de fyra räknesätten för heltal. Som en bonus-feature ingår operatorprioritet i denna release, vilket möjliggör beräkning även av något mer komplicerade ekvationer än de allra enklaste.

# Inledning/Syfte

Vår uppgift var att leverera källkod och ett färdigbyggt applikationspaket till uppdragsgivaren Evil Corporation som behövde en miniräknare som minst klarade av de fyra räknesätten. Källkoden skulle versionshanteras med git och vi behövde skapa ett repository för detta på github.com.

# Genomförande

## Arbetsmetodik

Vi använde ett repository på github.com där vi alla hade skrivrättigheter. Vi förde det mesta av diskussionerna via Slack.

Vi valde en MIT-licens därför att detta är en enkel applikation där källkoden kommer vara åtkomlig för alla via github. Det är inte heller något nytt, komplicerat eller innovativt så det finns anledning till större begränsningar av den anledningen.

Då vi såg olika möjligheter till att lösa den kodmässiga delen av uppgiften skapade vi varsin dev-branch att arbeta i. Därefter skulle vi välja approach att gå vidare med, och den skulle då lyftas till testing-branchen. Godkänd och körbar version från testing skulle sedan läggas ut i master tillsammans med JAR-filen.

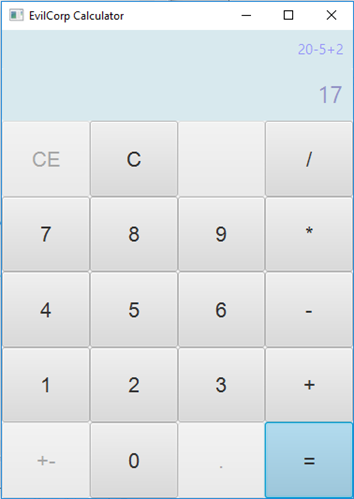
Under arbetets gång blev flödet kanske inte fullt så prydligt, men efterhand närmade vi oss en lite mer genomarbetad arbetsprocess. Mot projektets slut såg den ut ungefär så här:

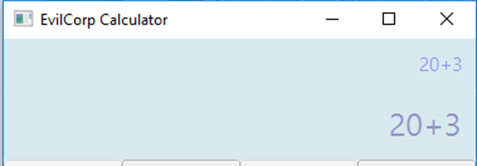
För att göra en ändring hämtade man hem den senaste versionen från testing-branchen till sin develop-branch, implementerade ändringen, skapade en pull-request till testing, någon/övriga granskade den och var den ok så mergade vi med testing (annars stängdes pull-requesten och ändringen förkastades eller modifierades).

Såsmåningom började vi skapa issues, vilket gjorde problemen som skulle lösas mer avgränsade och förenklade arbetet.

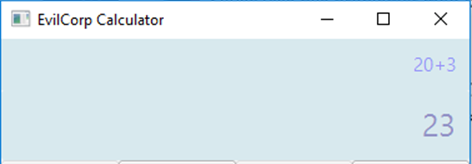
För att släppa en ny relase mergade vi testing med master. Stora ändringar – version 1.0, 2.0 etc, mindre – version 1.1, 1.2 och så vidare.

## Kalkylatorn

Vi valde att arbeta i Javafx och att implementera ett grafiskt gränssnitt för en enkel miniräknare som kan hantera de fyra räknesätten för heltal. Som en extra feature implementerade vi operatorprioritet.

Inmatning sker genom att klicka på knapparna i GUI:t, men kalkylatorn klarar också att hantera tangentbordsinmatningar. Inmatade värden visas i displayen 

och när man väljer likhetstecknet visar displayen resultatet och ekvationsfältet ovanför visar vilken ekvation som gett resultatet ifråga.



För att påbörja en ny beräkning efter att man tryckt på likhetstecknet måste man först nollställa displayen med C-knappen.

Funktionerna addExact, substractExact, multiplyExact och floorDiv från klassen Math har använts för de matematiska beräkningarna.

De knappar som är ljusgrå i gränssnittet står för funktionalitet som inte är implementerad.

# Avgränsning

Detta är en enkel kalkylator som vid första leverans klarar att hantera de fyra räknesätten för heltal med operatorprioritet (multiplikation och division utförs före addition och subtraktion), men däremot inte kan hantera decimaltal eller parenteser. Negativa tal hanteras inte heller. Division med 0 genererar fel och bör i en senare version ge ett tydligt meddelande till användaren om att operationen är otillåten. CE-knappen (clear entry) är inte implementerad. Knappar för icke-implementerad funktionalitet har ljusgrå text.

# Slutsatser/diskussion

## Kalkylatorns implementering

Uppdraget var att göra en enkel kalkylator som klarar att hantera de fyra räknesätten för heltal, och det har vi levererat.

En fördel med vår implementering är att kalkylatorn kan hantera inmatningar både via tangentbordet och genom att man klickar på knapparna i GUI:t. Det gör den mer användarvänlig. En annan fördel är att den visar ekvationen som ligger bakom resultatet som visas på displayen.

Till en början kunde ekvationsvisningen dock i vissa lägen uppfattas förvirrande när ekvationen visade t ex 6+4\*2 och resultatet blev 20 (när hjärnan väntade sig 14 efter att ha utfört multiplikationen före additionen). Denna nackdel valde vi att eliminera genom att implementera operatorprioritet.

En nackdel med nuvarande version av kalkylatorn är dock att för att påbörja en ny beräkning efter att man tryckt på likhetstecknet och visat resultatet, så måste man först nollställa displayen med C-knappen. Att bara direkt skriva en siffra fyller annars bara på resultatet i displayen. Detta är inte i linje med hur exempelvis Windows kalkylator fungerar och bör åtgärdas i en framtida release.

## Arbetsmetodik

En fördel med att arbeta med github är att det är lätt att ta del av vad de andra har gjort.

En nackdel med vårt upplägg var att vi inte alltid var tillräckligt tydliga med vem som skulle jobba med vad, så det blev en del dubbelarbete.

En viktig insikt är att det vid ett samarbetsprojekt där deltagarna sitter på olika ställen och dessutom arbetar på olika tider finns mycket att vinna på att redan innan själva arbetet börjar tydliggöra exakt hur arbetsprocessen är tänkt att se ut. Vi hade varsin utvecklings-branch och en gemensam testing-branch (och förstås gemensam master), men vi kunde varit tydligare från start i vem som skulle ta ansvar för vad, hur vi skulle ta oss mellan brancherna och hur besluten skulle dokumenteras. Issue-hanteringen hade vi också kunnat definiera tydligare. Två orsaker till att vi inte lyckades få till en mer optimal arbetsprocess var att både git och github är relativt nya för oss, och att vi inte tog oss tillräckligt med tid i början för att sätta upp en genomtänkt process. Vi var mer fokuserade på funktionalitet i applikationen.

Om vi skulle göra om projektet idag skulle vi utse en projektledare som fick ha mer fokus på processen medan de andra ägnade sig åt själva koden.

## Resultat

Slutresultatet blev att vi levererade en applikation som uppfyllde kraven, att vi lärde oss en del om Javafx samt en hel del om versionshantering och hur det är att arbeta tillsammans med kod.