Lunds Tekniska Högskola
Datavetenskap
Lennart Andersson

EDAF25
Projekt 2
2021-04-19

Kalkylprogrammet XL

Syfte

Efter att du gjort denna uppgift skall du kunna

- modellera ett system på egen hand,
- designa en MVC-arkitektur,
- utforma ett system med icketriviala beroenden,
- tillämpa principer för paketdesign,
- använda designmönstren *Observer* och *Listener*,
- göra felhantering samt
- implementera UML kvalificerade associationer.

Uppgift

Utforma och implementera det program vars referensmanual finns på projekthemsidan (`manual.pdf`). Programmet är ett mycket enkelt program för kalkylark där man kan skriva enkla uttryck och referera till andra celler i arket.

Som utgångspunkt för arbetet skall användas ett skalprogram som kan laddas ner från projekthemsidan och som beskrivs i detalj längre ner i detta dokument. För att bli klar med uppgiften skall saknad och ofullständig funtionalitet i skalprogrammet slutföras.

Arkitekturen för er lösning skall vara en MVC arkitektur med sammanfogad Modell/Controller: det inte nödvändigt att separera controllern från modellen – det blir enklast att ha dem som en enhet i det här programmet. Den huvudsakliga uppgiften i projektet är att designa och implementera modellen som kan hålla reda på all information i kalkylarket och beräkna alla cellers värden när något uttryck i en cell ändras.

Designkrav

Modellen skall separeras från vyn (användargränssnittet) med hjälp av *Observer*-mönstret så att de klasser som utgör modellen skall kunna kompileras utan tillgång till något användargränssnitt.

Skalprogrammet

Skalprogrammet har ett minimalt användargränssnitt som skall utökas så att programmet uppfyller funktionaliteten som beskrivs i referensmanualen. Skalprogrammet har även klasser för att representera och beräkna aritmetiska uttryck.

Skalprogrammets användargränssnit är utvecklat med JavaFX, ett modernt bibliotek för att bygga användargränssnitt i Java. För att kompilera och köra programmet bör en Java 11 utvecklingsmiljö användas. Se avsnittet Utvecklingsmiljö för mer information.

I skalprogrammet finns följande paket:

expr innehåller klasser för att representera aritmetiska uttryck med metoder för att beräkna värdet av ett uttryck och att skapa den interna representationen från strängar. Du behöver inte göra några förändring eller tillägg av klasserna i detta paket. Om du anser att det är nödvändigt skall du diskutera detta med handledaren innan du gör förändringen. Paketet innehåller en test-klass som illustrerar användningen.

gui innehåller det grafiska användargränssnittet. Huvudklassen är `gui.XL`, en JavaFX-applikation som innehåller det mesta av koden för användargränssnittet. I gränssnittet kan man markera celler genom att klicka på dem i kalkylarket, men funktionaliteten för att spara en ny ekvation i en cell saknas och behöver läggas till.

gui.menu innehåller klasser för programmets meny-rad. Här skall du lägga till funktionalitet som saknas – den enda funktionalet som redan finns färdig är Exit-alternativet i File-menyn.

util innehåller några klasser som behövs i flera paket och några som skall flyttas till ett annat paket. `XLPrintStream` behöver en liten anpassning till din modell medan `XLBufferedReader` behöver kompletteras. Studera dokumentation av `Set`, `Map` och `Map.Entry` i `java.util`.

För att implementera er lösning kan ni göra ändringar i befintlig kod så som att att läga till attribut i klasser eller att lägga till parametrar till befintliga metoder/konstruerare. Lägg dock märke till att det är ganska få ändringar som behöver göras i befintlig kod för att slutföra projektet och om ni märker att ni håller på att ändra väldigt många saker så kanske ni är på ett dåligt spår och har en för komplicerad design.

Beräkning av uttryck

Gränssnittet `Expr` specificerar metoden `value(Environment): ExprResult`, som returnerar uttryckets värde. Parametern används för att få tag på variablers värden. Variabler är i det här fallet referenser till celler (t.ex. `B3`). Paketet innehåller också `ExprParser` som används för att bygga `Expr`-objekt.

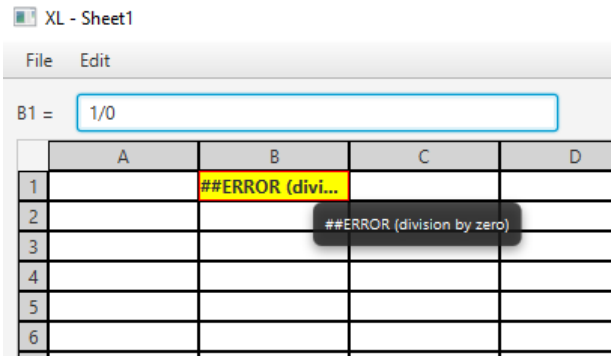
Om det finns syntaxfel i ett uttryck så kastas undantag av typen `XLException` från `ExprParser`.

Felhantering

Programmet skall inte krascha eller bete sig dåligt om det finns fel i någon ekvation i en cell i kalkylarket. Fel skall istället visas som text i de celler som har fel i sina ekvationer: då kan användaren se felet och ändra ekvationen för att korrigera felet.

För att hantera ekvationer med fel i så finns gränssnittet `expr.ExprResult` som representerar resultatet av en ekvation – resultatet kan vara antingen `ValueResult` för en felfri ekvation eller `ErrorResult` om det fanns ett fel som förhindrade ekvationen från att beräknas.

Fel som kan finnas i en ekvationer är: syntaxfel i ekvation, division med noll, att en tom cell refereras, att ekvationen har ett cirkulärt beroende, att värdet av en kommentar-cell refereras, eller att en cell med ett annat fel i refereras. Dessa fel bör visas som text inuti en cell, och får ej krascha programmet.



Filformat

XL-programmet skall kunna spara och läsa in hela kalkylark till/från fil.

Katalogen `test` innehåller några testfiler. De flesta av dessa är felfira men det finns även filer som innehåller `error` i namnet och som har fel i ekvationerna. Dessa filer kan användas för att testa att inläsning från fil fungerar, samt att felhanteringen fungerar.

När kalkylark sparas som filer skall innehållet i varje icketom ruta sparas som en sträng med adressen, ett "=" och innehållet på samma sätt som det visas i editorn. Varje ruta beskrivs på en rad. Alla mellanslag är signifikanta. Det kalkylark som som visas i referensmanualen skapas om en fil med följande innehåll öppnas.

```
a1=#x =
a2=#y =
a3=#x*y =
b1=2
b2=3
b3=b1*b2
```

Klassen String innehåller lämpliga metoder för att extrahera komponenterna i en rad.

Utvecklingsmiljö (IntelliJ/Eclipse)

Skalprogrammet är ett Gradle-projekt som kan byggas och köras från en Linx/Mac/Windows-terminal med kommandot `./gradlew` (på Windows måste man utesluta `./`). För att köra programmet kan man lägga till `run` efter `gradlew`, och för att köra test så används `test` kommandot.

För att få en bra utvecklingsmiljö kan man använda IntelliJ: det går enkelt att importera Gradle-projektet till IntelliJ (testat med version 2021.1). Eclipse kan också användas men det kan vara knepigt att få Eclipse att hantera beroendena från Gradle-projektet korrekt, så det är rekommenderat att använda IntelliJ om möjligt.

Om ni vill använda Eclipse så kan man importera skalprogrammet som ett Gradle-projekt i Eclipse. Om det finns många fel i programmet efter import så kan man provat att först köra `./gradlew cleanEclipse eclipse` i terminalen för att generera ett Eclipse-projekt och sedan öppna det i Eclipse utan att importera Gradle-projektet. Om projetket redan var öppet i Eclipse så kan man markera projektet och välja "Refresh" för att ladda om det från hårddisk. Detta testades i Eclipse version 2020-12 (4.18.0).

Redovisning

Gruppen skall träffa en lärare vid två designmöten. Vid det första mötet skall användningsfall, paketindelning och klassdiagram för programmet presenteras. Elektronisk inlämning skall göras senast 24 timmar före mötet till kursens Canvas-sida.

Designmötena är en del av examinationen och därmed obligatoriska. Om du blir sjuk eller har ett godtagbart skäl att utebli skall du meddela din lärare, om möjligt i förväg.

Inlämningen skall omfatta användningsfall och klassdiagram.

- Varje användningsfall skall beskrivs med några rader. Beskrivningen skall ange vad användaren gör, vad som skall hända utifrån användarens perspektiv och vilka fel som kan inträffa. UML-diagram för användningsfall skall ej konstrueras.
- Ett klassdiagram för varje paket. Diagrammen skall vara skapade eller genererade med ett datorbaserat verktyg och skall visa klasser, attribut, metoder, arv och generaliseringar men inte associationer och andra beroenden. Diagrammen bifogas som jpg, tiff, png eller pdf.

Inför arbetet i gruppen bör du fundera på följande. Frågorna kommer att dryftas på första designmötet.

1. Vilka klasser bör finnas för att representera ett kalkylark?
2. En ruta i kalkylarket skall kunna innehålla en text eller ett uttryck. Hur modellerar man detta?
3. Hur kommer ni hantera fel i uttrycken i en eller flera celler?
4. Vilka klasser skall vara observatörer och vilka skall observeras?
5. Vilket paket och vilken klass skall hålla reda på vad som är "Current slot"?
6. Vilken funktionalitet är redan färdig och hur fungerar den? Titta på klasserna i `view`-paketet och testkör.
7. Det kan inträffa ett antal olika fel när man försöker ändra innehållet i ett kalkylark. Då skall undantag kastas. Var skall dessa undantag fångas och hanteras?
8. Vilken klass används för att representera en adress i ett uttryck?
9. När ett uttryck som består av en adress skall beräknas används gränssnittet `Environment`. Vilken klass skall implementera gränssnittet? Varför använder man inte klassnamnet i stället för gränssnittet?
10. Om ett uttryck i kalkylarket refererar till sig själv, direkt eller indirekt, så kommer det att bli bekymmer vid beräkningen av uttryckets värde. Föreslå något sätt att upptäcka sådana cirkulära beroenden! Det finns en elegant lösning med hjälp av strategimönstret som du får chansen att upptäcka. Om du inte hittar den så kommer handledaren att avslöja den.

Inför det andra mötet skall klassdiagram och källkod för ett fungerande program lämnas in. Elektronisk inlämning skall göras senast 24 timmar före mötet. Inlämningen skall innehålla ett klassdiagram för varje paket utom `expr` och en zip-fil som innehåller hela projektkatalogen.

Korrigeringar och kompletteringar till detta dokument kan dyka upp på kursens hemsida.