Kalkylprogrammet XL

Syfte

Efter att gjort denna uppgift skall du kunna

- modellera ett system utan att uppdragsgivaren styr designen,
- använde MVC-arkitektur,
- utforma ett system med icketriviala beroenden,
- tillämpa principer för paketdesign,
- använda designmönstren *Observer* och *Listener*,
- göra felhantering samt
- implementera kvalificerade associationer.

Uppgift

Utforma och implementera det program vars referensmanual finns på projekthemsidan. Programmet liknar i vissa avseende ett kommersiellt program med ett likalydande namn. Notera dock att varje kalkylark i XL har ett eget fönster i användargränssnittet med egna menyer och att tomma rutor ej får refereras i uttryck.

Designkrav

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

Implementering

Du skall själv designa och implementera de klasser som utgör modellen. Modellen skall hålla reda på den information som en användare av programmet matar in via användargränssnittet.

Huvuddelen av användargränssnittet och ett paket för att representera aritmetiska uttryck finns att hämta på projekthemsidan. Där finns också ett dokument som ger en översiktlig beskrivning av denna kod.

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 utan några referenser till den modell som du skall konstruera. Det går att starta programmet, men nästan all funktionalitet saknas. Du skall tillfoga funktionaliteten genom att komplettera lyssnare med anrop av metoder i modellen och observatörer som uppdaterar vyn när tillståndet i modellen förändras. Detta kräver att du tillfogar attribut och konstruerare i många klasser samt implementerar `update`-metoden i de klasser som skall vara observatörer. Eventuellt kan det vara bra att tillfoga någon klass. Klassen `XL` innehåller en `main`-metod för att starta kalkylprogrammet.

gui.menu innehåller meny-klasserna. Du skall tillfoga funktionaliteten genom att implementera metoden `actionPerformed`. Detta kräver ofta att klassen ges tillgång till vyn eller modellen. Några klasser är kompletta.

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`.

Katalogen `test` innehåller några testfiler. Ert program får förutsätta att filer som läses är skapade med ett program som inte kan spara kalkylark med fel. Man kan i allmänhet inte räkna ut värdet i en ruta innan hela arket är inläst. De filer som innehåller `error` i namnet är manuellt konstruerade och innehåller fel. Ert program behöver inte hantera dessa fel men de skall föranleda en felrapport, till exempel som ett ej fångat undantag i terminalfönstret.

Paketen och testkatalogen finns att hämta från projekthemsidan som ett eclipse-projekt.

Det är ett krav från uppdragsgivaren att minnesbehovet för modellen av kalkylarket ej skall bero på arkets storlek utan bara på den mängd information som matats in i arket.

När kalkylblad 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. Om någon rad är syntaktiskt felaktig eller beräkningen av kalkylarket ger fel räcker det att programmet rapporterar det fel som upptäcks först.

Gör inga optimeringar, utom den som uppdragsgivaren föreskrivit, förrän det visar sig att de behövs!

Redovisning

Gruppen skall 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 `edaf10` på domänen `cs.lth.se` med Subject-raden *`xl by user1 user2 user3 user4`* . Grupperna registrerar sig via Sam.

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 text, ungefär som rutorna i Martin på sidorna 195–198. 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 Eclipse-baserat 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 skall man hantera uppdragsgivarens krav på minnesresurser?
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.

Till det andra mötet skall klassdiagram och källkod för ett fungerande program inlämnas. 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 eclipse-projektet. E-postbrevet skall också innehålla uppgift om hur mycket tid gruppmedlemmarna använt för inlämningsuppgiften.

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