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 som bilaga. 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.

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.

view 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 `Gui` innehåller en `main`-metod för att starta kalkylprogrammet.

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

xl innehåller några abstraktioner och klasserna `XLBufferedReader` och `XLPrintStream`. De senare skall flyttas till det paket där de hör hemma. `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. De filer som innehåller `error` i namnet är manuellt konstruerade och innehåller fel som programmet skall upptäcka. Filer som sparas med programmet skall vara korrekta och kunna läsas in igen.

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 kursnamnet (`eda061` eller `edaf10`) på domänen `cs.lth.se` med

Subject-raden *xl* by *user1 user2 user3 user4* . För eda061 är mötet schemalagt till vecka 5 i HT1. För edaf10 är motsvarande möte schemalagt till vecka 2 i HT2. Grupperna registrerar sig via Sam.

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 alla nya 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 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?
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 som du får chansen att upptäcka. Om du inte hittar den så kommer handledaren att avslöja den.

Vid det andra mötet (i vecka 7 under HT1 resp. vecka 3 i HT2) skall ett fungerande program demonstreras och förändringar i designen redovisas. 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. Eventuella kompletteringar skall utföras före läsperiod 2 resp. senast vecka 4 i HT2.

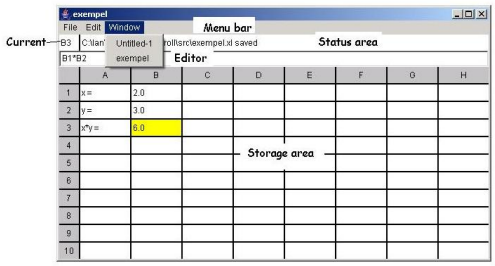
Det är tillåtet för studenter i eda061 att göra inlämningar enligt edaf10:s schema och omvänt efter anmälan till kursansvarig.

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

Introduction

The XL[1] spread sheet program provides a scratch pad for problems involving arithmetic. The program maintains five different areas on the screen: the menu bar, the current slot indicator, the status field, the editor, and the storage area. The menu bar has two pull down menus. The current slot indicator and the status field are located just below the menu bar. They are used for displaying information and error messages about the spread sheet.

The editor is located at the following line and provides a simple line editor that can be used for entering data into the storage area. The storage area occupies the remainder of the frame window. It will show the contents of the spread sheet, as it is interactively changed through the editor and the menus.



The storage area

The storage area is a matrix of slots, with columns ranging from A to H and rows from 1 to 10. The left border of the storage area shows the line numbers and the top border shows the column letters. Each slot can store an expression or a comment. If a slot contains an expression the computed value of that expression is displayed. If it contains a comment the text is displayed. Each slot has an address constructed by the column and row indices. The address of the first slot is *A1* or equivalently *a1*. The last slot is named *H10* or `h10`.

The user may construct expressions over slots and thereby produce new values from those already present in the storage area. The main purpose of the spread sheet program is to relate the positions of the numbers rather than the actual numbers. An example: If the numbers 2 and 3 are entered into the `b1` and `b2` and the expression *b1*b2* is entered into `b3` then the value displayed in `b3` will be 6. Since the program stores the relations between the slots it is possible to change the value of `b1` or `b2`, forcing that also the value of `b3` is changed to remain the product of the values in `b1` and `b2`.

Each slot can be of three kinds: *blank*, *expression* or *comment*. A blank slot has no value. A comment is a string.

The value of an expression slot is a floating point number. An expression is built of constants, slot addresses, additions, subtractions, multiplications, and divisions. Parentheses may be used to change the order of computation in the usual way. These expressions are entered by the user and may be changed at any time.

[1] This programming exercise is adapted from "Liskov, Guttag: Abstraction and Specification in Program Development".

An expression slot must not refer to an empty slot. A comment slot will return the value 0. An expression slot must not refer to it self directly nor indirectly. E.g. you must not store the expression *a1+1* in slot `a1` and you must not store the expression *b2* in slot `b1` and the expression *b1* in slot `b2`. In such cases the program will report an error in the status area and leave the contents of the storage unchanged.

It is not possible to enter an expression so that a division by 0 occurs anywhere in the storage area.

There is always one slot in the storage area that is considered to be the *current slot*. This slot is marked with a background color and its address is displayed in the current slot indicator. The initial current slot is `A1`. Clicking on another slot will make it current.

The contents of the current slot will be displayed in editor when a new current slot is selected. If the slot contains an expression then this expression is displayed without unnecessary parentheses.

The editor

The editor is used to enter an expression or a comment into the current slot. Anything starting with a # character is considered to be a comment. The # character is not part of the comment and is not shown in the storage area.

If the editor is not empty and the return key is pressed its contents is put into the current slot provided that no error occurs.

The menu bar

The menu bar has three pull down menus. The **File** menu has five alternatives.

New Creates a new XL frame. The title of the frame is `Untitled-` followed by a running number. Each new frame has its own menu bar, current slot indicator etc.

Print A print dialog opens and the current spread sheet can be printed.

Save A file dialog opens and the contents of the current spread sheet may be saved.

Open A file dialog opens and the contents of the current contents may be replaced by the file contents.

Close The current XL window is closed and all information contained i the spread sheet is lost. Closing the the last window will terminate the program.

The **Edit** menu has two alternatives; one for making the current slot empty and one for clearing all slots. The **Window** menu shows a list of all open spread sheets. Selecting an item will bring the corresponding window to the front.

The status area

This area is used for displaying error messages.

Current slot indicator

At all times the current slot indicator displays the address of the current slot.