Laboration 1c: En översättare (ordbok)

1 Syfte

- Fortsättning på testdriven utveckling, programmeringsprinciper och designmönster.
- Konstruktion av applikationer med grafiska användargränssnitt (MVC-modellen)

2 Uppgift

Vi skall nu använda modulerna från föregående laborationer för att på så sätt skapa ett helt fungerande program. Genom vår noggranna och systematisk arbete samt modulernas goda design skall detta vara en relativt smärtfri process. Vi förutsätter pekskärm (som simuleras m.h.a. musen), d.v.s. tangentbord behöver inte fungera.

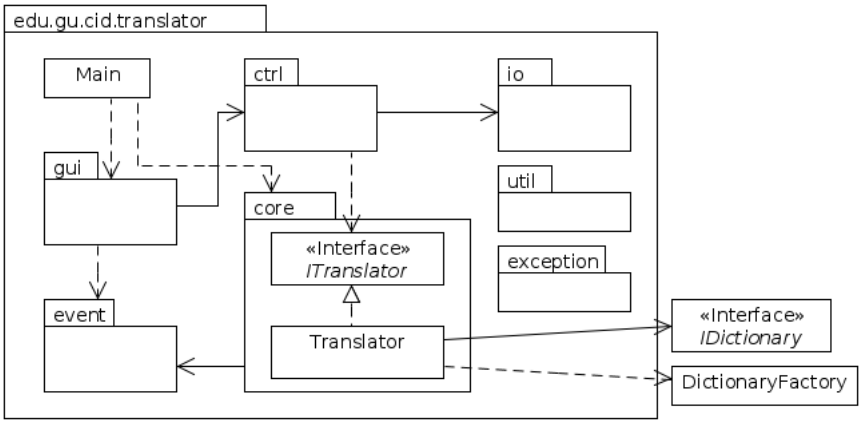
3 Funktionalitet

Funktionaliteten ges av GUI:et, se lab1a, förutom en extra finess; applikationen skall komma ihåg senast valda språk. Se vidare nedan. Försök att tänka utifrån användarens perspektiv. Hur skall GUI:et bete sig då man t.ex. byter språk? Tänk igenom detta nu så slipper ni göra om sedan (skriv ner).

4 Designmodell

En övergripande design visas i Figur 1.

- Main är en klassen som ansvarar för att starta applikationen. Innehåller static public void main(). Skapar applikationens huvudfönster och modellen.
- ctrl innehåller klasser som ingår i control-delen av MVC.



Figur 1: Övergripande design för översättaren.

- io innehåller en klass för att läsa/skriva senaste språkval till disk.
- gui innehåller view-klasserna i MVC, någon hjälpklass och en fabrik för att konstruera GUI:et. Ni behöver inte koda något "visuellt", allt som skall synas på skärmen är klart. Däremot måste ni implementera händelsehanteringen (actionPerformed()) och observatörs-delarna (onEvent() om ni använder EventBus).
- core paketet innehåller modellen i MVC (två klasser) och gränssnittet till denna. Använder IDictionary och DictionaryFactory. Ni skall själva bestämma metoderna i ITranslator och implementera dessa i Translator.
- util innehåller applikationsoberoende hjälpklasser.
- event innehåller de klasser som implementerar EventBus (en variant av Observer-mönstret).
- exception innehåller en applikationsspecifik exception-klass.

4.1 MVC

Applikationen skall byggs enligt MVC-modellen. Dessvärre finns flera sätt att implementera MVC modellen. Utdelad kod kanske måste ändras ifall ni vill göra på något annat sätt. En sak som påverkar mycket är hur ITranslator:s metoder beter sig, de anropas från control-klasserna.

- Om ITranslator:s metoder har returvärden så kan control-klassen skicka detta till view, sker genom anrop via gränssnittet util.IWriteable (annars cirkulära beroenden mellan view och control) eller annat sätt.
- Om metoderna är void så kan data skickas till vyn m.h.a. observer-mönstret.

Den konkreta implementeringen av observer-mönstret kan göras på valfritt sätt, dock rekommenderas EventBus (om EventBus inte används, ta bort paketet).

- Antingen ligger händelsehantering i någon modell-klass eller så kan ni även använda en Proxy.

4.2 Felhantering

Fel hanteras i första hand där man kan göra något för att rätta till det hela. Går inte detta fångas felen i control-klasserna. Visa en dialogruta (JOptionPane). Finns en Exception-klass. Översätt lågnivå- fel till denna typ.

4.3 Användarvänlighet

Applikationen skall bete sig på ett människovänligt sätt, d.v.s. orimliga val skall justeras till rimliga. T.ex skall programmet smidigt hantera om man väljer samma från- och tillspråk.

5 Implementation

Tips: Ha alltid en UML-skiss så att ni vet var ni är, vad ni håller på med och så att båda är införstådda med detta!

Ni måste förstå MVC-modellen innan ni börjar implementera! Följande process rekommenderas;

1. Ta fram någon (en) central metod i gränssnittet ITranslator och implementera denna i Translator (utan observer-delar). Testa. Se till att man kan sortera klassen Language och att man kan serialisera den till disk.

2. Implementera så att hela kedjan; knapp->lyssnare->control->model->observer->view fungerar för den implementerade metoden. Om control:en fungerar som lyssnare blir kedjan kortare. Det finns färdiga controllers. Beroende på implementation lägg ev. till observer-funktionalitet i Translator. Kan isolera servicen genom att använda privata set-metoder för attribut.

3. Fortsätt på samma sätt med övriga metoder. Skapa en separat control för språkvalen.

4. Hantering av senast valda språk: Lägg till en control som anropas då programmet startar och en anropas när det avslutas (windowOpened och windowClosing-metoderna).

6 Android App

(Frivilligt) Så vitt jag kan bedöma skall vi kunna flytta hela applikationen, förutom GUI:et, till någon Android-baserad telefon. Om du har tid och lust prova...

7 Redovisning

Körningsgodkännande samt kodinspektion. Görs under laborationspassen. Se till att bli avprickad. Följande kommer att kontrolleras:

- Inga tester behövs.
- Inga varningar skall finnas.
- Vi kommer att göra en "code review" (kodanalys) av hela applikationen genomgånget material några exempel:
  - stil, namngivning, ...
  - programmering mot gränssnitt
  - beroenden, cirkulära, ...
  - informationsgömning, klasser, paket, hantering av null
  - immutabilitet o.s.v.
  - design, klasser, metoder, ...
- Något verktyg för cirkulära beroenden skall köras t.ex. STAN eller JDepend.

Inlämningsdatum Se kurssida.