Laboration 1c: En översättare (ordbok)

1 Syfte

- Fortsättning på 1a och 1b.
- 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?

4 Designmodell

En övergripande design visas i Appendix (en hel del utelämnat, för detaljer se koden).

- Main är en klassen som ansvarar för att starta applikationen. Innehåller static public void main().
- gui-paketet innehåller allt som har med det grafiska gränssnittet att göra .tex. MainFrame som är applikationens huvudfönster. Desutom finns en fabrik som bygger hela 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).

- LanguageCtrl är en klass som samordnar interaktionen mellan GUI:et och modellen då språk skall bytas.
- core paketet innehåller MVC-modellen, d.v.s. klassen Translator, och gränssnittet till denna samt några hjälpklasser. Gränssnittet är det som GUI:et skall arbeta mot. Ni skall själva bestämma metoderna i gränssnittet och implementera dessa i Translator. EventTranslator är en utökning av Translator som använder EventBus:en för att skicka händelser (till GUI:et). D.v.s. i modellen, Translator, finns ingen kod med EventBus inblandad.
- event-paketet innehåller de klasser som implementerar EventBus (en variant av Observer-mönstret).

4.1 MVC

Applikationen skall byggs enligt MVC-modellen. Dessvärre finns flera sätt att implementera MVC modellen (ingen allmänt accepterad standar). 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;

- Om ITranslator:s metoder har returvärden så kan dessa ibland användas för att uppdatera GUI:et. Kan innebära att vi måste skapa associationer mellan kontrol- och GUI-klasser och/eller mellan GUI:klasser.
- Om metoderna är void så kan data skickas till vyn m.h.a. observer-mönstret. Alternativt: Metoderna har returvärden men man använder inte dessa.

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

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 lågnivåfel 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. Vad skall visas vi tom inmatning m.m.?

5 Implementation

Varning: Ni måste ha förstått MVS/EventBus innen ni börjar. Finns kodexempel på kurssida.

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

Följande process rekommenderas (det tar en stund innan man har bekantat sig med applikationen, gå igenom koden och försök förstå);

1. Ta fram någon (en) central metod i gränssnittet ITranslator och implementera denna i Translator (utan observer/EvenBus-delar). Testa!
2. Läggt till klassen EventTranslator som ärver Translator och låt klassen override:a metoden från föregående. Skicka en lämplig händelse.
3. Implementera så att hela kedjan; knapp -> lyssnare (->control) ->model -> eventbus ->view fungerar för den implementerade metoden. Beroende på implementation lägg ev. till observer-funktionalitet i Translator. Kan isolera servicen genom att använda privata set-metoder för attribut.
4. Fortsätt på samma sätt med övriga metoder. Skapa en separat control för språkvalen.

6 Användarinställningar

(Frivillig uppgift)

1. Lägg till så att applikationen kommer ihåg senast valda språk när den startar. Använd den givna Serializer för att t.ex. skriva ut en Map med inställningar. När programmet startar läser man in Map:en och sätter värden.

7 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...

8 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.

Appendix