Laboration 1b: En lexikonmodul

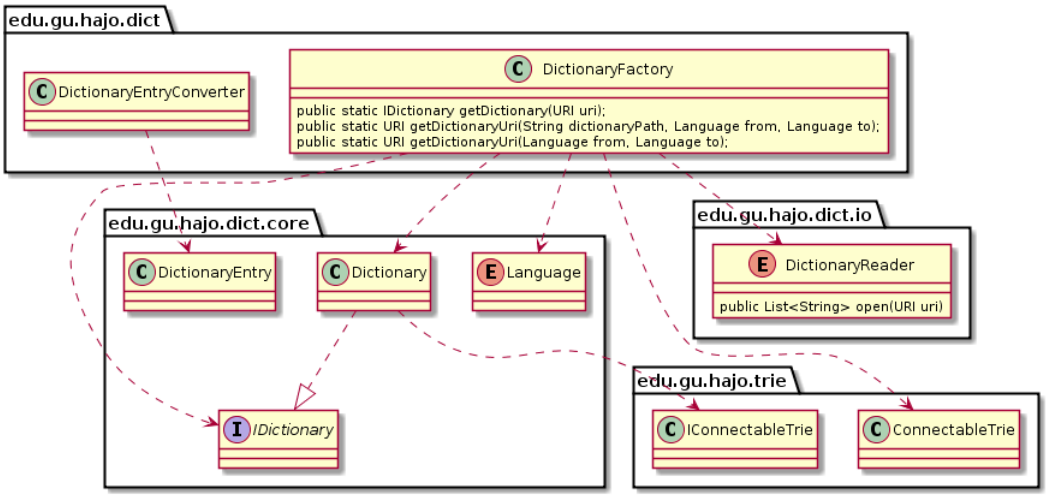
1 Syfte

Fortsättning på laboration 1a.

2 Uppgift

Ni skall implementera Dictionary-modulen, se lab 1a. Modulen använder typer från Trie-projektet. Vi måste därför lägga till ett beroende (dependency) på trie-projektet i pom.xml filen för directory projektet (inspektera pom.xml).

3 Designmodell



Figur 1: Designmodell för Dictionary.

- Dictionary är implementationen av gränssnittet IDictionary som Translator senare skall använda sig av. Dictionary använder sig av två IConnectableTrie (objekt som implementerar IConnectableTrie). En för utgångsspråket och en för målspråket.

- DictionaryEntry är en given klass som håller ett ord och alla översättningar. DictionaryEntryConverter omvandlar mellan strängar och DictionaryEntry. Ordlistorna lagras som textfiler (strängar).
- Paketet io innehåller en Singleton, DictionaryReader, för att läsa rader från en textfiler (skall egentligen ligga i egen modul).
- DictionaryFactory är en fabrik för Dictionary-objekt. Fabriken använder DictionaryReader, DictionaryEntryConverter m.m. för att skapa ett Dictionary (returneras som gränssnitts-typen IDictionary). Dessutom finns två allmänna hjälpmetoder som en service till Translator.
- Language är en given klass för att representera språk.

4 Problem

Tyvärr upptäcker vi i detta läge att Trie-modulens gränssnitt eventuellt innehåller (med avsikt) en grov felaktighet. Dictionary kommer att kunna skaffa sig alldeles för mycket information om ConnectableTrie! Vad är problemet? Åtgärda! TIPS: Använd en wrapper-klass, se returtyp för insert (klassdiagram i lab 1a).

Amn. Har vi en robust design skall detta inte leda till några större problem.

5 Data

Det finns textfiler som innehåller översättningar mellan Svenska och Engelska (US) m.m.. Se Other Sources/dict i projektet.

6 Felhantering

Ingen felhantering här, felen skickas vidare till Translator, använd throws.

7 Implementation

Implementera de delar som fattas för att få ett fungerade lexikon. Det finns ett påbörjat NetBeans-projekt på kursidan. Några punkter;

- Du får själv bestämma vilka metoder som skall finnas i IDictionary. Tänk utifrån användarens (Translators) perspektiv. Translator klassen kommer att hantera knapptryckningar från GUI:er. Vad skall ske då man t.ex. trycker ett “a”? Vilka medtoder i IDictionary skulle behövas?
- Klassen Dictionary implementerar IDictionary. Klassen kan ha metoder som används internt i modulen, dessa ingår inte i gränssnittet.

- DictionaryFactory påminner om Factory-metoden i ConnectableTrie. Metoden getDictionary konstruerar hela lexikonet och returnerar ett objekt som implementerar IDictionary. Metoden skall ta en URI som anger vart modulen skall hämta ordlistan som den behöver (från fil eller över nätet eller,..). För att underlätta detta finns 2 hjälpmetoder för att konstruera URI:er. Se eventuellt javadoc för klassen URI.
- Se till så att man kan sortera DictionaryEntry utifrån utgångsspråket. På så sätt kan användaren få en sorterad lista med ord och översättningar (översättningar ej sorterade). Sortering skall göras med Collections.sort(..).

8 Testning

Det finns färdiga och/eller påbörjade tester. Gör klart eller skapa övriga nödvändiga (icke-triviala) tester parallellt med implementationen.

9 Redovisning

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

- Inspektion av reviderad Trie-modul.
- Allmän kodstil, indentering, krullparenteser, hårdkodade värden, ...
- JUnit-tester skall finnas och vara godkända.
- Inga varningar skall finnas.
- Så lite som möjligt av modulen skall synas utåt.

Inlämningsdatum Se kurssidan.