Markov Monster Chat
Specyfikacja implementacyjna

Adrian Zdanowicz

18 maja 2015

UWAGA: Poniższy dokument utworzony został w oparciu o wczesną koncepcję programu. Niektóre z zawartych w nim założeń oraz opisywane sposoby użycia mogą ulec zmianie w późniejszej fazie tworzenia projektu.

1 Język

Program **Markov Monster Char** napisany jest w języku Java przy użyciu środowiska IntelliJ IDEA.

2 Zewnętrzne biblioteki

Powłoka graficzna programu tworzona jest z pomocą biblioteki Swing.

3 Struktury danych

Program podzielony będzie na dwa główne pakiety - **monster** oraz **gui**.

3.1 monster

Pakiet **monster** spaja klasy, które implementują rdzeń programu - funkcjonalność analogiczną do tej oferowanej przez projekt stworzony w języku C.

Głównym elementem pakietu jest klasa **Monster**, która funkcjonuje jako główna klasa pakietu. Tak jak w projekcie w jęz. C, przechowuje ona ustawienia generatora oraz zawiera klasy **Pool** (przechowującą dane o słowach) oraz **MarkovGen** (przechowującą dane o n-gramach).

Publiczne metody udostępniane przez **Monster**:

```
void ReadTextFiles(String[] fileNames);
void GenerateStats();
void ReadDictionaryFiles(String[] fileNames);
void GenerateDictionaryFile(String fileName);
```

```
// Do czytania oraz generowania odpowiedzi
void ReadChatLine(String line);
String GenerateNextChatLine();
```

Dla uproszczenia programu, pole przechowujące referencję na klasę Pool będzie dostępne dla innych klas, dzięki temu w razie potrzeby pakiet **gui** może odczytać statystyki bezpośrednio z publicznych pól klasy Pool bez potrzeby wprowadzania nowych typów danych.

Klasa **Pool** została uproszczona względem implementacji w języku C (ponieważ Java udostępnia struktury danych, które poprzednio implementowano w Pool ręcznie) i teraz jest głównie wrapperem wokół wektora klas WMEntry:

```
public class WWEntry
{
        public String           rawWord;
        public int                      wordHash;
        public int                      index;

        public void                     weHaveUsedAWord();
};
```

3.2 gui

Pakiet **gui** składa się z klas wygenerowanych przez środowisko podczas generowania okienka programu. Action listenery wykonują operacje, wywołując odpowiednie metody z pakietu **monster**.

4 Wzorce projektowe

Kluczową rolę przy tworzeniu kodu programu odgrywa fasada, ukrywająca implementację klas. Fasada zostanie zastosowana na kilku poziomach, tj. ukryta zostanie zarówno implementacja głównej klasy Monster, jak i (w miarę możliwości) klas używanych przez nią.

Z racji braku potrzeby polimorfizmu, w projekcie nie znajdą zastosowania bardziej złożone wzorce projektowe, takie jak np. fabryka.

5 Interfejs programu

Interfejs programu w całości realizowany jest przez powłokę graficzną, prezentowaną wcześniej w specyfikacji funkcjonalnej.

6 Testy

By zweryfikować poprawność wytworzonego kodu, do klas stworzone zostaną testy jednostkowe. Przewidziane jest głównie bezpośrednie testowanie klasy

Monster, jednak w przypadku wystąpienia jakichkolwiek problemów z klasami, na których zależy główny moduł, zostaną dopisane do nich odpowiednie testy. Mowa tu głównie o testach poprawności operowania na kontenerze słów oraz poprawności generowania łańcuchów Markova.

Interfejs graficzny będzie testowany ręcznie.