Java Techniki Programowania

Zestaw 2.

Termin 2 IV 2009

1 Rate, Office, Money

1. Napisać klasę `Rate`, która będzie służyła do przeliczania pary walut po odpowiednim kursie. Klasa zawiera następujące pola (`private`):

   - `String base` Nazwa waluty bazowej, np. "PLN".
   - `String currency` Nazwa waluty obcej, np. "EUR".
   - `float bid` Kurs kupna.
   - `float ask` Kurs sprzedaży.

   Napisać konstruktory, które przyjmują wszystkie cztery parametry, tylko trzy ostanie oraz tylko nazwe waluty.
   Zaimplementować następujące funkcje

   - `set*` Ustawia odpowiedni parametr.
   - `get*` Zwraca wartość odpowiedniego pola.
   - `float buy(float val)` Zwraca kwotę w walucie `currency` jaką można kupić za `val` w walucie `base`.
   - `float sell(float val)` Zwraca kwotę w walucie `base` jaką można kupić za `val` w walucie `currency`.

2. Napisać klasę `Office`, która pełni rolę kantora. Klasa ma przechowywać kursy (`Rate`) dla wielu walut i na żądanie przeliczać pomiędzy nimi po zadanym kursie. Zaimplementować następujące metody:

   - pobiera kurs m.in. z pliku CSV.
   - dodaje nowy kurs.
   - modyfikuje kurs.
   - zwraca kurs bid i ask dla podanej pary walut.
   - `float change(String src, String dest, float val)`: zwraca kwotę w walucie `dest` jaką można kupić za `val` w walucie `src`.
   - Powyższe metody mają wyrzucać wyjątek `RateNotFound` gdy podana para nie istnieje.

3. Zaimplementować wyjątek `RateNotFound` (klasa która rozszerza java.lang.Exception).

4. Zaimplementować klasę `Money`, która przechowuje pewną kwotę w zadanej walucie.
   Napisać metody, które pozwalają na:

   - Ustawienie kwoty i waluty.
   - Zwrot kwoty i waluty.
   - Dodanie/odjęcie kwoty w aktualnej walucie (`add`).
   - Dodanie kwoty w po przecliczeniu z obcej waluty.
   - Zamianę na inną walutę używając klasy `Office` (`void change(String newCurrenct)`).
   - Przeciążyć metodę `equals`.

   Napisać odpowiednie konstrukotry.

5. Do wszystkich plików źródłowych dodać komentarze, wykorzystywane przez `javadoc`.

6. Klasy umieścić w odpowiednim pakiecie.

2 Java Logging

Do klasy `Money` dodać logowanie.

1. Zaimportować pakiet `java.util.logging`.
2. Dodać prywatne pole statyczne typu `Logger` i utworzyć objekt o nazwie `"MoneyLog"`.
3. Dodać blok statyczny `static{...}`, a w nim utworzyć handler typu `FileHandler`, eksportujący logi to pliku `"log.txt"`.
4. Ustawić poziom handlera na `ALL` i formatowanie typu `SimpleFormatter`.
5. Dodać handler do loggera.
6. Ustawić poziom loggera na `Level.FINE`.
7. Przechwycić odpowiednie wyjątki.
8. Utworzyć własny plik konfiguracyjny `logging.properties` i podmienić go opcją `-Djava.util.logging.config.file` podczas testowania w ostatnim zadaniu.

3 build.xml

Napisać skrypt *Apache Ant* `build.xml` w języku *XML*, który automatycznie kompiluje powyższe klasy i tworzy podpisane własnym kluczem archiwum `.jar`.

4 MyPrint

Napisać własne zadanie *Apache Ant*, które wypisuje zawartość pliku tekstowego i numeruje wiersze.

1. Napisać klasę `MyPrint`, która dziedziczy po klasie `org.apache.tools.ant.Task`
2. Zaimportować odpowiednie klasy.
3. Zadanie przyjmuje argumenty `name` (nazwa pliku do wydruku lub objekt `java.io.File`) oraz `numerate` typu `boolean` (wypisuje numery linii gdy `true`).
4. Zaimplementować metodę `public void execute()`.
5. Zaimplementować odpowiednie metody `set*`.
6. Dodać element `lines`, który posiada argumenty `min` i `max` oznaczające zakres wypisywanych linii.
7. Zaimplementować klasę `Lines` i metody `createLines, addLines`.
8. W pliku `build.xml` dopisać *cel*, który wykonuje zadanie `MyTask`.

Uwaga! Przy kompilacji klasy, należy dołączyć archiwum `ant.jar`, w którym znajduje się pakiet `org.apache.tools.ant`. W poszukiwaniu archiwum może pomóc komenda `locate ant.jar`.

5 MoneyTest

1. Pobrać archiwum *JUnit* z `www.junit.org`.
2. Napisać klasę `MoneyTest`, która rozszerza klasę `TestCase` i testuje klasę `Money`.
3. Zaimportować odpowiednie klasy z pakietu `junit.framework` oraz klasę `org.junit.Assert`.
4. Napisać konstruktor, który wywołuje konstruktor klasy nadrzędnej z argumentem typu `String`.

5. W szczególności przetestować konstruktory, metody `get*, add` i `change`.
6. Napisać funkcję `public static Test suite()` i zdefiniować zbiór testów.
7. Skompilować klasę:

   ```
   javac MoneyTest.java -cp junit-4.5.jar:.
   ```

8. Uruchomić testy:

   ```
   java -cp junit-4.5.jar:. junit.textui.TestRunner MoneyTest
   ```

Andrzej Görlich
atg@th.if.uj.edu.pl