Programowanie Obiektowe
Pliki
Zadanie oceniane nr 3c
29-05-2022

W dniu dzisiejszym komponenty do zadania znajdują się w Państwa repozytorium (katalog zadanie_oceniane_3c/resources). Po zakończeniu pracy konieczne jest wgranie zmian do repozytorium (add + commit + push) w katalogu o nazwie w stylu: zadanie_oceniane_3c. Fakt wgrania plików do swojego repozytorium można sprawdzić samodzielnie logując się (via www) na swoje konto i sprawdzając czy pojawiły się tam wszystkie zmiany.

"Asteroidy"

Dzisiaj zajmiemy się parsowaniem i prztwarzaniem opisów asteroidó. Tradycyjnie, należy wczytać zawartość pliku zawierającego tym razem ogólnodostępne informacje o wspomnianych obiektach astralnych, przeprocesować uzyskane dane i dostarczyć implementację wymaganych funkcjonalności towarzyszących.

Prace do wykonania:

1. **Zastanowić się nad naturą danych, które przyjdzie Państwu parsować.**
   Plik asteroidy.txt zawiera informacje o dostrzeżonych i zarejestrowanych przez NASA asteroidach z których każda zajmuje jedną linijkę. Poszczególne składowe oddzielone są sekwencją znaków: ___. Opisy schematu danych znajdują się poniżej:

   id____nazwa____średnica (max—min)____zagrożenie dla ziemi ("Ja" – tak, "Nej" – nie)____data bliskiego przejścia____prędkość____dystans____orbituje w okół (ZIEMIA/INNE/BRAK_DANYCH)___link____

   Przykład:
   `2162162____162162 (1999 DB7)____(0.5834988155--0.2609486033)____true____2015-09-11____8.5598624685____63895756.445236749____Earth____http://api.nasa.gov/neo/rest/v1/neo/2162162____`

2. **Model danych**
   Stworzyć klasy których instancje przechowują dwa różne rodzaje informacji (ogólna oraz szczegółowa) o dostrzeżonych asteroidach zaczerpnięte z wierszy pliku. Informacja ogólna zawiera tylko id, nazwę i średnicę, natomiast szczegółowa zawiera wszystkie wymienione powyżej. Dla każdej wyizolowanej z danego wiersza danej (a dane są rozdzielone za pomocą "____") powinno być stworzone tylko jedno pole odpowiedniego typu. Średnica ma być dedykowanym typem zawierającym dziesiętne wartości min i max (a nie Stringiem), a jej reprezentacja tekstowa wypisuje te wartości z dokładnością do dwóch miejsc po przecinku. Trzeba mieć na uwadze, żeby obiekty klas opisujących informacje szczegółowe i ogólne można było wrzucić do jednej kolekcji. Kryteria tego czy dany wiersz powinien być wczytany jako informacja ogólna lub szczegółowa znajdują się w kolejnym punkcie. Prędkość i dystans powinny być zakorąglone do jednego miejsca po przecinku.

   Należy pamiętać że:
   - zakładamy że struktura danych w pliku i ich typy są poprawne
   - to czy dana informacja przechowywana przez obiekt danego typu jest **równa** innej zależy od tego czy ich id są takie same.
   - jedna linijka z pliku = jedna asteroida
   - poprawne zasięgi, odpowiednia hierarchia, elementy statyczne (jeśli wypada ich użyć) to sprawy tutaj oczywiste

3. **Klasa AstroParser**
   - Zawiera metodę parseReliableAsteroidInformation(namiary na plik) zwracająca kolekcję wczytanych informacji, w której są one na bieżąco ustawiane w porządku malejącym względem id. To czy informacja jest ogólna czy szczegółowa zależy od tego czy asteroida zagraża Ziemi (jeśli zagraża to obiekt jest typu zawierającego pełe informacje). Jeśli minimalna średnica jest większa od maksymalnej (w przypadku asteroid które zagrażają Ziemi), to znaczy że coś tu nie gra i jest to sytuacja uniemożliwiająca dalszą pracę tej metody zgodnie z deklarowanym kontraktem, ponieważ oczekuje on rzetelnych danych. Powinna być ona w dobrze znany sposób przerwana.
   - Plik asteroidy.txt ma znajdować się w projekcie w katalogu "from_nasa" ale poza katalogiem "src". Dostęp do niego ma być względny i **niezależny** od miejsca uruchomienia aplikacji (working dir). Jest zakodowany w UTF-8.

4. **Klasa AsteroidProcessor**
   - Zawiera metodę getDangerousAsteroidsOnly(kolekcja obiektów zwróconych przez poprzednią metodę), która z przekazanej kolekcji wyizolowuje obiekty zawierające informacje szczegółowe. Metoda zwraca listę w której zachowana jest kolejność wyizolowanych informacji ale nie ma duplikatów.

5. **Klasa MyUnforgvingBufferedReader**
   Stworzyć klasę MyBufferedReader mającą wszystkie fukcjonalności klasy BufferedReader. Dodatkowo wystawia ona metodę myReadLine() która robi to samo co metoda readLine(), jednak w przypadku natrafienia na duplikat danej linijki (String, który już wcześniej był przez myReadLine() lub readLine() zwrócony) zwraca ją oraz wypisuje ją na konsoli dopisując ile razy takowa linikja tekstu od momentu uruchomienia aplikacji została już zwrócona (przez obie wspomniane metody).