# Programowanie Obiektowe
# Pliki
Zadanie oceniane nr 3b
23-05-2022

**W dniu dzisiejszym komponenty do zadania znajdują się w Państwa repozytorium (katalog zadanie_oceniane_3b/resources).** Po zakończeniu pracy konieczne jest wgranie zmian do repozytorium (add + commit + push) w katalogu o nazwie w stylu: zadanie_oceniane_3b. Fakt wgrania plików do swojego repozytorium można sprawdzić samodzielnie logując się (via www) na swoje konto i sprawdzając czy pojawiły się tam wszystkie zmiany.

# "Czarter"

Oferty czarterowe będą biznesowym tematem przewodnim dzisiejszego zadania. Tradycyjnie, należy wczytać zawartość pliku zawierającego tym razem ogólnodostępne informacje o ofertach czarterowych, przeprocesować uzyskane dane i dostarczyć implementację wymaganych funkcjonalności towarzyszących.

## Prace do wykonania:

1. **Zastanowić się nad naturą danych, które przyjdzie Państwu parsować.**
   Plik oferty_czarterowe.txt zawiera informacje o trzech rodzajach ofert czarteru jednostek pływających z których każda zajmuje jedną linijkę. Poszczególne składowe oddzielone są sekwencją znaków: **----**. Opisy schematu danych znajdują się poniżej:

   **rodzaj----id----model----port----lokalizacja (długość i szer. geograficzna)----cena----przedpłata----**

   Przykład:
   ```
   Jacht----5fdb4017e8fb4d040b0b0895----Maxus 33.1----Port AZS Wilkasy----
   54.008645;21.738013----639.87----50----
   ```

   **rodzaj----id----model----port----lokalizacja (długość i szer. geograficzna)----cena----moc silników----poimprezowy?----**

   Przykład:
   ```
   House boat----638dca94b6bc8a517f07c37a----30----Port Tło dla Mew----
   53.963983;21.757142----494.23----50----N----
   ```

   **rodzaj----id----model----port----lokalizacja (długość i szer. Geograficzna)----cena----moc silników----**

   Przykład:
   ```
   MOTORÓWKA----62bab2dfe8ee666341446cb7----Galia 700----Marina Gdynia----
   54.517676;18.54999----786.79----250----
   ```

2. **Model danych**
   Stworzyć klasy których instancje przechowują informacje o trzech rodzajach ofert zaczerpnięte z wierszy pliku (każda z nich to inny typ). Dla każdej wyizolowanej z danego wiersza danej (a dane są rozdzielone za pomocą "----") powinno być stworzone tylko jedno pole odpowiedniego typu. Lokalizacja ma być dedykowanym typem a nie Stringiem, a jej reprezentacja tekstowa dokłada do szerkości geograficznej literkę "N", a do długości "E". Trzeba mieć na uwadze, żeby obiekty klas opisujących wszystkie rodzaje ofert można było wrzucić do jednej kolekcji. Znaczniki "Jacht", "House boat" i "MOTORÓWKA" służą do rozpoznania typu w pliku, więc docelowo nie są przechowywane przez obiekty.

   **Należy pamiętać że:**
   - zakładamy że struktura danych w pliku i ich typy są poprawne
   - to czy zdarzenie przechowywane przez obiekt danego typu jest **równe** innemu zależy od tego czy ich id są takie same.
   - jedna linijka z pliku = jedna oferta
   - poprawne zasięgi, odpowiednia hierarchia, elementy statyczne to sprawy tutaj oczywiste

3. **Klasa OffersParser**
   - Klasa zawiera metodę parseNonDuplicatedOffers(namiary na plik) zwracająca kolekcję wczytanych zdarzeń, w której są one poustawiane w kolejności w jakiej były parsowane. Ponowne pojawienie się identycznej linijki tekstu (podczas wczytywania danych przez metodę) jak ta która kiedyś już była procesowana (w czasie życia obiektu klasy OffersParser) jest sytuacją uniemożliwiającą dalszą pracę tej metody, ponieważ oczekuje ona niezduplikowanych danych. Powinna być ona z dobrze znany sposób przerwana.

- Plik oferty_czarterowe.txt ma znajdować się w projekcie w katalogu "my_offers" ale poza katalogiem "src". Dostęp do niego ma być względny i **zależny** od miejsca uruchomienia aplikacji (working dir). Jest zakodowany w UTF-8.

4. **Klasa OffersProcessor**
   - Zawiera metodę getAtMostFourSelectedHouseBoatOffers(kolekcja zdarzeń z poprzedniego punktu), która z przekazanej kolekcji wyizolowuje oferty na house boaty które nie są poimprezowe. Oferty w zwróconej kolekcji powinny być ułożone w odwrotnej kolejności niż zostały wczytane i powinno być ich co najwyżej 4.

5. **Klasa MyCleverFile**
   - Posiada te same funkcjonalności co klasa File z tą różnicą, że jej konstruktor przyjmujący jeden argument (Stringa z namiarami na plik z rozszerzeniem) robi to samo co konstruktor klasy File oraz w przypadku gdy plik (na którego namiary zostały przekazane jako argument) nie istnieje:
     1) szuka w katalogu (który musi sam ustalić) w którym ten plik powinien być innych plików zaczynających się nazwą tego pliku (bez rozszerzenia).
     2) jeśli takowe znajdzie to wypisuje na konsoli informację "Plik nie istnieje, czy może chodziło Ci o... " i listuje to co został znalezione.