5 maja 2021

Programowanie obiektowe

Lista 10.

Poniższa lista zadań jest do zrobienia w Ruby. Każde zadanie to 4 punkty. Wykonaj dwa z tych zadań.

Zadanie 1
Popularne algorytmy sortowania kolekcji wymagają kilku elementarnych operacji na kolekcji takie jak `swap(i, j)` zamieniające miejscami elementy, `length()` wracająca długość kolekcji, czy `get(i)` zwracająca i-ty element kolekcji.
Zaimplementuj dwie klasy:

1. ***Collection*** implementująca dowolną kolekcję wraz z wyżej wymienionymi metodami (ew. jeszcze dodatkowe, jeśli będzie taka potrzeba);
2. ***Sorter*** implementującą dwa wybrane algorytmy (lub strategie) sortowania kolekcji. Klasa ta powinna implementować metody `sort1(kolekcja)` i `sort2(kolekcja)`, których argumentem jest obiekt klasy ***Collection***. Która metoda jest szybsza? Zaznacz to w komentarzu w pliku źródłowym

Ważne w tym zadaniu jest to, aby metoda korzystała wyłącznie z metod udostępnianych przez kolekcję i nie odwoływała się w żaden sposób do implementacji ***Collection***.

Zadanie 2
Zaprogramuj klasę ***Kolekcja*** implementującą kolekcję przechowującą elementy dowolnego typu w kolejności rosnącej. Lista ta powinna być zaprogramowana jako lista dwukierunkowa. Zaprogramuj również klasę ***Wyszukiwanie*** wraz z dwiema różnymi metodami wyszukiwania elementu w kolekcji. Dobrymi kandydatami są tu: algorytm wyszukiwania binarnego oraz jego wariant: wyszukiwanie interpolacyjne. Podobnie jak w poprzednim zadaniu, kolekcja powinna być argumentem wywołania metody, a metoda powinna być tak napisana, aby nie korzystała ze szczegółów implementacji kolekcji.

Zadanie 3
Zadanie polega na implementacji klasy (a właściwie kilku klas) służącej do zapamiętania w czytelnej postaci chwilowego stanu programu, tj. stanu obiektów istniejących w systemie. Chcemy, aby była możliwość zapisania stanu w pliku zarówno w formacie html, jak i formacie *reStructuredText*.
Zaprogramuj:

1. klasę ***Snapshot*** z metodą `run`, która zbiera informacje o stanie programu, tj. przynajmniej stan zmiennych lokalnych i listę istniejących obiektów;
2. dwie lub więcej klas formatujących w czytelną postać informacje zebrane przez ***Snapshot***, powiedzmy ***HtmlFormatter*** i ***ReFormatter***. Ważne jest, by te klasy implementowały metody o tych samych nazwach.

Są dwie możliwe realizacje: w jednej ***HtmlFormatter*** i ***ReFormatter*** są podklasami ***Snapshot***, w drugiej instancja klasy ***HtmlFormatter*** bądź ***ReFormatter*** są argumentami `Snapshot.run`.

To już ostatnia lista zadań. Na SKOS'ie są informacje o projekcie.

Marcin Młotkowski