7 kwietnia 2022

Programowanie obiektowe

Lista 6.

Poniższe zadania mają być zaimplementowane w Javie. Dla każdego zadań proszę podać krótki program ilustrujący możliwości zaimplementowanych klas.

Za każde zadanie można otrzymać do 4 pkt, jednak można oddać nie więcej niż 2 zadania.

Zadanie 1

Wybierz dowolne zadanie z poprzednich list dot. kolekcji (listy, grafy itp) bądź zadanie 3 z tej listy i zaprogramuj je w Javie. Wymuś, aby implementowana kolekcja implementowała interfejs ***Serializable*** (z pakietu `java.io`) tak, aby można było zapisywać i odczytywać kolekcję z pliku dyskowego.

Jako ilustrację programu podaj program który zapisuje kolekcję na dysku a następnie ją odzyskuje.

Można skorzystać z dostępnych w internecie artykułów opisujących jak implementować ten interfejs.

Zadanie 2

Wybierz dowolną klasę z poprzednich list dot. kolekcji (listy, grafy itp). Dodaj do tej klasy deklarację implementacji interfejsu ***Collection<E>*** z `java.utils` i zaprogramuj metody wskazane w interfejsie.

Poszukaj informacji, jakie korzyści daje implementacja tego interfejsu i zaprezentuj te korzyści w przykładach.

Zadanie 3

Zaprogramuj klasę implementującą dostęp do bufora o stałym rozmiarze prze- chowującym elementy typu generycznego `T`. Rozmiar bufora jest stały, zadawany w konstruktorze. Implementacja powinna umożliwić działanie takiego bufora w środowisku wielowątkowym (*thread–safe*). Kolejność elementów pobieranych z bufora powinna być taka sama jak kolejność ich wkładania do bufora.

Korzystając z tej klasy zaimplementuj problem *producenta–konsumenta*[1]: producent produkuje wyniki (napisy) i wkłada je do bufora. Jeśli bufor jest pełny, to producent zasypia czekając aż zwolni się miejsce w buforze. Konsument, jeśli w buforze jest jakiś element (napis), to go pobiera i „konsumuje". Zaimplementuj producenta i konsumenta jako dwa odrębne wątki. Przed implementacją tego zadania warto zapoznać się z metodami `wait()` i `notify()` klasy ***Thread***.

Zadanie 4

Algorytm sortowania tablicy elementów przez scalanie działa następująco: najpierw tablica jest dzielona na pół. Następnie każda z tych mniejszych tablic jest porządkowana. Na końcu obydwie posortowane tablice są scalane.

Zaprogramuj sortowanie przez scalanie tablicy elementów implementujących interfejs ***Comparable*** tak, aby operacje sortowania podtablic były odrębnymi wątkami.

Marcin Młotkowski

[1] https://en.wikipedia.org/wiki/Producer-consumer_problem