

# Programowanie obiektowe

## Lista 3.

**Zadanie 1.** Zaprogramuj klasę *Lista* $\langle T \rangle$  implementującą metody dodawania i usuwania elementów z początku i końca listy, oraz metodę sprawdzania jej niepustości. Istotne jest, aby elementy listy nie były obiektami klasy *Lista*, lecz elementami innej klasy, której polami są: pole zawierające wartość typu  $T$ , oraz odnośniki do innych elementów listy. Przyjmij taką implementację klasy *Lista*, aby działała ona efektywnie zarówno gdy jest wykorzystywana jako kolejka, jak i stos, tj. aby operacje dodawania i usuwania elementów na początek i koniec działały w czasie stałym. Operacja (metoda) usuwania elementu powinna zwracać jako wartość usuwany element.

**Zadanie 2.** Zaimplementuj klasę *Słownik* $\langle K, V \rangle$  przechowującą pary elementów, gdzie pierwszym elementem pary jest klucz, a drugim wartość. Klasa powinna implementować metodę dodawania, wyszukiwania i usuwania elementu wskazywanego przez klucz.

**Zadanie 3.** Na wykładzie został omówiony wzorzec *Singleton*, który pozwala na utworzenie tylko jednej instancji klasy. Zaprogramuj klasę *TimeNTon*, która będzie działała w następujący sposób:

- w godzinach pracowni tworzy co najwyżej  $N$  instancji klasy,  $N$  jest ustaloną w kodzie źródłowym stałą. Przyjmij, że jeżeli zostanie utworzonych  $N$  instancji, to kolejne żądania obiektu zwrócą kolejne istniejące już instancje klasy;
- poza godzinami pracowni zawsze zwracany jest ten sam obiekt. Wcześniej "wydanych" obiektów nie trzeba usuwać.

Zaprogramuj klasę w wersji leniwej.

**Zadanie 4.** Zaprogramuj klasę *Wektor* implementującą wektory swobodne. Przyjmij, że współrzędne wektora są pamiętane za pomocą liczb typu *float*. Zaprogramuj operatory dodawania wektorów oraz iloczynu skalarnego wektorów i iloczynu wektora przez liczbę.

Korzystając z tej klasy zaprogramuj klasę *Macierz* jako tablicę wektorów z operacjami dodawania macierzy, mnożenia i mnożenia przez wektor.

Zastanów się, czy jest możliwa implementacja tych klas jako klas generycznych, tak aby np. można było łatwo implementować wektory lub macierze liczb zespolonych.

## Dodatkowe informacje

- Implementacje klas skompiluj w postaci modułów dll. Do każdego zadania dołącz też krótki przykładowy program ilustrujący wykorzystanie zbudowanej biblioteki. Odpowiednie informacje jak to zrobić można znaleźć np. w dokumentacji polecenia *csc* (Windows) lub *mcs* (Mono).
- Powyższe zadania należy wykonać nie wykorzystując klas bibliotecznych.

Za każde zadanie można otrzymać do 4 pkt, jednak można oddać nie więcej niż 2 zadania. Proszę do każdego ocenianego zadania dołączyć króciutki program ilustrujący możliwości zaprogramowanych klas. Zadania należy zaprogramować w C#.

**Przypomnienie:** Rozwiązanie proszę wrzucić do serwisu SKOS do najbliższej środy do godz. 8:00