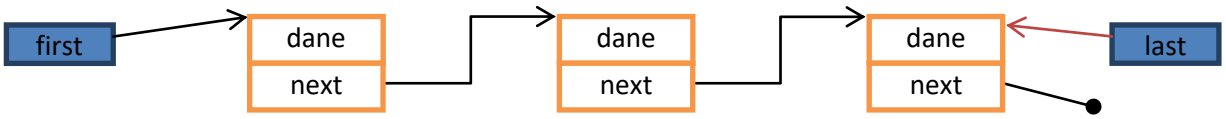
Algorytmy i złożoność obliczeniowa

Laboratorium 4: Listy jednokierunkowe i dwukierunkowe.

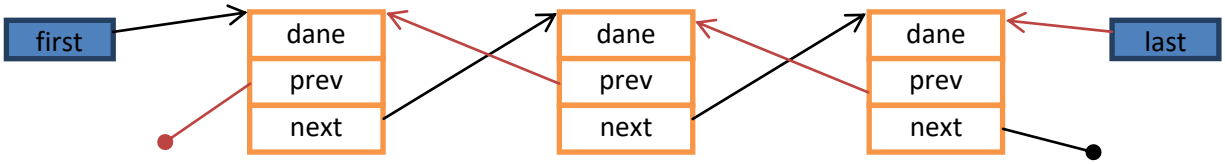
1. **Koncepcja list jednokierunkowych**

   Lista jednokierunkowa jest strukturą danych pozwalającą na pamiętanie danych w postaci uporządkowanej, a także na bardzo szybkie wstawianie i usuwanie elementów do i z listy. Lista zapisana jest w postaci elementów zawierających porcję danych oraz wskaźnik (adres) następnego elementu. W przypadku języka Java tak naprawdę mamy do czynienia z referencjami do obiektów. W ten sposób wystarczy pamiętać referencję do pierwszego elementu listy, by pamiętać całą listę. Strukturę listy jednokierunkowej ilustruje poniższy rysunek:



2. **Listy dwukierunkowe.**

   Listy dwukierunkowe to uporządkowane, liniowe struktury danych. Różnią się od list jednokierunkowych tylko tym, że nawigacja pomiędzy elementami listy może odbywać się w dwóch kierunkach. Typowa lista dwukierunkowa składa się z elementów, które oprócz danych przechowywanych w liście zawierają informacje o poprzednim i następnym elemencie listy. W zależności od języka programowania używanego do implementacji list odwołania do sąsiednich elementów to zazwyczaj wskaźniki lub referencje. Strukturę listy dwukierunkowej ilustruje poniższy rysunek:



3. **Klasy generyczne.**

   Klasy generyczne są to klasy o parametryzowanych typach danych. Klasy generyczne posiadają kompletną implementację, jednak nie definiują typów danych wykorzystanych w tej implementacji. Np.:

```
public class Element<T> {
    private T dane;
    private Element<T> next, prev;
}
```

   Parametrem klasy Element jest typ T. Możemy teraz używać instancji tej klasy do przechowywania danych różnego typu. Np. (po zdefiniowaniu odpowiedniego konstruktora):

```
Element<Integer> e = new Element<Integer>(5);
Element<String> e2 = new Element<String>("Ala ma kota");
```

   Od wersji 7 kompilator języka Java posiada mechanizm dedukcji typów generycznych (*type inference*), dzięki czemu nie musimy za każdym razem powtarzać parametrów (typów). W związku z czym następujące wyrażenie jest poprawne:

```
Element<Integer> n = new Element<>(5);
```

   Na dzisiejszych zajęciach należy wykorzystać mechanizmy programowania generycznego dostępne w języku Java. **Uwaga!** Typy podawane jako parametry muszą być typami **referencyjnymi**!

4. **Klasy i interfejsy potrzebne do rozwiązania zadań.**

   Archiwum zawierające klasy i interfejs należy pobrać bezpośrednio ze strony WWW.
   Dostępne są dwie klasy: *ElemOne* i *ListException* oraz interfejs *IList*.

5. **Zadania.**

   1. Utworzyć nowy projekt Java w środowisku NetBeans lub Eclipse. Utworzyć pakiet *aiz.list.* Pobrać ze strony WWW klasy *ElemOne* i *ListException* oraz interfejs *IList* i umieścić je w pakiecie *aiz.list*. W pakiecie tym utworzyć klasę ***ListOne*** implementującą interfejs *IList*, a następnie korzystając z opcji *„generate abstract methods”* wygenerować szkielet wszystkich metod interfejsu. **[1p]**

   2. Zaimplementować wszystkie metody klasy ***ListOne*** realizujące interfejs *IList.* Klasa ta ma być generyczną listą jednokierunkową wykorzystującą obiekty klasy *ElemOne* jako elementy listy.
   Wskazówka: w klasie *ListOne* zadeklaruj pola *first* i *last* – referencje do pierwszego i ostatniego elementu listy, a także pole *count* – licznik elementów. **[3p]**

   3. Napisać program wykorzystujący wszystkie metody klasy ***ListOne***, który zaprezentuje ich działanie. **[1p]**

   4. Utworzyć klasę reprezentującą element listy dwukierunkowej. Zaimplementować listę dwukierunkową implementującą interfejs *IList*. Napisać program testujący działanie listy dwukierunkowej. **[3p]**

   5. Zaimplementować metodę ***IList<T> join(IList<T> druga)*** dokonującą złączenia dwóch list i zwracającą nową listę. Wykonaj kopiowanie elementów (deep copy), tak żeby nowa lista była całkowicie niezależna od list wejściowych. **[1p]**

   6. Zaimplementować metodę ***boolean similar(IList<T>)***, która sprawdza czy podana lista jest podobna (do tej, na rzecz której metoda jest wywołana). Metoda ma zwracać **true** tylko wtedy, gdy jedna lista jest permutacją drugiej (obie zawierają tyle samo takich samych elementów ale w dowolnej kolejności).**[1p]**

Praca domowa:

1. Napisać program wczytujący z pliku tekstowego do listy tylko liczby parzyste.

2. Napisać program wczytujący liczby z pliku tekstowego do listy dwukierunkowej i sumujący co trzecią liczbę.
3. Napisać program wczytujący do listy liczby całkowite i obliczający:
   a. element maksymalny i minimalny,
   b. sumę,
   c. średnią arytmetyczną,
   d. odchylenie standardowe.

4. Napisać program wczytujący liczby z pliku lub z klawiatury w taki sposób, że w każdym momencie działania programu lista jest posortowana.
5. Zaimplementować metodę *iterator()* zwracającą obiekt implementujący Iterator<T>. https://docs.oracle.com/javase/7/docs/api/java/util/Iterator.html

Na następnych zajęciach:

Grafy. Reprezentacje grafów. Algorytmy grafowe.