Wyższa Szkoła Bankowa
Gdańsk Gdynia

Analiza danych w języku Java I

Zajęcia 8 - JCF

Anna Pakeizer

Data

Java Collections Framework

Linki:

[1] https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/package-tree.html

[2] https://docs.oracle.com/javase/tutorial/collections/

[3] https://docs.oracle.com/javase/tutorial/collections/intro/index.html

[4] https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/packagesummary.html#CollectionsFramework

JCF

W wielu językach programowania istnieję swoiste biblioteki gotowych do wykorzystania konkretnych realizacji abstrakcyjnych struktur danych.

W Javie nazywa się to Java Collections Framework.

Jakie korzyści daje nam JCF

- brak konieczności definiowania rozmiaru ilości elementów w trakcie pisania programu;
- zmniejsza wysiłek związany z programowaniem zapewniając przydatne struktury danych i algorytmy;
- zwiększa szybkość i jakość programu;
- umożliwia interoperacyjność między niepowiązanymi interfejsami API;
- zmniejsza wysiłek związany z nauką i używaniem nowych interfejsów API;
- zmniejsza wysiłek związany z projektowaniem nowych interfejsów API;
- ułatwia ponowne wykorzystanie oprogramowania.

Co to jest kolekcja?

"*A collection — sometimes called a container — is simply an object that groups multiple elements into a single unit.*"

Kolekcja – czasami zwana kontenerem – jest po prostu [jednym] obiektem, który grupuje wiele elementów w pojedynczej jednostce.

Czym jest framework?

Frame to rama. Rama ogranicza pewien obszar. Jest to pewne ograniczenie. Work to praca, a więc framework to praca w określonych ramach. Framework coś nam daje co ułatwia nam pracę, ale jednocześnie zabiera trochę wolności ograniczając nasze działania do określonych ram.

Frameworki to zbiory napisanych funkcjonalności, które możemy natychmiast użyć bez potrzeby własnej implementacji. Frameworki dotyczą zwykle określonych obszarów, np. tworzenia aplikacji sieciowych.

Frameworki wymuszają programowanie w określony sposób, ale zaletą takiego podejścia jest to, że dzięki temu zyskujemy lepsze zrozumienie kodu napisanego przez innego programistę oraz dostajemy wsparcie społeczności, np. https://stackoverflow.com/.

Collections framework

Collections framework jest ujednoliconą architekturą do prezentacji kolekcji oraz ich modyfikacji.

Framework kolekcji zawiera następujące elementy:

- interfejsy
- implementacje
- algorytmy

Podstawowe interfejsy w JCF

- Set
- List
- Queue
- Map

Przykład 1

List: https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/List.html

ArrayList: https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/ArrayList.html

<> nawiasy kątowe

Class ArrayList<E>

java.lang.Object
    java.util.AbstractCollection<E>
        java.util.AbstractList<E>
            java.util.ArrayList<E>

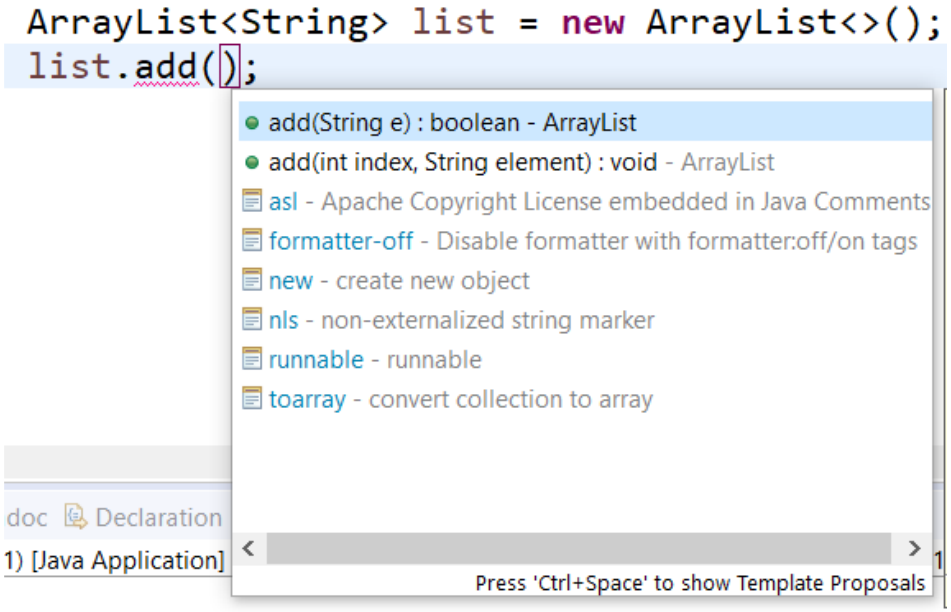
Type Parameters:

E - the type of elements in this list

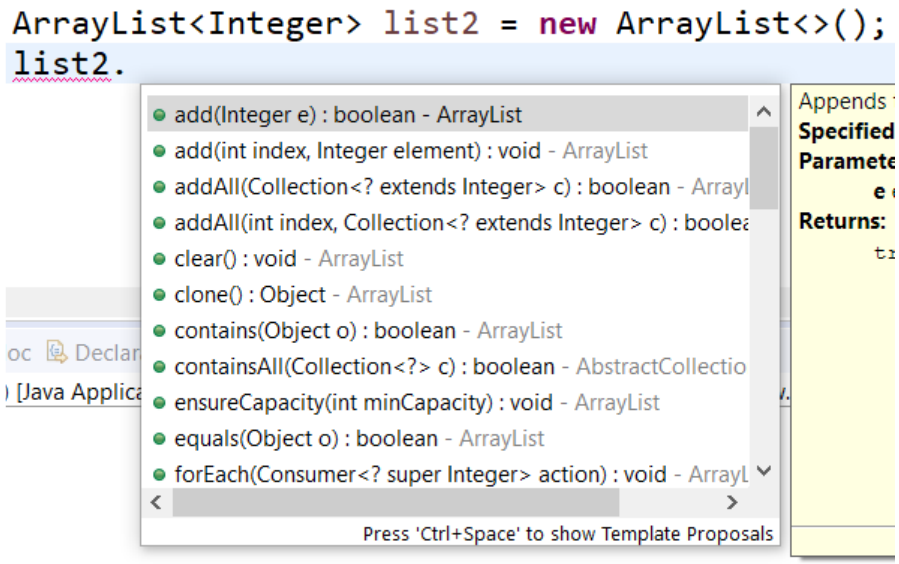
```
ArrayList<String> list = new ArrayList<>();
System.out.println(list);
//  []
ArrayList<Integer> list2 = new ArrayList<>();
System.out.println(list2);
//  []
```

Przykład 2

- boolean add (E e)
- dodaje element *e* o typie *E* na koniec listy i zwraca prawdę lub fałsz w zależności od rezultatu dodawania.
- STS podpowiada zgodnie z określonym typem w definicji ArrayList'y



Przykład 2 c.d.




```
        ArrayList<String> lista = new ArrayList<>();
//      lista.add("a");
//      System.out.println(lista);
//      String x = "b";
//      lista.add(x);
//      System.out.println(lista);
//      System.out.println(lista.add("c")); // true
//      System.out.println(lista);
```

Zadanie 1

Przykład 3

void add(int index, E element)

dodaj element *element* o typie *E* na określonej pozycji *index*.

```
ArrayList<String> lista = new ArrayList<>();
//        lista.add(0, "a");
//        System.out.println(lista);
//        lista.add(100, "b");
//        System.out.println(lista);
```

```
        ArrayList<String> lista = new ArrayList<>();
//        lista.add("a");
//        lista.add("c");
//        System.out.println(lista);
//        lista.add(1, "b");
//        System.out.println(lista);
```

Zadanie 2

Przykład 4

boolean addAll(Collection<? extends E> c)

- dodaj wszystkie elementy z kolekcji *c* (np. ArrayList) do mojej listy;
- to co dodaję musi implementować interfejs Collections;
- ? zapytania oznacza typ, którego nazwy nie znamy, bo sami go sobie mogliśmy wymyślić, jednak ten typ musi dziedziczyć po typie elementów listy do której chcemy dodać nowe elementy, może to być też ten sam typ, np. String.

Module java.base

Package java.util

Class ArrayList<E>

java.lang.Object
    java.util.AbstractCollection<E>
        java.util.AbstractList<E>
            java.util.ArrayList<E>

Type Parameters:

E - the type of elements in this list

All Implemented Interfaces:

Serializable, Cloneable, Iterable<E>, Collection<E>, List<E>, RandomAccess

Direct Known Subclasses:

AttributeList, RoleList, RoleUnresolvedList

```java
ArrayList<String> lista = new ArrayList<>();
lista.add("a");
lista.add("b");
lista.add("c");

ArrayList<String> lista2 = new ArrayList<>();
lista2.add("d");
lista2.add("e");
lista2.add("f");

lista.addAll(lista2);
System.out.println(lista);

// [a, b, c, d, e, f]
```

Przykład 4 c.d.

```
public class Main {

    public static void main(String args[]) {

        ArrayList<Zwierze> al = new ArrayList<>();
        Zwierze z1 = new Zwierze();
        al.add(z1);
        System.out.println(al);

        ArrayList<Pies> alPies = new ArrayList<Pies>();
        Pies p1 = new Pies();
        alPies.add(p1);

        al.addAll(alPies);
        System.out.println(al);

    }

}

class Zwierze {
    int wiek;
}

class Pies extends Zwierze {

}
```

Zadanie 3

Przykład 5

boolean addAll(int index, Collection<? extends E> c)

dodaje elementy kolekcji *c* do mojej listy zaczynając od określonej pozycji *index*.

```
ArrayList<String> al1 = new ArrayList<String>();
al1.add("a");
al1.add("d");
System.out.println(al1);

ArrayList<String> al2 = new ArrayList<String>();
al2.add("b");
al2.add("c");

al1.addAll(1, al2); // al1.addAll(0, al2);
System.out.println(al1);

// al2.set(0, "x");
// System.out.println(al1);
// System.out.println(al2);
```

Zadanie 4

Przykład 6

void clear()

```
ArrayList<String> al1 = new ArrayList<String>();
al1.add("a");
al1.add("b");
System.out.println(al1);

al1.clear();


System.out.println(al1);
```

Zadanie 5

Przykład 7

boolean contains(Object o)

sprawdź czy istnieje obiekt w liście

```
ArrayList<String> al1 = new ArrayList<String>();
al1.add("a");
al1.add("b");
System.out.println(al1);

System.out.println(al1.contains("a"));
System.out.println(al1.contains("x"));
```

Zadanie 6

Przykład 8

E get(int index)

Pobierz element z listy znajdującego się na danej pozycji.

```
ArrayList<String> al1 = new ArrayList<String>();
al1.add("a");
al1.add("b");
System.out.println(al1);

String z = al1.get(0);
System.out.println(z);
```

Przykład 9

int indexOf(Object o)

Na której pozycji znajduje się obiekt o który pytam.

```
ArrayList<String> al1 = new ArrayList<String>();
al1.add("a");
al1.add("b");
System.out.println(al1);

System.out.println(al1.indexOf("b"));
System.out.println(al1.indexOf("z"));
```

Zadanie 7

Przykład 10

boolean isEmpty()

Sprawdź czy lista jest pusta.

```
ArrayList<String> al1 = new ArrayList<String>();
al1.add("a");
al1.add("b");

ArrayList<String> al2 = new ArrayList<String>();

System.out.println(al1.isEmpty());
System.out.println(al2.isEmpty());
```

Zadanie 8

Przykład 11

Iterator<E> iterator()

https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/Iterator.html

Zadaniem iteratora jest przemieszczanie się po ciągu elementów

```
ArrayList<String> al1 = new ArrayList<String>();
//      al1.add("a");
//      al1.add("b");
//      al1.add("c");
//      al1.add("d");
        Iterator<String> it = al1.iterator();
//      System.out.println(it.hasNext()); // ustawiam się przed pierwszym
elementem i sprawdzam czy element pierwszy istnieje
//      System.out.println(it.next()); // zwróć element
//      System.out.println(it.hasNext());
//      System.out.println(it.next());
```

Przykład 11 c.d.

Nie ma potrzeby wiedzy o pozycjach, na których znajdują elementy w liście.

```
ArrayList<String> al1 = new ArrayList<String>();
al1.add("a");
al1.add("b");
al1.add("c");
al1.add("d");
Iterator<String> it = al1.iterator();
while (it.hasNext()) {
    String tmp = it.next();
    System.out.println(tmp);
}
```

Zadanie 9

Przykład 12

int lastIndexOf(Object o)

Zwraca pozycję ostatniego wystąpienia.

```
ArrayList<String> al1 = new ArrayList<String>();
al1.add("a");
al1.add("b");
al1.add("c");
al1.add("d");
al1.add("a");

System.out.println(al1);
System.out.println(al1.lastIndexOf("a"));
```

Przykład 13

E remove(int index)

Usuwa element na danej pozycji. Zwraca element, który został usunięty.

```
ArrayList<String> al1 = new ArrayList<String>();
al1.add("a");
al1.add("b");
al1.add("c");
al1.add("d");
al1.add("a");

System.out.println(al1.remove(4));
```

Przykład 14

boolean remove(Object o)

Usuwa pierwsze wystąpienie elementu w liście.

```
ArrayList<String> al1 = new ArrayList<String>();
al1.add("a");
al1.add("b");
al1.add("c");
al1.add("d");
al1.add("a");

System.out.println(al1.remove("a"));
System.out.println(al1);

ArrayList<Integer> al2 = new ArrayList<>();
al2.add(1);
al2.add(0);
System.out.println(al2.remove(0));
System.out.println(al2);
```

Zadanie 10

Przykład 15

boolean removeAll(Collection<?> c)

Usuwa z mojej listy wszsytkie te elementy, które sie znajdują w kolekcji *c*.

```
ArrayList<String> al1 = new ArrayList<String>();
al1.add("a");
al1.add("b");
al1.add("c");
al1.add("d");

ArrayList<String> al2 = new ArrayList<String>();
al2.add("d");
System.out.println(al1.removeAll(al2));
System.out.println(al1);
```

Zadanie 11

Przykład 16

E set(int index, E element)

Ustawia nową wartość na danej pozycji.

```
ArrayList<String> al1 = new ArrayList<String>();
al1.add("a");
al1.add("b");
al1.add("c");
al1.add("d");

System.out.println(al1.set(0, "z")); // zwraca wartość sprzed zmiany
System.out.println(al1);
```

Przykład 17

int size()

```
ArrayList<String> al1 = new ArrayList<String>();
al1.add("a");
al1.add("b");
al1.add("c");
al1.add("d");

System.out.println(al1.size());
```

Przykład 18

List<E> subList(int fromIndex, int toIndex)

Zwraca wycinek listy od fromIndex do toIndex.

fromIndex – włączone

toIndex - wyłączony

```
ArrayList<String> al1 = new ArrayList<String>();
al1.add("a");
al1.add("b");
al1.add("c");
al1.add("d");

System.out.println(al1.subList(1, 3)); // 1,2  - zawarte w podliście, 3 – nie
zawarty
```

Zadanie 12

Przykład 19

Object[] toArray()

Skopiuj elementy listy do tablicy obiektów.

```
    String[] tablica = new String[2];
        tablica[0] = "a";
        tablica[1] = "b";
        tablica[2] = "c";

//      List<String> al = new ArrayList<String>();
//      al.add("a");
//      al.add("b");
//      al.add("c");
//
//      Object[] tablica = new String[al.size()];
//      tablica = al.toArray();
//
//      for(Object s : tablica)
//          System.out.println(s);
```

Zadanie 13

Przykład 20

https://docs.oracle.com/javase/7/docs/api/java/util/Collections.html

sort – sortowanie

```
//      List<String> al = new ArrayList<String>();
//      al.add("a");
//      al.add("z");
//      al.add("c");
//
//      System.out.println(al);
//      Collections.sort(al);
//      System.out.println(al);
//
//      List<Integer> al2 = new ArrayList<>();
//      al2.add(3);
//      al2.add(2);
//      al2.add(1);
//
//      System.out.println(al2);
//      Collections.sort(al2);
//      System.out.println(al2);
```

Przykład 21

shuffle – tasowanie, przekładanie

```
List<Integer> al2 = new ArrayList<>();
al2.add(1);
al2.add(2);
al2.add(3);
al2.add(4);
al2.add(5);
al2.add(6);

System.out.println(al2);
Collections.shuffle(al2);
System.out.println(al2);
```

Przykład 22

reverse – odwróć kolejność

```
List<Integer> al2 = new ArrayList<>();
al2.add(1);
al2.add(2);
al2.add(3);
al2.add(4);
al2.add(5);
al2.add(6);

System.out.println(al2);
Collections.reverse(al2);
System.out.println(al2);
```

Przykład 23

rotate – przesuń o *x* miejsc

```
List<Integer> al2 = new ArrayList<>();
al2.add(1);
al2.add(2);
al2.add(3);
al2.add(4);
al2.add(5);
al2.add(6);

System.out.println(al2);
Collections.rotate(al2, 1);
System.out.println(al2);
```

Przykład 24

swap – zamiana miejscami

```
List<Integer> al2 = new ArrayList<>();
al2.add(1);
al2.add(2);
al2.add(3);
al2.add(4);
al2.add(5);
al2.add(6);

System.out.println(al2);
Collections.swap(al2, 1, 2);
System.out.println(al2);
```

Przykład 25

replaceAll – zamień wszystkie wystąpienia określonej wartości na inną

```
List<Integer> al2 = new ArrayList<>();
al2.add(1);
al2.add(1);
al2.add(1);
al2.add(4);
al2.add(5);
al2.add(6);

System.out.println(al2);
Collections.replaceAll(al2, 1, 2);
System.out.println(al2);
```

Przykład 26

fill – zamienia wszystkie elementy listy nową wartością

```
List<Integer> al2 = new ArrayList<>();
al2.add(1);
al2.add(2);
al2.add(3);
al2.add(4);
al2.add(5);
al2.add(6);

System.out.println(al2);
Collections.fill(al2, 9999);
System.out.println(al2);
```

Zadanie 14

Przykład 27

indexOfSubList – czy lista 2 zawiera się w liście 1? Jeśli tak, to od której pozycji?

```
List<Integer> al2 = new ArrayList<>();
al2.add(1);
al2.add(2);
al2.add(3);
al2.add(4);
al2.add(5);
al2.add(6);

System.out.println(al2);

ArrayList<Integer> al3 = new ArrayList<>();
al3.add(3);
al3.add(4);

System.out.println(Collections.indexOfSubList(al2, al3));
```

Przykład 28

lastIndexOfSubList – zwróć indeks od którego zaczyna się ostatnia znaleziona podlista

```
List<Integer> al2 = new ArrayList<>();
al2.add(1);
al2.add(2);
al2.add(3);
al2.add(4);
al2.add(3);
al2.add(4);
al2.add(5);
al2.add(6);
System.out.println(al2);

ArrayList<Integer> al3 = new ArrayList<>();
al3.add(3);
al3.add(4);

System.out.println(Collections.lastIndexOfSubList(al2, al3));
```

Zadanie 15

Przykład 29

Set

https://docs.oracle.com/javase/10/docs/api/java/util/Set.html

Zbiór nie może zawierać duplikatów.

boolean add(E e) – dodaj element e o typie E do zbioru

```
//        Set<String> s1 = new HashSet<>();
//        s1.add("a");
//        System.out.println(s1);
//        s1.add("a");
//        s1.add("a");
//        System.out.println(s1);
```

Przykład 30

boolean addAll(Collection<? extends E> c) – dodaj wszystkie elementy z innej kolekcji o ile elementy te już nie istnieją w obecnym zbiorze do którego dodajemy

```
ArrayList<String> al = new ArrayList<>();
al.add("a");
al.add("a");
al.add("a");
System.out.println(al);

Set<String> s1 = new HashSet<>();
s1.addAll(al);
System.out.println(s1);
```

Przykład 31

boolean contains(Object o) – sprawdź czy element znajduje się w zbiorze

```
Set<String> s1 = new HashSet<>();
System.out.println(s1.contains("a"));
s1.add("a");
System.out.println(s1.contains("a"));
```

Przykład 32

boolean containsAll(Collection<?> c) – sprawdź, czy ta kolekcja zawiera wszystkie elementy innej kolekcji

```
ArrayList<String> al = new ArrayList<>();
al.add("a");
al.add("b");
//      al.add("z");
Set<String> s1 = new HashSet<>();
s1.add("a");
s1.add("b");
s1.add("c");
System.out.println(s1.containsAll(al));
```

Przykład 33

boolean remove(Object o) – usuń element jeśli jest w zbiorze

```
Set<String> s1 = new HashSet<>();
s1.add("a");
s1.add("b");
s1.add("c");
System.out.println(s1);
s1.remove("c");
System.out.println(s1);
```

Przykład 34

boolean removeAll(Collection<?> c) – usuwa wszystkie, elementy ze zbioru, które zostały znalezione w drugiej kolekcji

```
Set<String> s0 = new HashSet<>();
s0.add("a");
s0.add("b");

Set<String> s1 = new HashSet<>();
s1.add("a");
s1.add("b");
s1.add("c");

s1.removeAll(s0);

System.out.println(s1);
```

Zadanie 16

Map

https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/HashMap.html

- Słownik, tablica asocjacyjna, mapa

- Mapa (ang. map) jest kolekcją, która pozwala przechować odwzorowanie zbioru kluczy na listę wartości.
- Klucze muszą być unikalne, wartości natomiast mogą się powtarzać.
- Kluczami w mapie powinny być obiekty, których nie można zmodyfikować (ang. immutable), np. instancje takich klas jak String lub Integer – po zainicjalizowaniu tych obiektów nie możemy ich zmodyfikować.

Przykład 35

Map

Class HashMap<K,V>

java.lang.Object
    java.util.AbstractMap<K,V>
        java.util.HashMap<K,V>

Type Parameters:

K - the type of keys maintained by this map

V - the type of mapped values

Przykład 35 c.d.

V put(K key, V value) – umieszcza w słowniku nową parę klucz-wartość

```
Map<Integer, String> hm = new HashMap<>();
hm.put(1, "a");
hm.put(1, "a");
hm.put(2, "b");
System.out.println(hm);
```

Przykład 36

boolean containsKey(Object key) – sprawdź czy słownik zawiera klucz

```
Map<Integer, String> hm = new HashMap<>();
hm.put(1, "a");
hm.put(1, "a");
hm.put(2, "b");
//      System.out.println(hm);

System.out.println(hm.containsKey(1));
System.out.println(hm.containsKey(9));
```

Przykład 37

boolean containsValue(Object value) – sprawdź czy słownik zawiera wartość

```
Map<Integer, String> hm = new HashMap<>();
hm.put(1, "a");
hm.put(1, "a");
hm.put(2, "b");
//       System.out.println(hm);

System.out.println(hm.containsValue("a"));
System.out.println(hm.containsValue("z"));
```

Przykład 38

V get(Object key) – zwróć wartość dla danego klucza

```
Map<Integer, String> hm = new HashMap<>();
hm.put(1, "a");
hm.put(1, "a");
hm.put(2, "b");
//      System.out.println(hm);

System.out.println(hm.get(1));
System.out.println(hm.get(9));
```

Przykład 39

Set<K> keySet() - zwraca zbiór kluczy znajdujących się w słowniku

```
Map<Integer, String> hm = new HashMap<>();
hm.put(1, "a");
hm.put(1, "a");
hm.put(2, "b");
//      System.out.println(hm);

System.out.println(hm.keySet());
Set<Integer> si = hm.keySet();
System.out.println(si);
```

Przykład 40

Collection<V> values() - zwraca kolekcję wartości znajdujących się w słowniku

```
Map<Integer, String> hm = new HashMap<>();
hm.put(1, "a");
hm.put(1, "a");
hm.put(2, "b");

Collection<String> c = hm.values();
System.out.println(c);
Set<String> s1 = new HashSet<>();
s1.addAll(c);
System.out.println(s1);
```

Zadanie 17

Przykład 41

Queue

Kolejka – jak u lekarza, kto pierwszy pojawił się w kolejce, ten pierwszy wejdzie do gabinetu i ten pierwszy wyjdzie z kolejki, first-in, first-out, czyli pierwszy przyszedł, pierwszy wszedł i pierwszy wyjdzie.

https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/Queue.html

https://docs.oracle.com/javase/tutorial/collections/implementations/queue.html

Summary of Queue methods

	Throws exception	*Returns special value*
Insert	`add(e)`	`offer(e)`
Remove	`remove()`	`poll()`
Examine	`element()`	`peek()`

LinkedList implementuje Queue

Przykład 41 c.d.

```
Queue<Integer> q = new LinkedList<Integer>();
// dodaj jeśli możliwe
System.out.println(q.offer(1));
System.out.println(q);
System.out.println(q.offer(2));
System.out.println(q);

// zwróć pierwszy element z listy (jeśli istnieje)
System.out.println(q.peek());
System.out.println(q);

// usuń
System.out.println(q.poll());
System.out.println(q);

// zwróć pierwszy element z listy (jeśli istnieje)
System.out.println(q.peek());
System.out.println(q);

// usuń
System.out.println(q.poll());
// null
// zwróć pierwszy element z listy (jeśli istnieje)
System.out.println(q.peek());
System.out.println(q);

// usuń
System.out.println(q.poll());
```

Zadanie 18