

# Kolekcje

## Platformy Technologiczne

Michał Piotrowski  
Michał Wójcik

Katedra Architektury Systemów Komputerowych  
Wydział Elektroniki, Telekomunikacji i Informatyki  
Politechnika Gdańska

4 marca 2011

1 Kolekcje

2 Generyki

3 Typy enum

Kolekcja — kontener, obiekt grupujący elementy.

Collection Framework — narzędzia służące reprezentacji i manipulacji kolekcjami:

- interfejsy — abstrakcyjne typy reprezentujące kolekcje,
- implementacje — konkretne implementacje interfejsów,
- algorytmy — metody pozwalające na wyszukiwanie oraz sortowanie, mogą być wykorzystywane niezależnie od implementacji kolekcji.

## Typy kolekcji:

- `Collection` — najbardziej ogólny rodzaj kolekcji, brak konkretnej implementacji, brak ograniczeń typu powtarzalność elementów lub kolejność,
- `Set` — nie zawiera dwóch lub więcej takich samych elementów,
- `List` — sekwencja elementów, może zawierać takie same elementy,
- `Queue` — sekwencja elementów (np. FIFO) lub posortowane elementy (kolejki priorytetowe),
- `Map` — mapowanie kluczy i wartości, nie może zawierać powtórzonych kluczy, jeden klucz może wskazywać jedną wartość,
- `SortedSet` — zawiera posortowane elementy,
- `SortedMap` — sortowanie względem kluczy.

# Operacje na kolekcjach

## Przeglądanie kolekcji:

- for-each:
  - `for (Object o: collection) {...}`,
  - nie należy usuwać elementów,
  - również dla tablic;
- iterator:
  - pozwala na usuwanie elementów (usuwać jedynie za pomocą iteratora!),
  - wykorzystanie podczas filtrowania,
  - `for (Iterator it = c.iterator(); it.hasNext(); )  
  {it.next(); it.remove();}`.

## Bulk operations — operacje na całej kolekcji:

- `containsAll`, `addAll`, `removeAll`, `retainAll`, `clear`.

## Konwersja na tablice:

- `Object[] toArray()` — zwraca kolekcję jako tablicę typu `Object`,
- `Type[] toArray(new Type[0])` — zwraca kolekcję jako tablicę typu `Type`.

## Najważniejsze informacje:

- `Collections.sort(l);` — sortowanie listy,
- interfejs `Comparable`:
  - pozwala na automatyczne sortowanie,
  - `int compareTo(T o);` — porównywanie obiektów.
- `int hashCode();`,
- `boolean equals();` — sprawdza czy obiekty są równe, równe obiekty posiadają ten sam `hashCode`, zgodna z metodą `CompareTo`,
- `Comparator`:
  - sortowanie obiektów nie implementujących `Comparable`,
  - `int compare(T o1, T o2);`

## Implementacje kolekcji:

- Set:
  - HashSet — brak zapewnienia sortowania,
  - TreeSet — sortowany na bieżąco,
  - LinkedHashMap — zachowana kolejność dodawanych elementów;
- List:
  - ArrayList — zachowania kolejność dodawanych elementów,
  - LinkedList — stały czas przeglądania i usuwania elementów, liniowy pozycjonowania elementów;
- Queue:
  - LinkedList,
  - PriorityQueue — sortuje według kryterium podanego przy inicjalizacji;
- Map:
  - HashMap — brak sortowania, największa wydajność,
  - TreeMap — sortowanie kluczy,
  - LinkedHashMap — zachowana kolejność dodawania elementów;

1 Kolekcje

2 Generyki

3 Typy enum

## Podstawowe informacje:

- parametryzowanie klas, metod, typów,
- usuwane na poziomie kompilacji,
- często używane podczas inicjalizacji obiektów kolekcji.

## Powszechnie używane nazewnictwo:

- E — element,
- K — klucz,
- N — numer,
- T — typ,
- V — wartość,
- S, U, V, itd — kolejne typy.

## Ograniczenia:

- `public class Box<T extends MyClass>`,
- `public class Box<T extends MyClass & MyInterface>`,
- `public void method(Box<Number> n);`
  - nie zaakceptuje `Box<Integer>`;
- `public void method(Box<? extends Number> n);`
  - zaakceptuje `Box<Integer>`;
- sprawdzanie typów jest niemożliwe (item instanceof E).

1 Kolekcje

2 Generyki

3 Typy enum

Enum — typ, którego pola składają się wyłącznie ze stałych:

- `public enum Day { SUNDAY, MONDAY, ... },`
- wykorzystanie w `switch`:
  - `case MONDAY:`,
  - nigdy! `case Day.MONDAY::`
- `for (Day d : Day.values()),`
- prywatny konstruktor do tworzenia stałych w momencie definicji.