Przetwarzanie struktur danych w javie

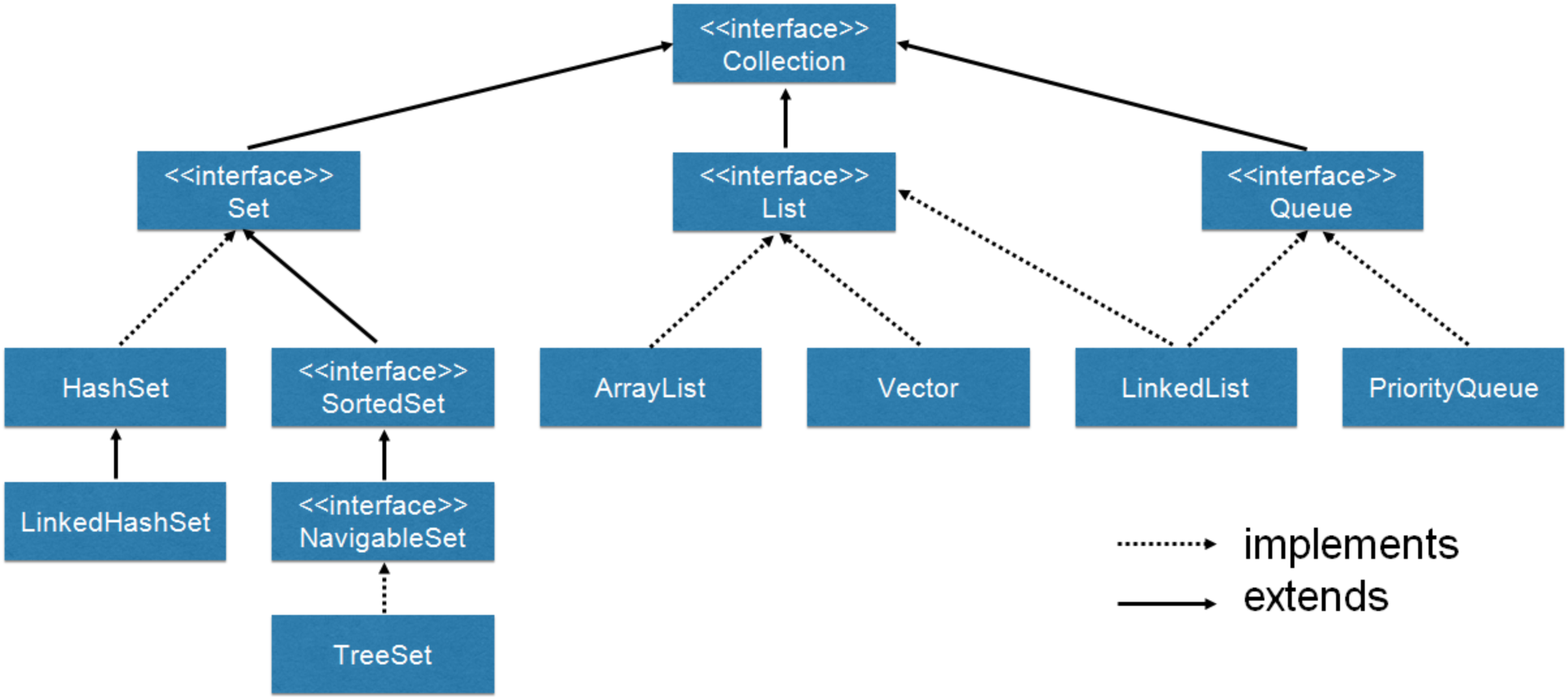
Trener: Mariusz Pawłowski

Gdańsk, 20 stycznia 2018 roku

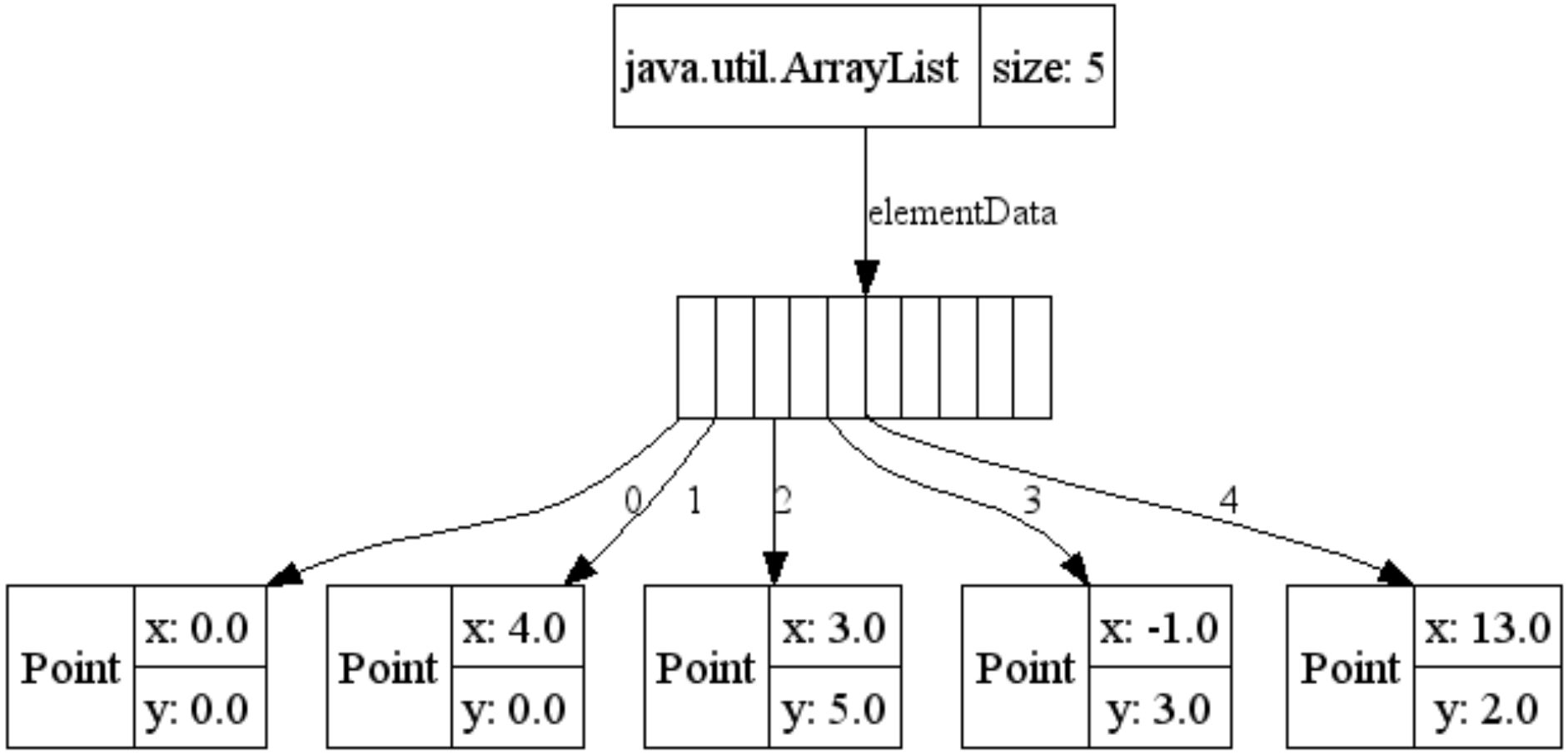
Struktury danych w javie - kolekcje

- Podstawowe typy kolekcji
  - zbiory (Set) - unikatowość elementów
  - listy (List) - uporządkowanie elementów
  - kolejki (Queue) – kolejkowanie elementów
  - mapy (Map) – tablica klucz-wartość

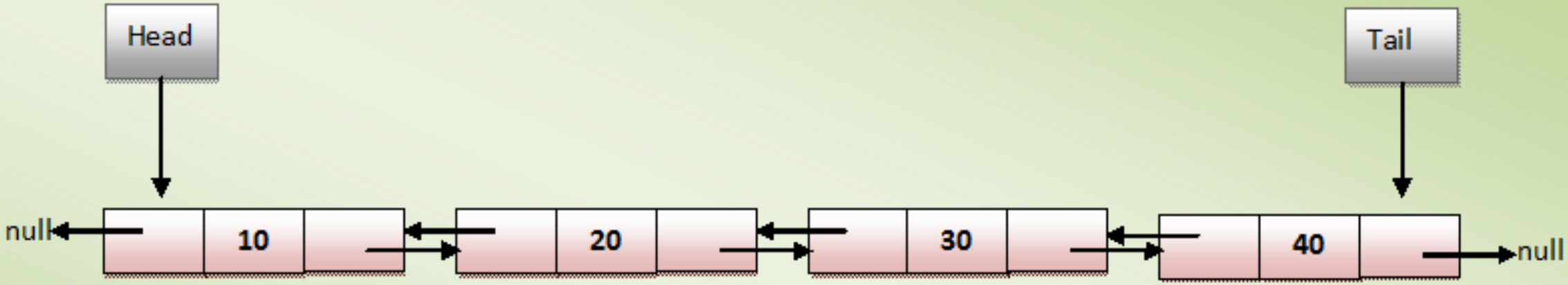
Collection Interface



Struktury danych w javie - ArrayList



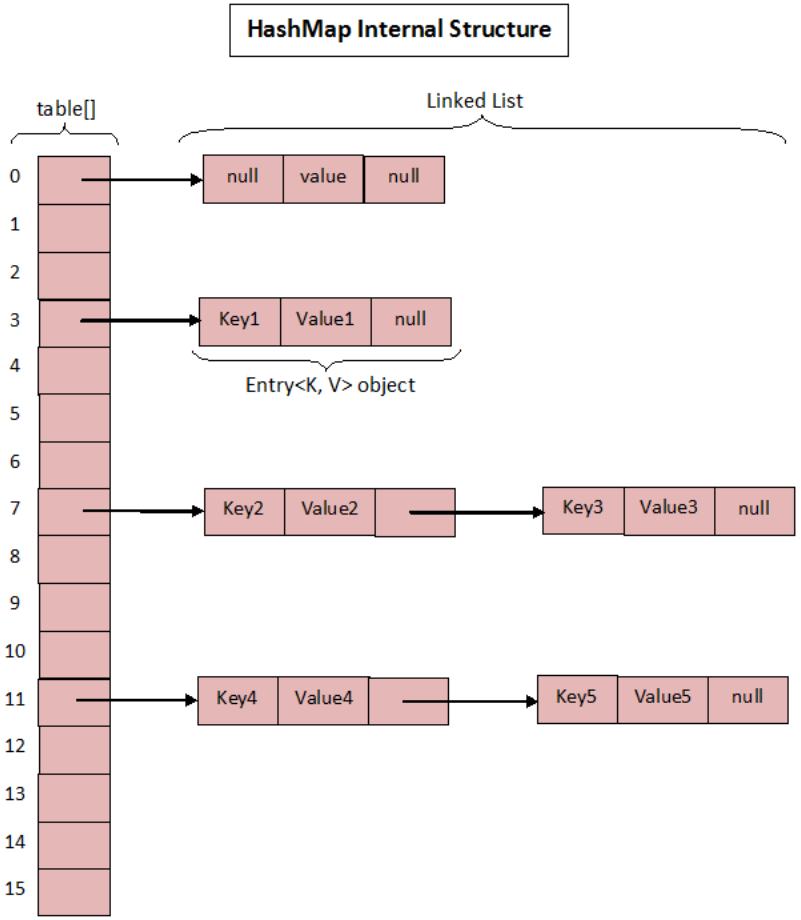
Struktury danych w javie - LinkedList



Insertions and Removals in LinkedList are faster than ArrayList. Because, You don't need to resize LinkedList after each insertions and removals.

Retrieval operations in LinkedList are slow compared to ArrayList. Because, it needs traversing from the beginning or from end to reach the element.

Struktury danych w javie - HashMap



ArrayList vs LinkedList

- ArrayList
  - Uniwersalne zastosowanie
  - Inne mają zastosowanie w specyficznych przypadkach
  - Szybsze pobieranie elementów niż w LinkedList
- LinkedList
  - Przechowuje elementy jako listę powiązanych ze sobą obiektów
  - Ma przewagę w przypadku dodawania pojedynczych elementów

Sortowanie - TreeSet

- TreeSet to struktura, która automatycznie przechowuje elementy posortowane (sortowanie naturalne lub poprzez komparator)
- Element, który wstawiamy do TreeSet musi implementować interfejs Comparable<TypElementu>

Zadanie 1

- Stworzyć metodę która zliczy wystapienia elementów w liście.
  - Stworzyć listę dowolnego typu
  - Wstawić 5 elementów
  - Wyświetlić liczbę elementów

Zadanie 2

- Stworzyć metode, która sprawdzi czy wszystkie znaki w Stringu są unikalne
  - dla stringa “ab” zwróci true
  - dla stringa “aa” zwróci false

Zadanie 3

- Stworzyć metodę: List<String> distinct(List<String> list) { }, która usunie duplikaty z Listy.

Zadanie 4

- Stworzyć klasę Person (String name, String surname, String pesel).
- Stworzyć kolekcję Map<String, Person> people = new HashMap();
- Dodać 4 przykładowe osoby do której kluczem jest pesel.
- Wydrukować wszystkie pesele z kolekcji
- Stwórz listę osób i posortuj ją alfabetycznie

Zadanie 5

- Stworzyc klasę Book (tytuł, imię i nazwisko autora, rok wydania, wydawnictwo)
- Stworzyc klase PozycjaNaPolce
- Stworzyć mapę <PozycjaNaPolce, Ksiazka>
- Wypisać wszystkie zajętę pozycje na półce

Zadanie 6

- Stwórz TreeSet posortowany rosnąco, wg nazwiska, korzystając z kodu z poprzedniego zadania, z wykorzystanie interfejsu Comparator (w odróżnieniu od poprzednich zajęć, gdzie korzystaliśmy z Comparable)

Zadanie 7

Pobierz kod z repozytorium dla zadania 7, zaimplementuj następujące metody z klasy UserService:

- `public static List<User> findUsersWhoHaveMoreThanOneAddress(List<User> users)`
dającą jako wynik listę użytkowników, którzy w obiekcie *personDetails* mają więcej niż jeden adres,
- `public static Person findOldestPerson(List<User> users)`
dającą jako wynik dane użytkownika (personDetails), który jest najstarszy (pole *age* w klasie *Person*),

Zadanie 7 cd

- `public static User findUserWithLongestUsername(List<User> users)`
  dającą jako wynik użytkownika o najdłuższej nazwie użytkownika,
- `public static String getNamesAndSurnamesCommaSeparatedOfAllUsersAbove18(List<User> users)`
  dającą jako wynik napis składający się z imion i nazwisk oddzielonych przecinkiem użytkowników, którzy mają więcej niż 18 lat

Zadanie 7 cd

- `public static List<String> getSortedPermissionsOfUsersWithNameStartingWithA(List<User> users)`
  dającą jako wynik posortowaną od ‘a’ do ‘z’ listę nazw uprawnień użytkowników, których imię zaczyna się na literę ‘A’,
- `public static void printCapitalizedPermissionNamesOfUsersWithSurnameStartingWithS (List<User> users)`
  wypisującą na ekran uprawnienia użytkowników, których nazwisko zaczyna się na literę ‘S’,

Zadanie 7 cd

- `public static Map<Role, List<User>> groupUsersByRole(List<User> users)`

  dającą jako wynik mapę, gdzie kluczem jest rola, a wartościami lista użytkowników, którzy mają daną rolę przypisaną,

- `public static Map<Boolean, List<User>> partitionUserByUnderAndOver18(List<User> users)`

  dającą jako wynik mapę, gdzie kluczem jest wartość logiczna (true lub false), a wartością dla false lista użytkowników którzy mają mniej niż 18 lat, a dla true lista użytkowników, którzy mają 18 lub więcej lat

Strumienie w Java 8

- klasy z pakietu java.util.stream
- do operacji funkcyjnych na sekwencjach elementów (strumieniach)
  - redukcje (reduce) – suma, średnia, itd.
  - odwzorowanie (map) –
  - filtrowanie (filter)
- możliwe zrównoleglenie operacji
- zintegrowane z Collections API

Java 8 - strumienie

- strumienie vs. kolekcje
  - nie przechowują elementów
  - natura funkcyjna (źródłowa kolekcja nie jest modyfikowana)
  - leniwe wyliczanie
  - mogą być nieograniczone
  - mogą być odwiedzone tylko raz w ich cyklu życia

Java 8 - strumienie

- Operacje pośrednie (intermediate)
  - wynik to nowy strumień,
  - są wyliczane leniwie
  - mogą przechowywać stan (np. sorted) lub nie (np. filter, map)
- Operacje końcowe (terminal)
  - obliczenie wyniku
  - kończą sekwencję operacji strumieniowych

Java 8 - strumienie

- operacja redukcji (reduce)
  - dla ciągu elementów daje wynik liczbowy (np. suma) bąd akumuluje elementy w nowej liście
  - ogólne operacje: reduce i collect
  - wyspecjalizowane: sum(), max(), count()

Java 8 - strumienie

- operacja kolekcji (collect)
  - podobnie jak reduce zbiera wszystkie elementy kolekcji
  - umożliwia grupowanie elementów
  - podział strumieni na dwa
  - wielopoziomowy podział

Java 8 - strumienie

```
Set<String> set = people.stream().map(Person::getName)
      .collect(Collectors.toCollection(TreeSet::new));

String joined = things.stream()
      .map(Object::toString)
      .collect(Collectors.joining(", "));

int total = employees.stream()
      .collect(Collectors.summingInt(Employee::getSalary)));

Map<Department, List<Employee>> byDept
      = employees.stream()
      .collect(Collectors.groupingBy(Employee::getDept));
```

Zadanie 8

- Wykonaj polecenia z zadania 7 z wykorzystaniem strumieni

Dziękuję za uwagę