Porovnání kontejnerů, třída `Collections`

```
<link rel="stylesheet"  href="http://cdnjs.cloudflare.com/ajax/libs/font-awesome/3.1.0/css/font-awesome.min.css">
```

Pomocná třída `Collections`

- Java Core API v balíku `java.util` nabízí třídu `Collections`
- nabízí jen statické metody a proměnné (tzv. utility class), nelze od ní vytvářet instance
- nabízí škálu užitečných metod pro práci s kontejnery

Javadoc třídy `Collections`

Souběžný přístup

- moderní kontejnery **nejsou synchronizované**
- jinak řečeno, nepřipouštějí souběžný přístup z více vláken
- standardní (nesynchronizovaný) kontejner lze však **zabalit**
- `synchronizedSet, synchronizedList, synchronizedCollection, ⋯`
- metoda vrátí novou synchonizovanou kolekci

```
List<String> list = new ArrayList<>();
List<String> syncedList = Collections.synchronizedList(list);
```

Získání nemodifikovatelných kontejnerů

- kontejnery jsou standardně modifikovatelné (read/write)
- nemodifikovatelné kontejnery se používají při vracení hodnot z metod
- `unmodifiableSet, unmodifiableList, unmodifiableCollection, ⋯`
- metoda vrátí novou nemodifikovatelnou kolekci

```
Set<String> set = Set.of("Donald Trump", "Barrack Obama", "Hillary Clinton");
return Collections.unmodifiableSet(set);
```

Getter na kolekci je vždy nemodifikovatelný!

Návrhový vzor *Proxy*

- Při vytváření nemodifikovatelné kolekce se nikam nic nekopíruje. Původní kolekci se pouze předřadí objekt (kolekce), který nemodifikační metody přeposílá původní kolekci, zatímco modifikační metody selhávají s výjimkou
- Jedná se použití návrhového vzoru Proxy:
  - Původní i nový objekt jsou stejného typu (v našem případě `Collection`).
  - Nový objekt na pozadí komunikuje s původním objektem.

Prázdné kontejnery

- třída obsahuje konstanty `EMPTY_SET, EMPTY_LIST, EMPTY_MAP`
- metody `emptyList(), emptyMap(), emptyIterator(), ⋯`
- preferujeme metody, protože konstanty postrádají typ, tj. chybí jim typová kontrola
- vrácené kolekce jsou **nemodifikovatelné**
- šetříme vytváření nové kolekce

```
Collections.<String>emptyList();
```

Metody v `Collections` I

- `Collections.binarySearch`
  - binární vyhledávání v kontejneru
- `Collections.reverseOrder, rotate`
  - obrácení, rotace pořadí prvků
- `Collections.swap`
  - prohazování prvků
- `Collections.shuffle`
  - náhodné zamíchání prvků

Metody v `Collections` II

- `Collections.sort`
  - uspořádání (přirozené, anebo pomocí komparátoru)
- `Collections.min, max`
  - minimální, maximální prvek (s definovaným uspořádaním)
- `Collections.nCopies`

    - vytvoří kolekci n stejných prvků
- `Collections.frequency`
    - kardinalita dotazovaného prvku v dané kolekci

Srovnání implementací kolekcí

- `ArrayList` — na bázi pole
    - rychlý přímý přístup (přes index)
- `LinkedList` — na bázi lineárního zřetězeného seznamu
    - rychlý sekvenční přístup (přes iterátor)
- `HashMap`, `HashSet` — na bázi hašovacích tabulek
    - rychlejší, ale neuspořádané
    - lze získat iterátor procházející klíče uspořádaně
- `TreeMap`, `TreeSet` — na bázi vyhledávacích stromů
    - pomalejší, ale uspořádané
- `LinkedHashSet`, `LinkedHashMap` — spojení výhod obou

Kontejnery a jejich rozhraní

- `Set` (množina)
    - `HashSet` (založena na hašovací tabulce)
    - `TreeSet` (černobílý strom)
    - `LinkedHashSet` (zřetězené záznamy v hašovací tabulce)
- `List` (seznam)
    - `ArrayList` (implementován pomocí pole)
    - `LinkedList` (implementován pomocí zřetězeného seznamu)
- `Deque` (fronta - obousměrná)
    - `ArrayDeque` (fronta pomocí pole)
    - `LinkedList` (fronta pomocí zřetězeného seznamu)
- `Map` (asociativní pole/mapa)
    - `HashMap` (založena na hašovací tabulce)
    - `TreeMap` (černobílý strom)
    - `LinkedHashMap` (zřetězené záznamy v hašovací tabulce)

Kontejnery a výjimky

Při práci s kontejnery může vzniknout řada *výjimek*.

Některé z nich i s příklady:

- `IllegalStateException`
  - vícenásobné volání `remove()` bez volání `next()` v iterátoru
- `UnsupportedOperationException`
  - modifikace **nemodifikovatelné** kolekce
- `ConcurrentModificationException`
  - iterovaný prvek (for-each cyklem) byl odstraněn

Většina výjimek je *běhových* (runtime), tudíž není nutné je řešit. (Samozřejmě je ale třeba psát kód tak, aby nevznikaly. :))

Nepovinné metody

- funkcionalita kontejnerů je předepsána *rozhraním*
- některé metody rozhraní jsou *nepovinné* — třídy jej nemusí implementovat
  - např. `add`, `clear`, `remove`
  - metoda existuje, ale nelze ji použít, protože volání *vyhodí výjimku* `UnsupportedOperationException`
- Důvod?
  - např. nehodí se implementovat zápisové operace, když kontejnery budou read-only (`unmodifiable`)