

# Абстрактни структури от данни. Колекции – имплементация в Java.

Кристиян Маринов / SAP Labs Bulgaria  
Ноември 2014

Public



# Теория и основи

---

## Спецификация

- Предусловия (requires)
  - Следствия (effects, throws, returns, modifies)
- Определя поведение

## Инвариант

- Свойства
  - Ограничения
- Определя състояние

# Инвариант

---

Ненарушими свойства на един обект – обещание на обектът пред себе си и неговите клиенти.

Гарантирано от всички публични методи, „споразумение“ помежду им.

***“I promise to leave the common area tidy when I exit. In return, I (the method) expect to get it tidy (satisfying specific constraints) every time I enter.”***

*MIT, 6.170, Lecture 8*

# Инвариант на практика

---

Често „имплементиран“ като Javadoc/коментар, но има и други начини – JML\*.

Трябва ли кодът да го проверява/налага?

- Да, ако е лесно
- Да, в debug режим (дори и да не е лесно)
- Трудно се намират аргументи против

На практика – **рядко**. Много определящи фактори (компонент, среда, натоварване и т.н.)

→ **Тестове!**

\* JML – Java Modeling Language

# Абстрактни структури от данни

---

Защо „абстрактни“? Защото се интересуваме само от концепциите, вътрешното представяне на данните няма значение.

**Дефиниция**      Набор от свойства, допустими операции и ограничения.

**Примери**      стек, опашка, дърво, точка, триъгълник и т.н.

Грубо казано, АСД == инвариант + спецификация.

# Примери за абстрактни структури

---

```
class Point {  
    public float x;  
    public float y;  
}
```

```
class Point {  
    public float r;  
    public float theta;  
}
```

**Цел**      концепцията „точка“

## Бонуси

- Отложени решения
- Оправяне на бъгове
- Производителност

# Създаване на абстрактни структури от данни

---

## Immutable

1. Overview
2. Abstract fields
3. Creators
4. Observers
5. **Producers**

## Mutable

1. Overview
2. Abstract fields
3. Creators
4. Observers
5. **Mutators**

**Пример** множество;  
`org.eclipse.swt.graphics.Point (x, y)`

# Създаване на абстрактни структури от данни (2)

---

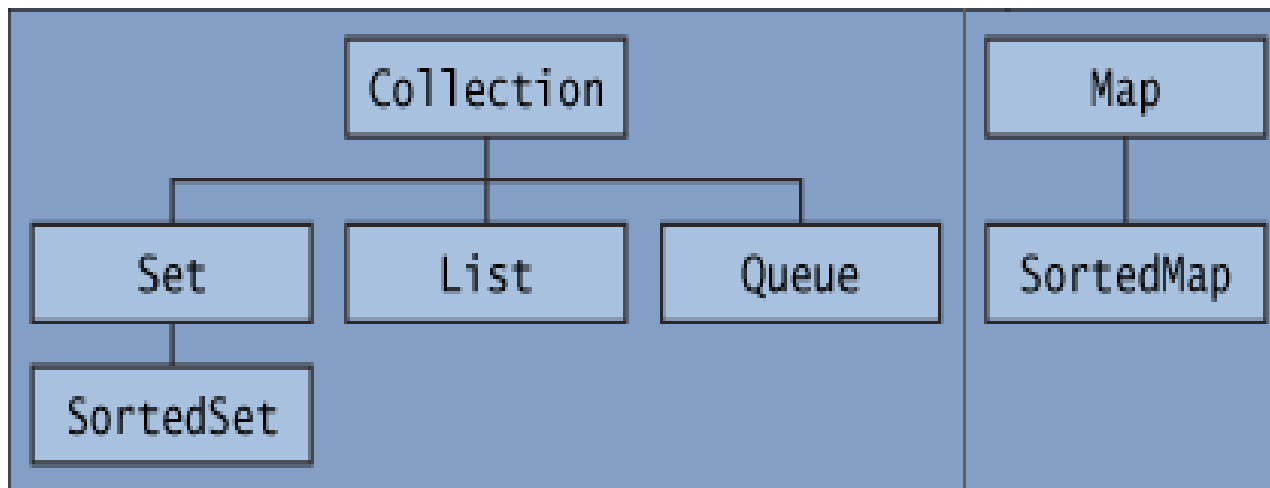
Два начина за изразяване в Java – чрез класове и интерфейси.

Различни предимства и недостатъци → **няма рецепта, само опит!**



# Абстрактни структури в Java

...от птичи поглед:



Още познати като **колекции**.

# Колекции

---

**Дефиниция**      контейнер, който групира няколко (логически свързани) елементи в едно цяло.

**Пример**              покер ръка (колекция от карти), пощенска кутия (колекция от писма) и т.н.

В Java – framework (интерфейси + имплементации + алгоритми).

## Ползи

- Топла вода (и по-малко работа) за нас!
- Качество + производителност
- Преизползване
- Оперативна съвместимост между API-та

# Колекции – класификация

---

## Всеобща употреба

ArrayList, LinkedList, TreeMap, TreeSet, HashMap, HashSet и др.

## Специализирани

EnumSet, CopyOnWriteArraySet, WeakHashMap и др.

## Конкурентни

ConcurrentMap, ArrayBlockingQueue, Vector, LinkedBlockingQueue, PriorityBlockingQueue и др.

# Колекции в детайли

---

**ArrayList** – списък върху разширяващ се масив. Константна амортизирана сложност за повечето операции

**LinkedList** – свързан списък

**TreeMap/TreeSet** – червено-черни дървета, запазват естествената наредба\*. Елементите трябва да имплементират интерфейсите Comparable & equals(). Гарантирана логаритмична сложност за повечето операции

**HashMap/HashSet** – хеш таблици, нямат наредба. Елементите трябва да имплементират hashCode() & equals(). Константна\*\* сложност за повечето операции

\*, \*\* Повече детайли в документацията на всеки клас

# Колекции в детайли – популярни методи

---

## **ArrayList<E>**

- add(E) / addAll(Collection<? extends E>)
- get(int)
- remove(Object) / remove(int) / removeAll(Collection<?>)
- iterator() / size()

## **HashSet<E>**

- add(E)
- contains(Object)
- remove(Object)
- iterator() / size()

## **TreeMap<K, V>**

- put(K, V)
- get(Object)
- remove(Object)
- keySet() / values()

## Колекции в детайли (2)

---

**EnumSet** – множество с елементи от enum типове. Силно оптимизиран за време/място. Честа употреба – битмаски. Гарантирано константно сложни операции

**WeakHashMap** – речник със „слаби“ референции (повече в следващите лекции)

И други в `java.util.*`

\* Повече детайли в документацията на всеки клас

## Колекции в детайли (3)

---

**ConcurrentMap** – речник с атомарни put/remove/replace методи.

**ArrayBlockingQueue** – ограничена по размер блокираща опашка, реализирана върху масив :)

И други в `java.util.concurrent`.\*

\* Повече детайли в документацията на всеки клас

# Алгоритми върху колекции

---

Съществена част от collections framework!

Най-често употребявани

- **Сортиране** – подобрен merge sort; бърз и стабилен
- **Търсене** – линейно, двоично
- **Разбъркване**
- **Манипулация** – обръщане, запълване, копиране, размяна, масово добавяне
- **Намиране на екстремални стойности** – минимум/максимум
- **Композиция** – честота на срещане, покриване

и други..





# Благодаря за вниманието!

Кристиян Маринов  
[k.marinov@sap.com](mailto:k.marinov@sap.com)

# ИЗПОЛЗВАНА ЛИТЕРАТУРА

---

Invariant и class invariant, също MIT 6.170 лекция 7/8

Java Modeling Language

Abstract Data Type, също MIT 6.170 лекция 7

Collection и прилежащите му

Java 7 Collections

Java Collections Tutorial