ULTERIORI IMPLEMENTAZIONI JAVA
`EnumSet & EnumMap`

Java

- **`EnumSet`** è un efficiente set per enumerativi
- A differenza delle altre implementazioni, questa è una classe astratta, *priva di costruttori pubblici*
- Le istanze si costruiscono tramite *metodi factory*
  - **`allOf(Class<E> elementType)`**
  - **`complementOf(EnumSet<E> s)`**
  - **`copyOf(Collection<E> c)`**
  - **`copyOf(EnumSet<E> s)`**
  - **`noneOf(Class<E> elementType)`**
  - cinque versioni di **`of(…)`**, da 1 a 5 argomenti
  - **`range(E from, E to)`**

Ciascuno di essi prima specializza la classe astratta *derivandone una concreta adatto al caso specifico*, poi la instanzia.

ESEMPIO: `EnumSet`

Java

- Creiamo un **`EnumSet`** di **`DayOfWeek`** e lavoriamoci un po':

```
jshell> Set<DayOfWeek> s = EnumSet.of(DayOfWeek.MONDAY)
s ==> [MONDAY]

jshell> s.add(DayOfWeek.FRIDAY)
$4 ==> true

jshell> s
s ==> [MONDAY, FRIDAY]
```

- Creazione di range (con o senza import static su **`DayOfWeek`**)

```
jshell> EnumSet.range(DayOfWeek.WEDNESDAY, DayOfWeek.SATURDAY)
$8 ==> [WEDNESDAY, THURSDAY, FRIDAY, SATURDAY]
```

```
jshell> import static java.time.DayOfWeek.*

jshell> EnumSet.range(MONDAY,WEDNESDAY)
$10 ==> [MONDAY, TUESDAY, WEDNESDAY]
```

- Creazione di un set inizialmente vuoto (occhio all'argomento!):

```
jshell> EnumSet<DayOfWeek> ss = EnumSet.noneOf(DayOfWeek.class)
ss ==> []
```

ESEMPIO: `EnumSet` (segue)

Java

- **`EnumSet`** non accetta oggetti nulli (altrimenti, NPE):

```
jshell> DayOfWeek d = null;
d ==> null

jshell> ss.add(d)
|  java.lang.NullPointerException thrown
|        at EnumSet.typeCheck (EnumSet.java:395)
|        at RegularEnumSet.add (RegularEnumSet.java:161)
|        at RegularEnumSet.add (RegularEnumSet.java:36)
|        at (#13:1)
```

- Ulteriori esempi d'uso:

```
jshell> EnumSet.complementOf(ss)
$15 ==> [MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY]

jshell> EnumSet.complementOf(EnumSet.range(MONDAY,WEDNESDAY))
$16 ==> [THURSDAY, FRIDAY, SATURDAY, SUNDAY]
```

```
jshell> for(DayOfWeek d: EnumSet.range(MONDAY,WEDNESDAY))
   ...> System.out.println(d)
MONDAY
TUESDAY
WEDNESDAY
```

ULTERIORI IMPLEMENTAZIONI Java
`EnumSet` & `EnumMap`

Java

- **`EnumMap`** è un'efficiente mappa per enumerativi
- A differenza di **`EnumSet`**, ha normali costruttori pubblici:
  - `EnumMap(Class<K> keyType)`
  - `EnumMap(EnumMap<K,? extends V> m)`
  - `EnumMap(Map<K,? extends V> m)`
- e offre i classici metodi delle mappe
  - `put`
  - `get`
  - `keySet`

Analogamente a **`EnumSet`**, anche questa aborrisce *chiavi* nulle

```
jshell> m.put(null, 0.5)
|  java.lang.NullPointerException thrown
|        at EnumMap.typeCheck (EnumMap.java:743)
|        at EnumMap.put (EnumMap.java:264)
|        at (#31:1)
```

ESEMPIO: `EnumMap`

- Creiamo una **`EnumMap`** di **`DayOfWeek`** inizialmente vuota, inseriamoci tre coppie (giorno, costo della sosta), e lavoriamoci un po':

```
jshell> EnumMap<DayOfWeek,Double> m = new EnumMap<>(DayOfWeek.class)
m ==> {}

jshell> m.put(TUESDAY, 0.50)
$22 ==> null

jshell> m
m ==> {TUESDAY=0.5}

jshell> m.put(FRIDAY, 1.10)
$24 ==> null

jshell> m.put(SATURDAY, 1.50)
$25 ==> null

jshell> m
m ==> {TUESDAY=0.5, FRIDAY=1.1, SATURDAY=1.5}

jshell> m.put(FRIDAY, 1.20)
$27 ==> 1.1

jshell> m
m ==> {TUESDAY=0.5, FRIDAY=1.2, SATURDAY=1.5}
```

Il metodo put restituisce il *precedente valore* associato a quella chiave (*null* se non c'era)

Java