

Objektovo Orientované programovanie

4. Prednáška
LS 2024/2025
Juraj Petrík

Projekt

- Odovzdanie do konca týždňa!
- Nakreslite si hlavné prvky GUI
- Ďalšie info (najneskôr) v pondelok ráno

Biznis raňajky s JetBrains

- 18.3.2025



Business Breakfast

Join us for an opportunity to discuss:
Internships and student opportunities at JetBrains.
Kotlin and other programming languages.
Mastering our cutting-edge tools.
Anything else you may have questions about.

8:00 – 11:30

Business Breakfast

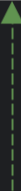
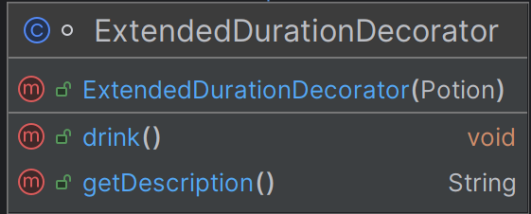
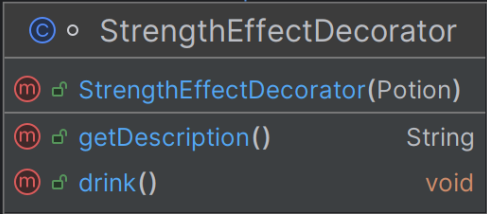
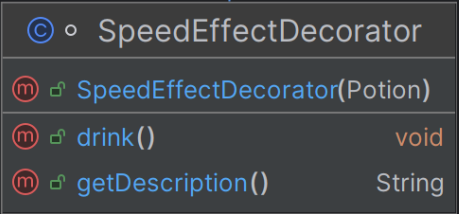
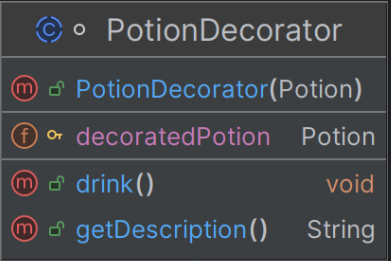
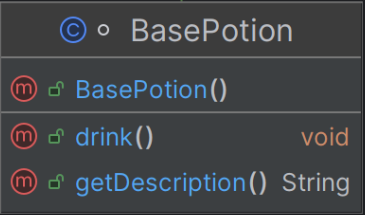
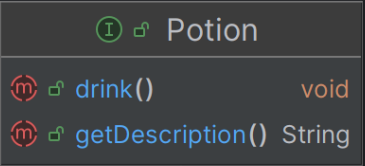
11:30 – 13:00

Lecture “What Makes Users
Love a Programming Language”

Decorator

- Pridávanie nového správania pomocou obaľovania (wrapping)
- Open-Closed Principle





Antivzory (pachy)

- Alebo ako niečo nemá vyzerat'
- Long Class
- Large Method
- Duplicate code
- Dead Code
- Inappropriate intimacy
- Circular dependency
- ...

Refactoring

- Odstránenie technologického dlhu
- Aby bol kód čistý (clean)
- Čistý kód =:
 - Menšia chybovosť
 - Jednoduchšia testovateľnosť
 - Jednoduchšia rozširiteľnosť
 - Jednoduchšie udržiavanie

Pamäť: Stack a Heap

- O uvoľnenie pamäte sa stará Garbage Collector
 - Keď na objekt neukazuje žiadna referencia
- Stack
 - Volania metód
 - Lokálne premenné (referencia na objekt je lokálna premenná!)
- Heap
 - Všetky objekty
- -Xmx2048m
- -Xms1048m

Fun with ~~flags~~ constructors

- Konštruktor sa volá pri vzniku objektu (new, ale nie len new 😊)
- Prekladač pridá prázdny konštruktor, ak sme nedefinovali konštruktor
- Super() – pridá compiler
- This() a Super()
- Pri dedení sa vykonávajú konštruktory z „hora dole“ (nadtrieda -> podtrieda)

Exceptions handling

- Exception (výnimka) nastáva keď sa stalo niečo neželané/neočakávané/nepovolené
- Try – catch - finally
- Exception – checked (compiler)
- RuntimeException – unchecked, tieto výnimky by nemali nastávať v „produkcii“, nie je odporúčané ich zachytávať
- Sú to objekty, sú polymorfné, od najkonkrétnejšieho po najvšeobecnejšie

```
try {}  
catch (ExceptionA e1) {}  
catch (ExceptionB e2) {}  
finally {}
```

Exceptions handling

- Throw/Throws
- Handle or declare

Custom Exception

- Extends Exception/RuntimeException

```
public class InvalidFoodException extends Exception {  
    public InvalidFoodException(String errorMessage) {  
        super(errorMessage);  
    }  
}  
  
public Food(String name, int amount, int type, int heal, int feed)  
throws InvalidFoodException {  
    super(name, amount, type);  
    if (amount <= 0 || type <= 0 || heal <= 0 || feed <= 0) {  
        throw new InvalidFoodException("Can't construct Food,  
can't have a negative amount, type, heal, feed");  
    }  
    this.setHeal(heal);  
    this.setFeed(feed);  
}
```

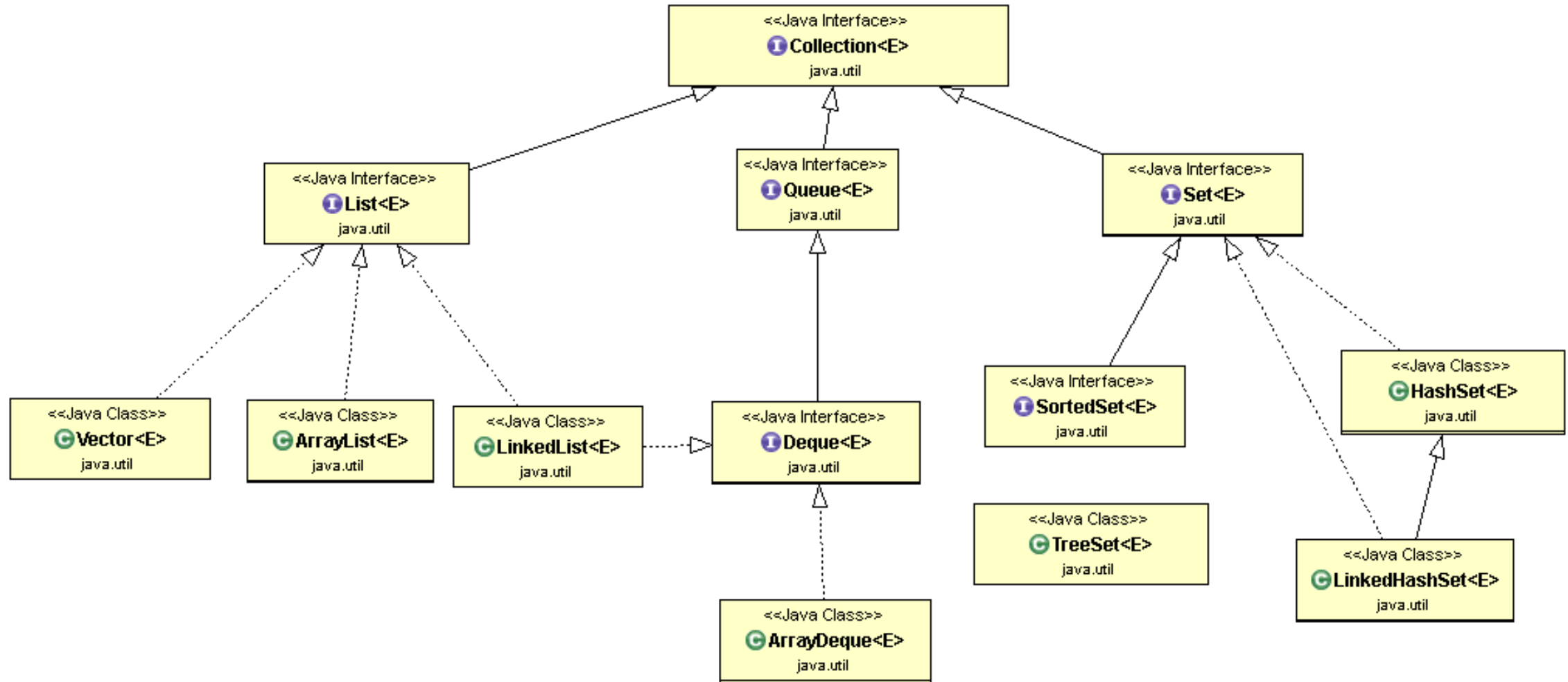
Collections and generics

- `List<String> names = new ArrayList<>();`
- `Collections.sort()` -> natural ordering
- Pod'me zoradit' Stevov

Generics

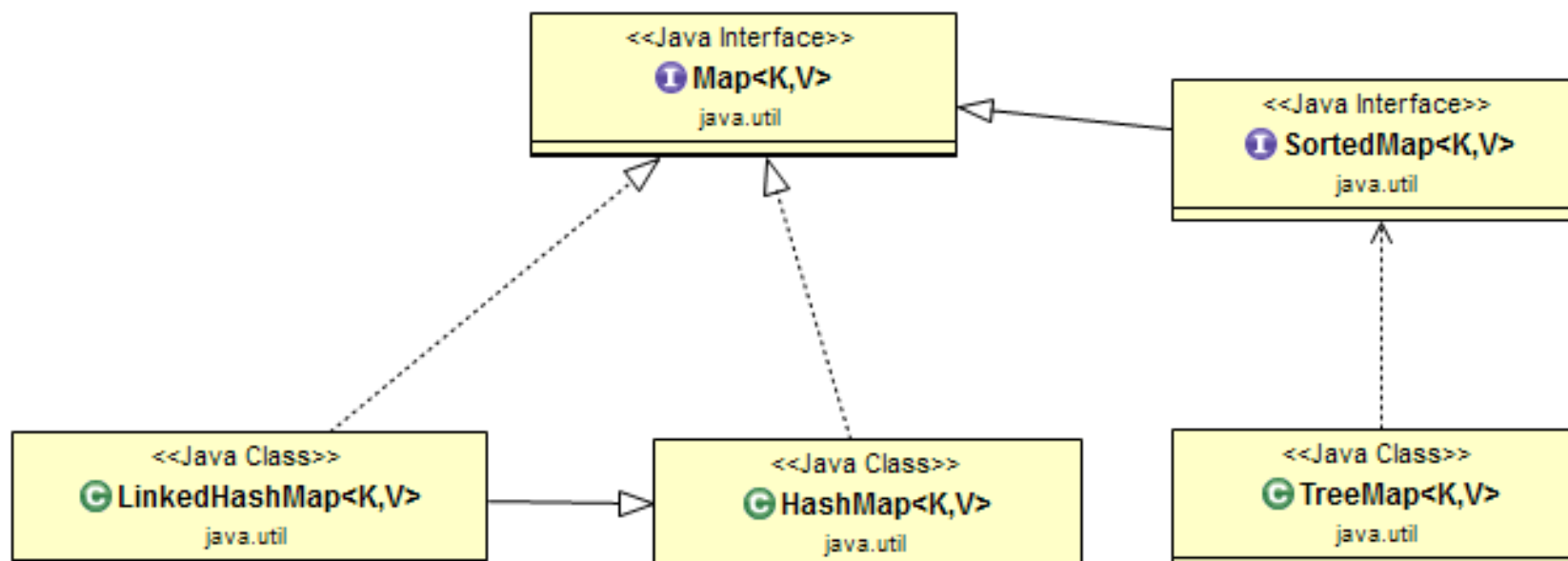
- Parametrizované typy <Typ>
- Generické triedy (napr. ArrayList):
 - Deklarácia triedy
 - Deklarácia metód pridávania
 - E je „typ“ s akým v triede chceme pracovať, compiler nahradí
 - E (T, R) je konvencia pomenovávania
- Generické metódy
 - `public <T extends SomeClass> void takeThing(ArrayList<T> list)`
 - `!= public void takeThing(ArrayList<SomeClass > list)`
- Extends = extends OR implements

List, Queue, Set



Map

- Stále považované že patří do Collections



Lambda expressions and double colon operator

- `steveArrayList.sort((one, two) -> one.getName().compareTo(two. getName()));`
- `steveArrayList.sort(Comparator.comparing(Character::getName));`

Testovanie

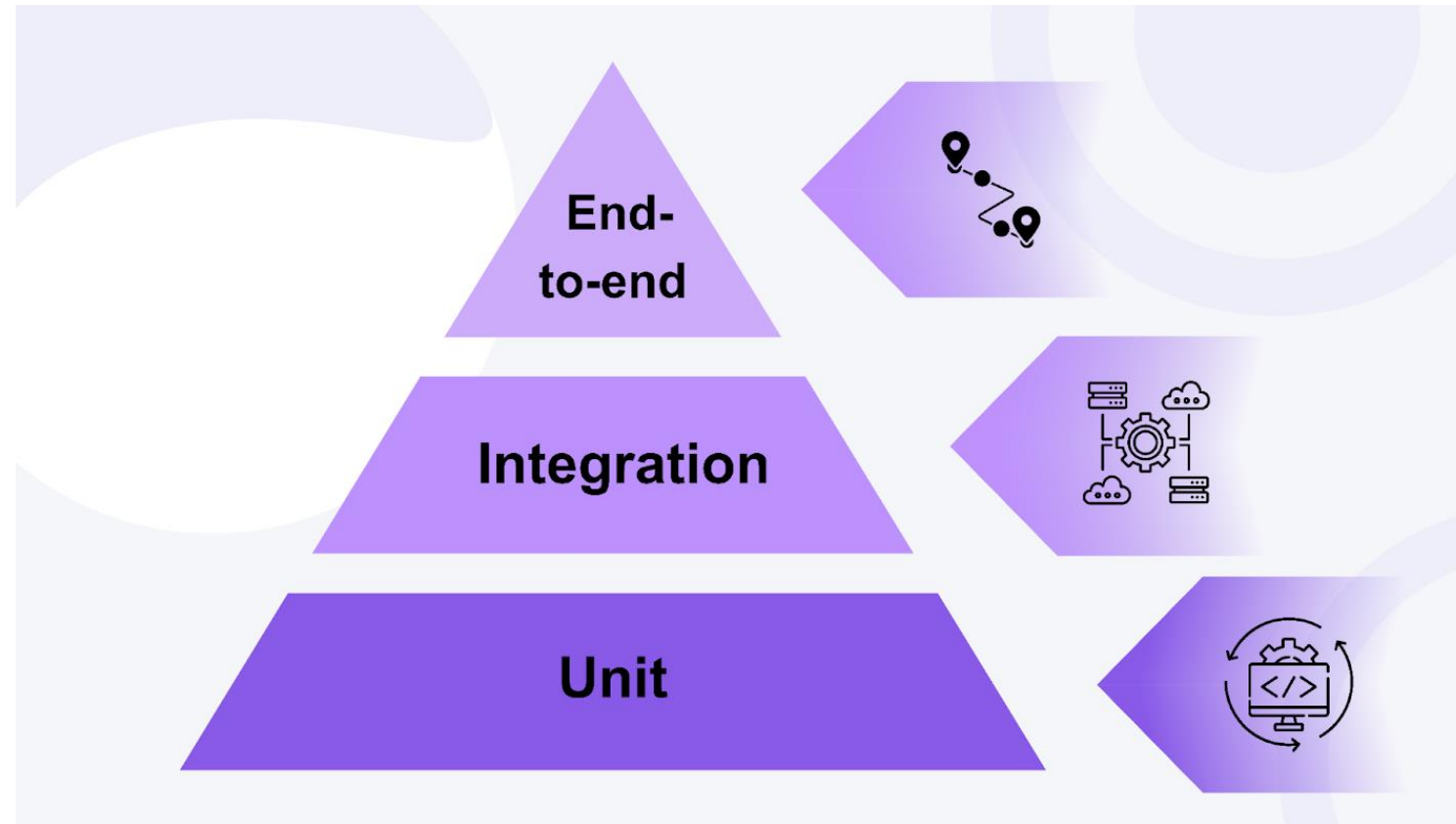
- Aké druhy testovania poznáte?

Testovanie

- Aký druh tu budeme riešiť?

Testovanie

- Jednotkové
- Integračné
- Funkčné
- Akceptačné
- Výkonnostné
- E2E
- Bezpečnostné



Jednotkové testovanie (JUnit)

- Testujeme jednotky (najmenšie funkčné celky)
- Integrálna súčasť vývoja - píše ich programátor čo implementuje danú funkcionálnosť
- Kód musí byť napísaný tak, aby bol testovateľný
- Napr. pri TDD (test driven development) najprv píšem testy, až potom implementujem
- Testujeme logiku, hraničné prípady, exception handling, atď
- Assert once

Jednotkové testovanie

- <https://junit.org/junit5/docs/current/user-guide/>
- <https://github.com/DiUS/java-faker>
- <https://site.mockito.org/>
- https://github.com/Jur1cek/jUnit_examples/

Quiz time