Více rozhraní a jejich rozšiřování (dědičnost)

Table of Contents

Implementace více rozhraní I

- Jedna třída může implementovat *více rozhraní*.
- Jednoduše v případě, kdy objekty dané třídy toho "mají hodně umět".
- Příklad: Třída `Person` implementuje 2 rozhraní:

```
public interface Informing { String retrieveInfo(); }
public interface Screaming { String scream(); }

public class Person implements Informing, Screaming {
  public String retrieveInfo() { ... }
  public String scream() { ... }
}
```

<<<<<<< HEAD == Více rozhraní se stejnými metodami ==

== Implementace více rozhraní II ==

Co kdyby obě rozhraní měla stejnou metodu? >>>>>>> 3388cf3 (ubuntu)

Obě rozhraní mají stejnou metodu?

```
public interface Informing { String retrieveInfo(); }
public interface Screaming { String retrieveInfo(); }
```

<<<<<<< HEAD .Úplně stejná hlavička = OK

```
- Mají-li metody úplně stejnou hlavičku, je to v pořádku:

>>>>>>> 3388cf3 (ubuntu)
[source,java]
public class Person implements Informing, Screaming {
  @Override
  public String retrieveInfo() { ... }
}
```

Implementace více rozhraní III

Mají-li stejný název i parametry, ale různý návratový typ, je to PROBLÉM.

```
public interface Informing { String retrieveInfo(); }
public interface Screaming { void retrieveInfo(); }
public class Person implements Informing, Screaming { ... }
```

To samozřejmě nelze!

Viz totéž u přetěžování metod

```
Person p = new Person();
// do we call method returning void or
// string and we ignore the result?
p.retrieveInfo();
```

Nesnesou se tedy metody lišící se *pouze návratovým typem*.

Rozhraní — vtip

Metody i samotné rozhraní by mělo obsahovat kvalitní dokumentaci s detailním popisem.

> Rozhraní je jako vtip. Když ho třeba vysvělovat, není tak dobré.

Zajímavost — rozhraní bez metod

- Občas se kupodivu používají i prázdná rozhraní, *nepředepisující žádnou metodu*.
- Úplně bezúčelné to není — deklarace, že třída implementuje určité rozhraní, poskytuje typovou informaci o dané třídě.
- Např. `java.lang.Cloneable`, `java.io.Serializable`.

Rozšiřování (dědičnost) rozhraní

- Rozhraní může převzít (můžeme říci též *dědit*) metody z existujících rozhraní.
- Říkáme, že rozhraní mohou být *rozšiřována* (*extended*).
- Rozšířené rozhraní by mělo *nabízet něco navíc* než to výchozí (rozšiřované).
- Příklad: Každá třída implementující `WellInforming` musí pak implementovat i metody z rozhraní `Informing`.

`WellInforming` jako rozšíření `Informing`

```
interface Informing {
  String retrieveInfo();
}
interface WellInforming extends Informing {
  String detailedInfo();
}
public class Person implements WellInforming {
  public String retrieveInfo() { ...  }
  public String detailedInfo() { ... }
}
```

Kde použít `implements` a kde `extends`

- Ztrácíte se v klíčových slovech?

  `implements`
  : třída implementuje rozhraní; ve třídě musím napsat kód (obsah) metod předepsaných rozhraním

  `extends`
  : když dědím ze třídy nebo rozšiřuji rozhraní, *dědím* metody automaticky

Vícenásobné rozšiřování rozhraní

- Rozhraní může dědit z více rozhraní zároveň:

```
public interface Informing { String retrieveInfo(); }
public interface Screaming { String scream(); }

public interface WellInforming extends Informing, Screaming {
  String detailedInfo();
}
```

- Každá třída implementující `WellInforming` musí implementovat všechny 3 metody.

Řetězení rozšiřování (dědičnosti)

- Dědičnost můžeme řetězit:

```
public interface Grandparent { int method1();  }
public interface Parent extends Grandparent { int method2(); }
public interface Child extends Parent { int method3(); }
```

- `Grandparent` pak obsahuje jednu metodu, `Parent` dvě, `Child` tři.

Kdy je vícenásobné rozšiřování možné

- Úplně stejné metody ze dvou rozhraní jsou OK

```
public interface A {
  void someMethod();
}
public interface B {
  void someMethod();
}
public interface AB extends A, B {
  // it is OK, methods have same signature
}
```

Kdy je vícenásobné rozšiřování možné

- Stejné metody s různými parametry ze dvou rozhraní jsou také OK.

```
public interface A {
  void someMethod();
}
public interface B {
  void someMethod(int param);
}
public interface AB extends A, B {
```

```
  // it is OK, methods have different params
}
```

Kdy vícenásobné rozšiřování není možné

- Dvě metody lišící se jen návratovým typem nejsou OK.
- Třída implementující rozhraní `AB` by musela mít dvě metody lišící se jen návratovým typem, a to nejde.

```
public interface A {
  void someMethod();
}
public interface B {
  int someMethod();
}
public interface AB extends A, B {
  // cannot be compiled
}
```