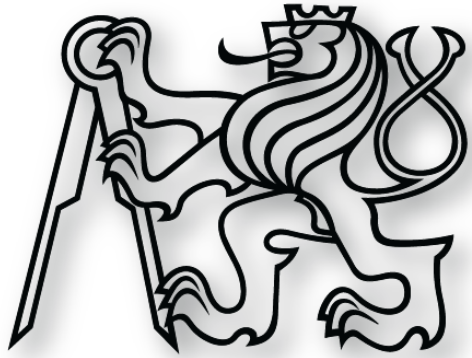


Kolekce

cvičení 10. března 2014

Karel Čemus
cemuskar@fel.cvut.cz

Opakování

- Třída
- Atribut
- Metoda
- Rozhraní
- Abstraktní třída
- Abstraktní metoda
- Překrývání (overriding)

- Kde v paměti jsou třídy?
- Kde v paměti jsou objekty?
- Kdy metodu deklarujeme ...
  - Ve dané třídě
  - V rodiči
  - V rozhraní
  - Abstraktní
- Kdy uděláme třídu abstraktní?

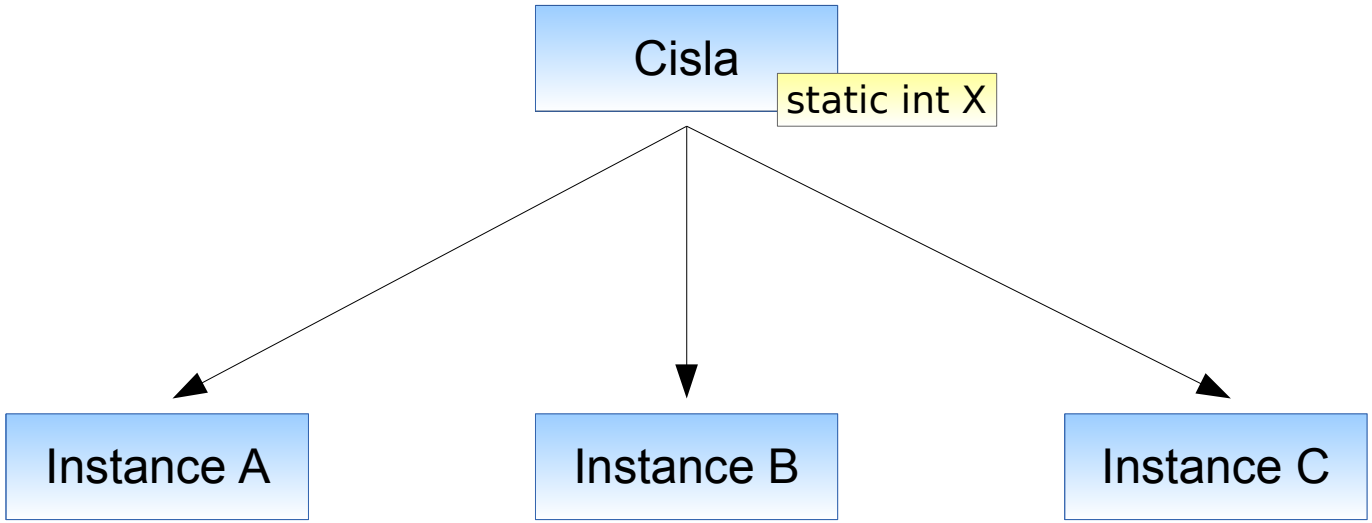
Domácí úkol

- Public vs private atributy
- Had: svlékni kůži
- Interface: metody pouze public
- Final metody

Dotazy?

Statické vs instanční

- Vlastník → adresace this vs Main

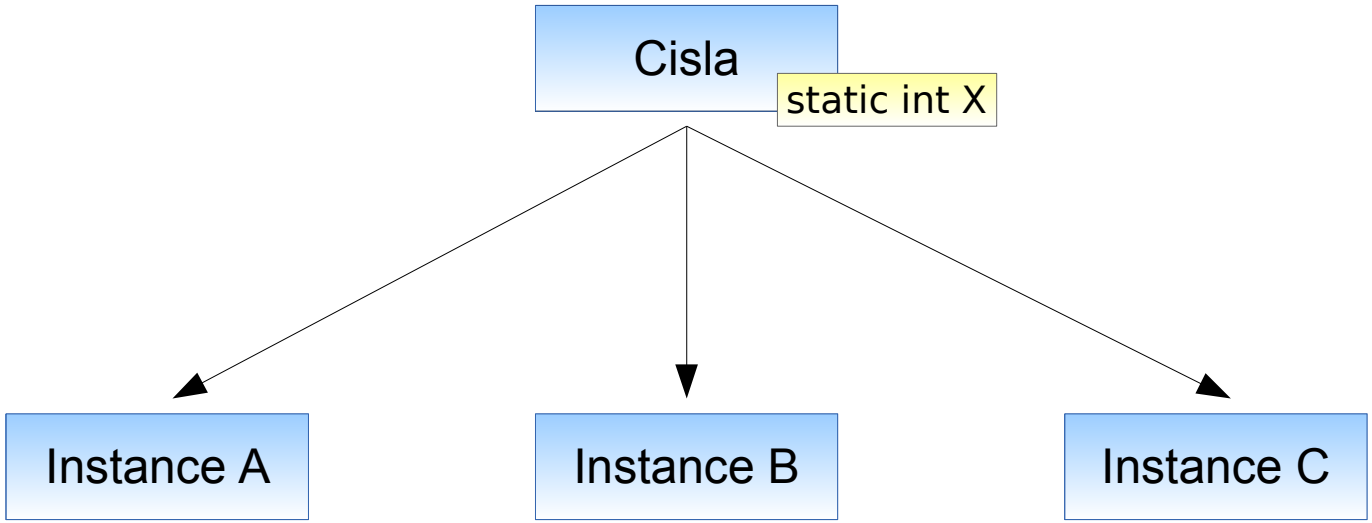



```
Cisla A, B, C;

A.X = 5
B.X = 7
C.X = 9
```

```
Cisla.X = ??
A.X = ??
B.X = ??
C.X = ??
A → this.X = ??
```

Statické vs instanční

- Vlastník → adresace this vs Main

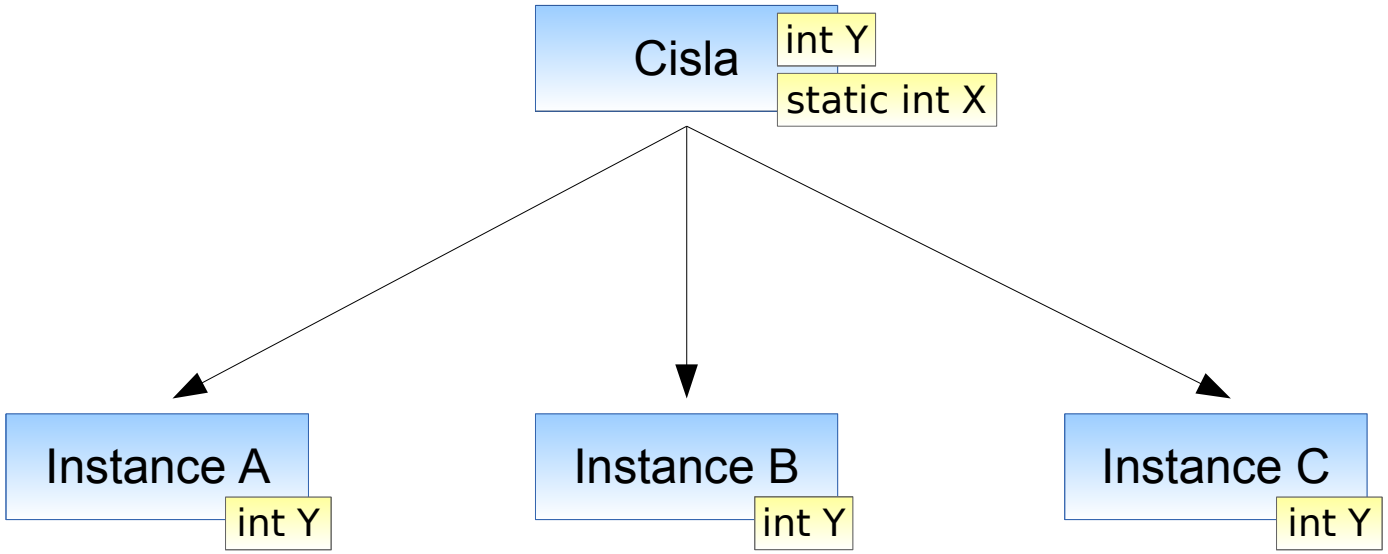



```
Cisla A, B, C;

A → this.X = 5
B → this.X = 7
C → this.X = 9
```

```
Cisla.X = ??
A.X = ??
B.X = ??
C.X = ??
A → this.A = ??
```

Statické vs instanční

- Vlastník → adresace this vs Main

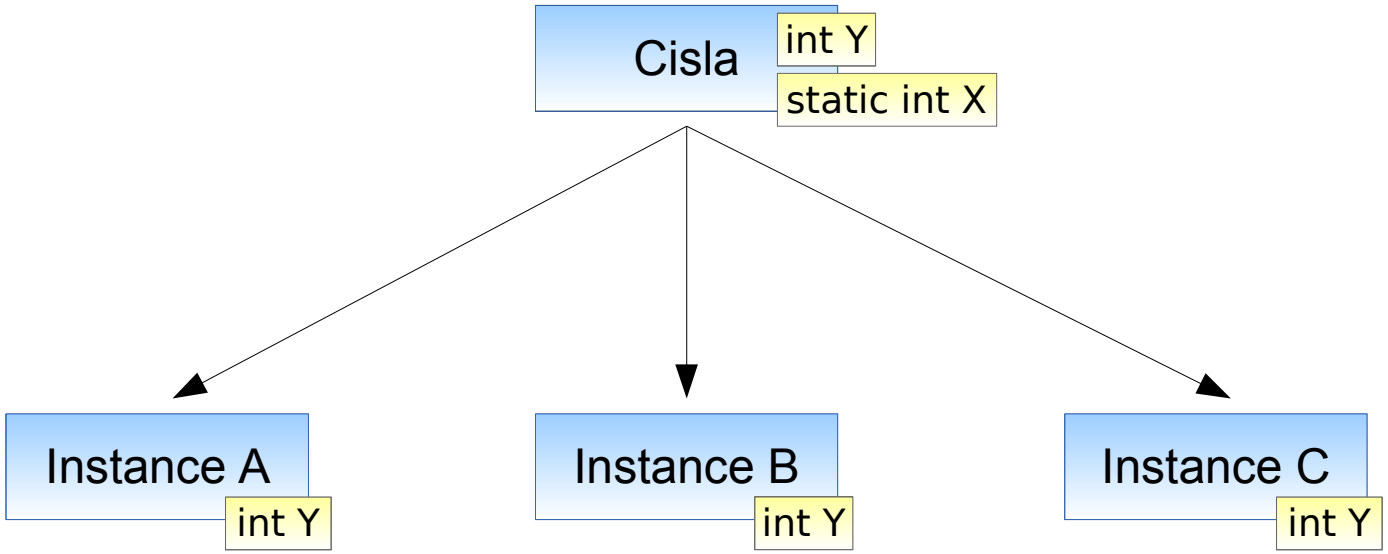



```
Cisla A, B, C;

A.Y = 5
B.Y = 7
C.Y = 9
```

```
Cisla.Y = ??
A.Y = ??
B.Y = ??
C.Y = ??
A → this.Y = ??
```

Statické vs instanční

- Vlastník → adresace this vs Main




```
Cisla A, B, C;

A → this.Y = 5
B → this.Y = 7
C → this.Y = 9
```

```
Cisla.Y = ??
A.Y = ??
B.Y = ??
C.Y = ??
A → this.Y = ??
```

Konvence

- Proměnné
- Instanční atributy
- Statické atributy
- Static final atributy
- Instanční metody
- Statické metody
- Static final metody
- Balíčky
- Třídy a rozhraní
- Abstract, Interface, Implem.
- Metody v rozhraních

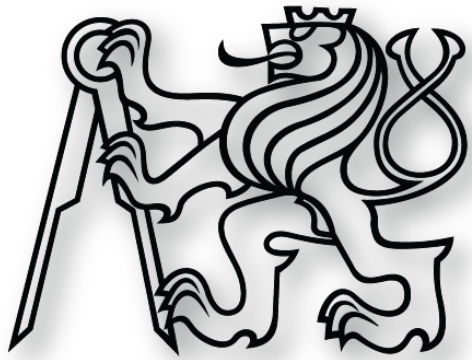
JavaDoc

- Speciální syntaxe pro psaní dokumentace
- Nad třídou, rozhraním, metodou, atributem
- Rozhraní místo implementace
- Podpora v IDE
- **NE** uvnitř metod

```
/**
 * Metoda provede soucet dvou zadanych cisel
 *
 * @param jeden scitanec
 * @param druhy scitanec
 * @return soucet
 */
public int secti(int a, int b);
```

Build

- Generuje řadu souborů
- *.class, *.jar, kopíruje konfiguraci atd.
- Generované **automaticky**
- **Neposílat, neukládat, nezálohovat**
- Zpravidla adresáře:
  - build/
  - target/
  - out/



Aktivita: Kvíz

Maximum 1 bod

Karel Čemus
cemuskar@fel.cvut.cz

Otázka 1

Určete výsledek programu

```
class Zvire {

  String jmeno = "";

  public Zvire(String jmeno) {
    jmeno = jmeno;
  }
}
```

```
Zvire pepicek = new Zvire("Pepicek");

System.out.println( pepicek.jmeno );
```

Otázka 2

Určete výsledek programu

```
class Zvire {


  private static int dalsi = 10;
  private int id = 0;

  public Zvire() {
    id = dalsi++;
  }
}
```

```
Zvire pepicek = new Zvire();

System.out.println( pepicek.id );
```

Otázka 3

Určete výsledek programu

```
class Zvire {

  static int dalsi = 10;
  int id = -10;

  public Zvire() {
    int dalsi = 0;
    Zvire.dalsi = dalsi + 1;
    id = dalsi;
  }
}

Zvire pepicek = new Zvire();
System.out.println( pepicek.id );
```

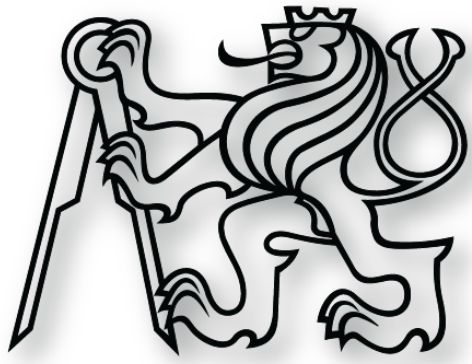
Otázka 4

Rekurze: Funkce, která vypíše zadané pole do konzole.

```
static void print(int[] pole) {

}
```

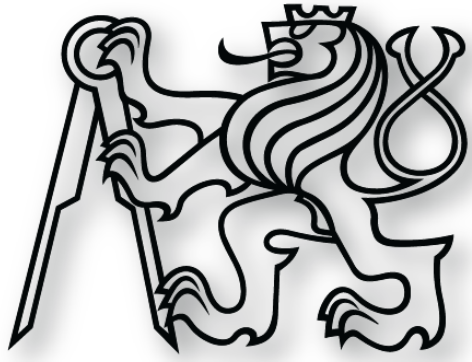
Nápověda:

- Potřebujete ještě jednu metodu
- Nezakládejte proměnné mimo metody



Odeslat emailem

Karel Čemus
cemuskar@fel.cvut.cz



Řešení

Karel Čemus
cemuskar@fel.cvut.cz

Otázka 1

Určete výsledek programu

```
class Zvire {

  String jmeno = "";

  public Zvire(String jmeno) {
    jmeno = jmeno;
  }
}
```

```
Zvire pepicek = new Zvire("Pepicek");

System.out.println( pepicek.jmeno );
```

Otázka 2

Určete výsledek programu

```
class Zvire {

  private static int dalsi = 10;
  private int id = 0;

  public Zvire() {
    id = dalsi++;
  }
}
```

```
Zvire pepicek = new Zvire();

System.out.println( pepicek.id );
```

Otázka 3

Určete výsledek programu

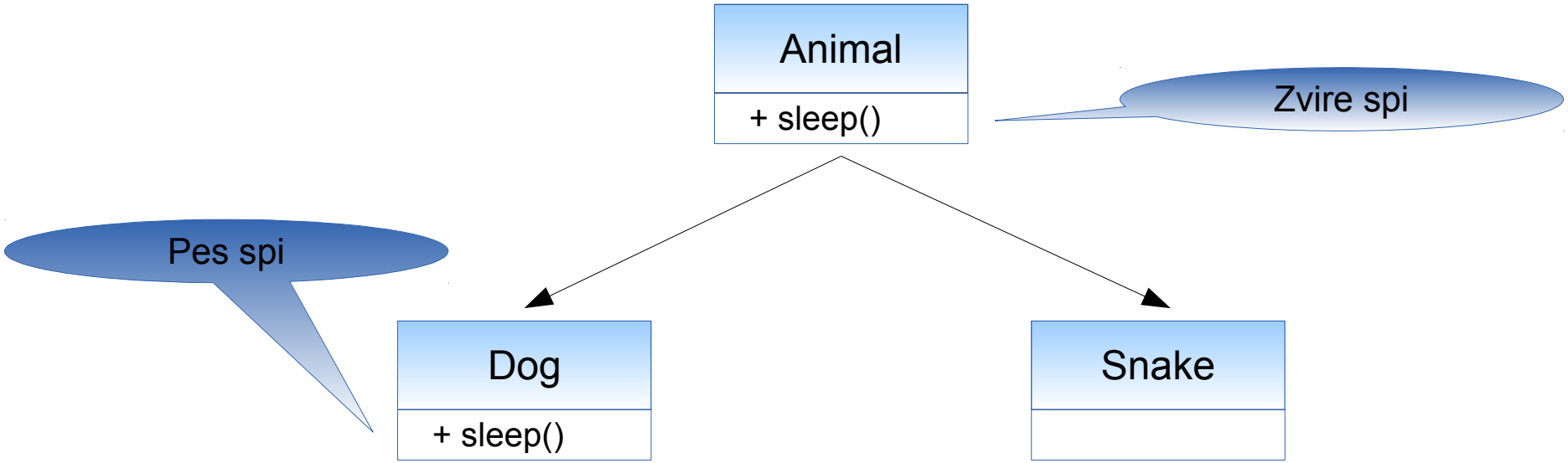
```
class Zvire {

  static int dalsi = 10;
  int id = -10;

  public Zvire() {
    int dalsi = 0;
    Zvire.dalsi = dalsi + 1;
    id = dalsi;
  }
}

Zvire pepicek = new Zvire();
System.out.println( pepicek.id );
```

Otázka 4

Rekurze: Funkce, která vypíše zadané pole do konzole.

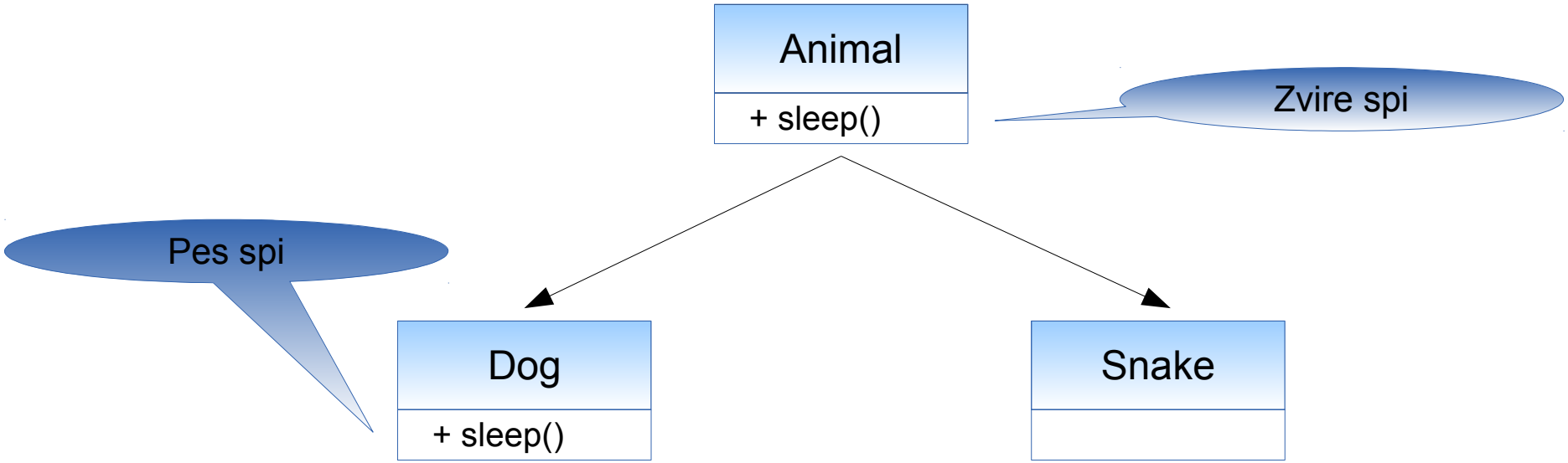
```
static int print(int[] pole) {
  print(pole, 0);
}

static int print(int[] pole, int index) {
  if ( index < pole.length ) System.out.println( pole[index] );
  else print(pole, index + 1);
}
```

Překrývání



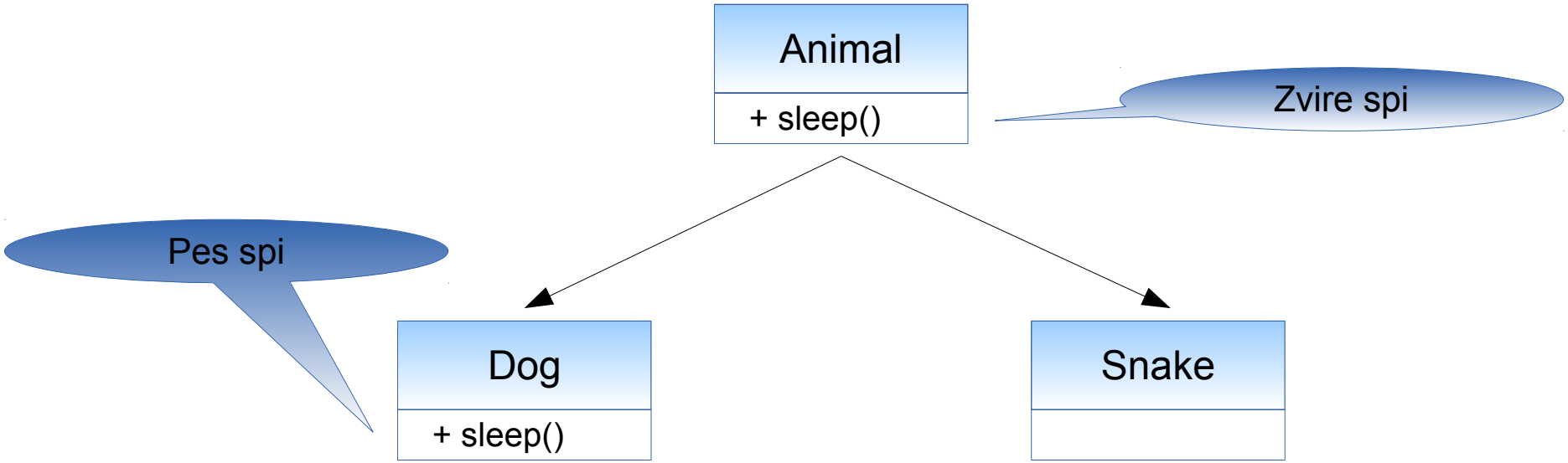

```
Animal animal = new Animal();

animal.sleep();
```

Překrývání



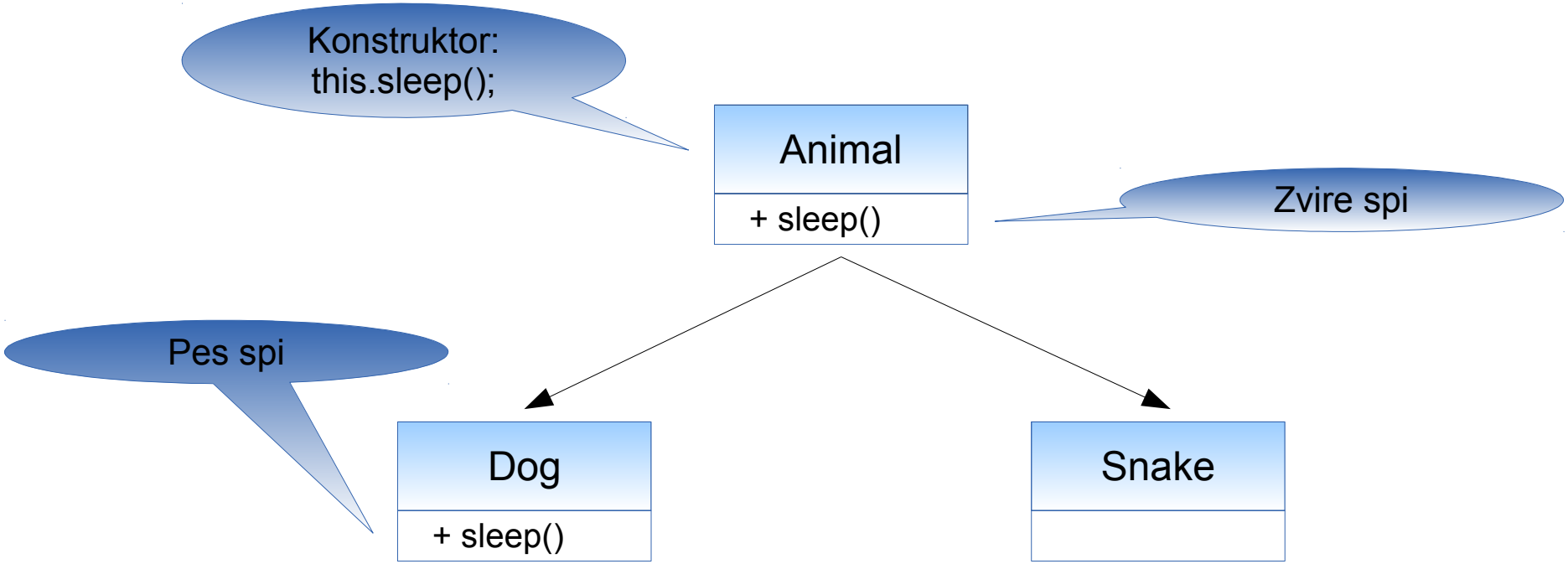

```
Animal animal = new Snake();

animal.sleep();
```

Překrývání



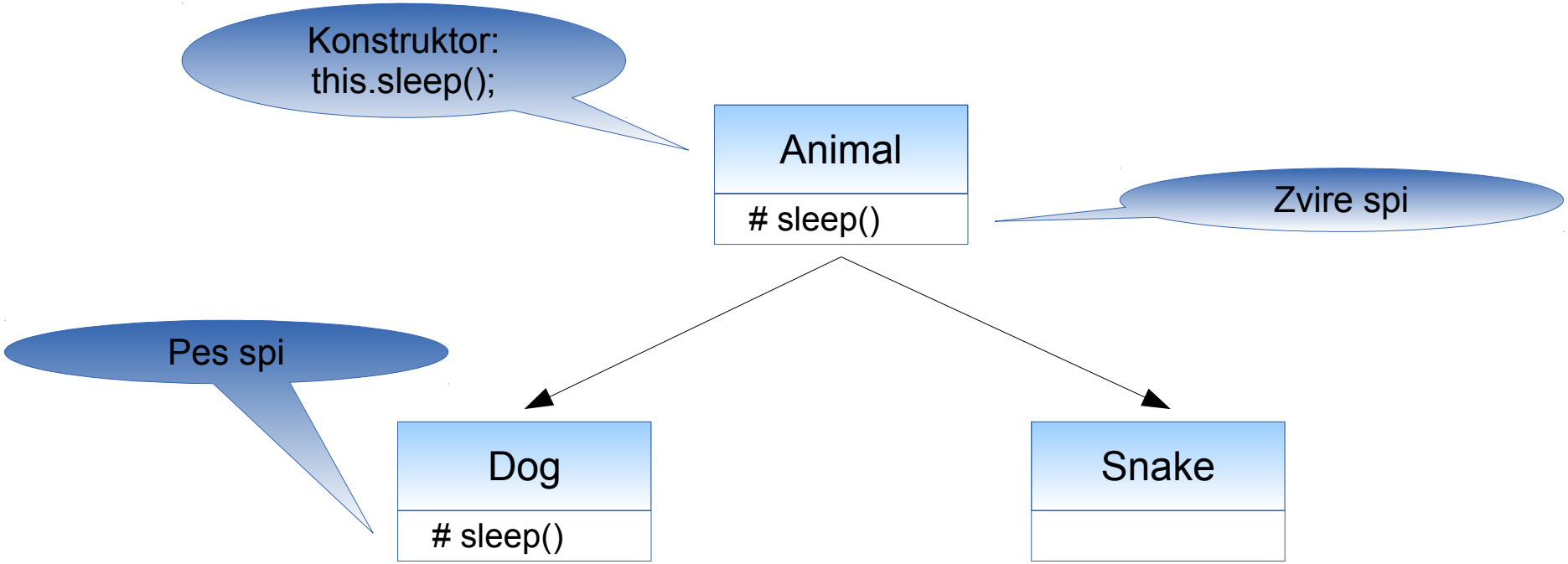

```
Animal animal = new Dog();

animal.sleep();
```

Překrývání



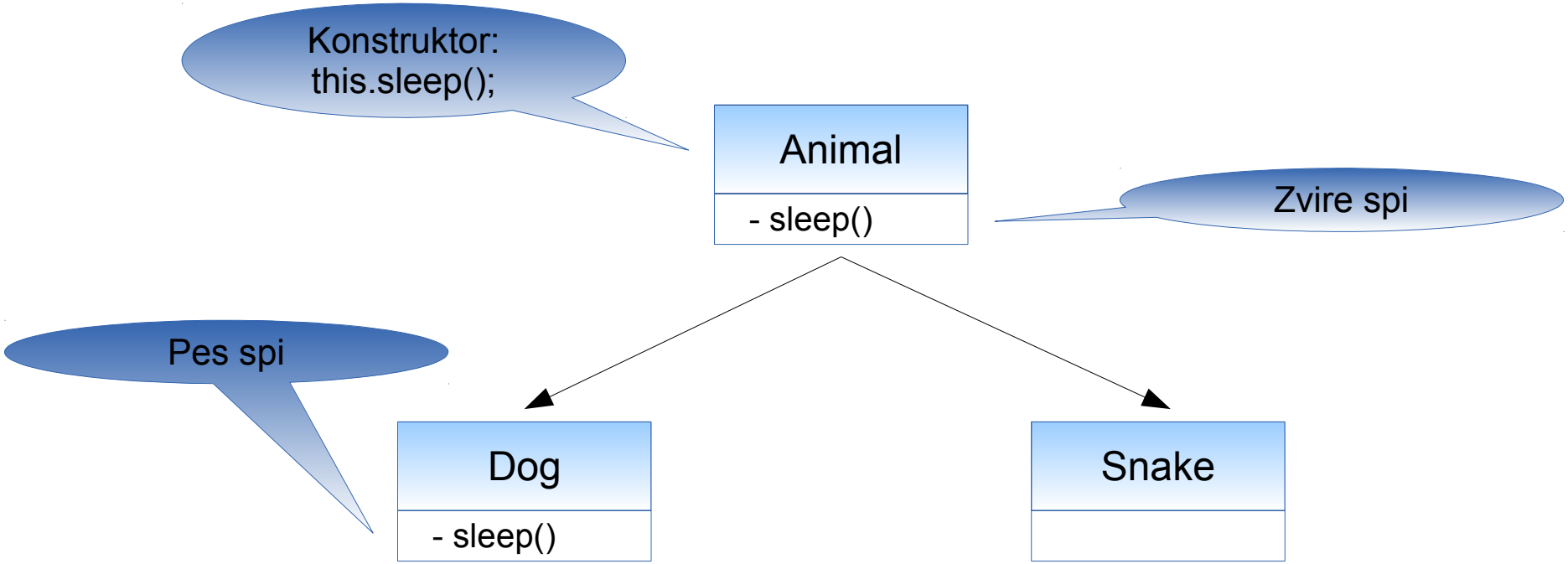

```
new Snake();
new Dog();
```

Překrývání



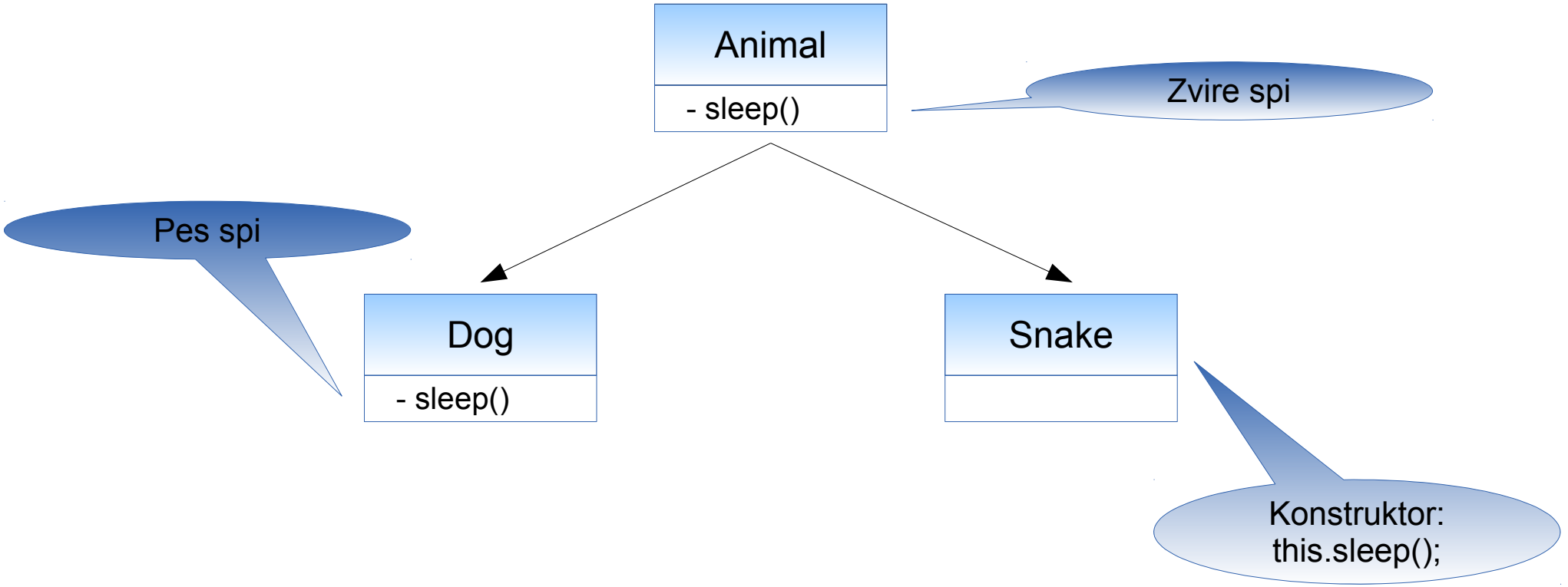

```
new Snake();
new Dog();
```

Překrývání




```
new Snake();
new Dog();
```

Překrývání

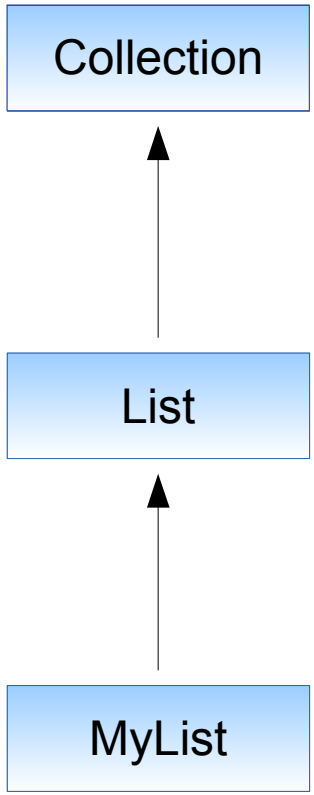



```
new Snake();
new Dog();
```

Substituční princip

- Překrývání nesmí porušit substituční princip!!!
- Možno upravit argumenty i návratovou hodnotu

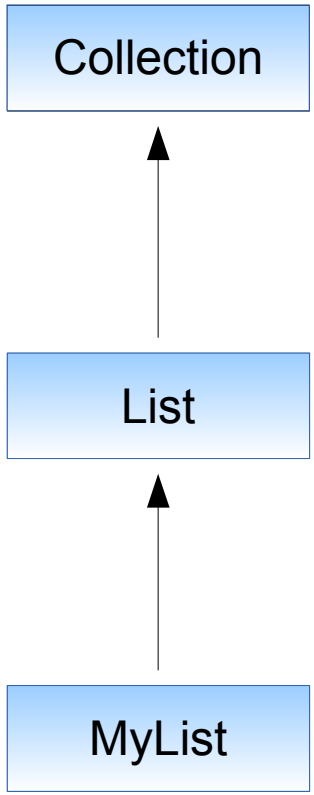
```
interface List {
  void addAll( List list );
}

interface MyList extends List {
  void addAll( Collection collection );
}
```



Substituční princip

- Překrývání nesmí porušit substituční princip!!!
- Možno upravit argumenty i návratovou hodnotu

```
interface List {
  List addAll( List list );
}

interface MyList extends List {
  MyList addAll( Collection collection );
}
```



Přetěžování

- Více metod se stejným názvem

```
interface ComplexNumber {
  ComplexNumber add( ComplexNumber number );
  ComplexNumber add( int real, int imaginary );
}
```

Přetěžování

- Více metod se stejným názvem

```
interface List {
  List insert( Object value );
  List insert( Object value, int position );
}
```

Přetěžování

- Více metod se stejným názvem

```
interface List {
  List insert( Dog value );
  List insert( Snake value );
}
```

Přetěžování

- Více metod se stejným názvem

NELZE!!!

```
interface List {
  List add( Animal animal );
  List add( Dog dog );
}
```

Přetěžování

- Více metod se stejným názvem

NELZE!!!

```
interface List {
  List       insert( Object value );
  Collection insert( Object value );
}
```

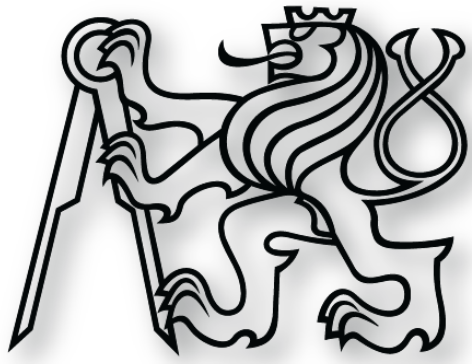
Polymorfismus

- Různé třídy, stejný kontrakt!
  - Stejné rozhraní
  - Stejný význam tříd

Test

- 10 otázek → 1 bod za otázku
- Na papír
- Viditelnost proměnných a atributů
- Překrývání metod
- Rozhraní, třídy
- Abstraktní metody a třídy
- Viditelnost atributů a metod
- Nalézt chybu v kódu

} kdy a proč, signatura metody



Kolekce

Karel Čemus
cemuskar@fel.cvut.cz