Dědičnost

Karel Richta a kol.

Katedra počítačů
Fakulta elektrotechnická
České vysoké učení technické v Praze

Objektové modelování, A7B36OMO
09/2016, Dědičnost

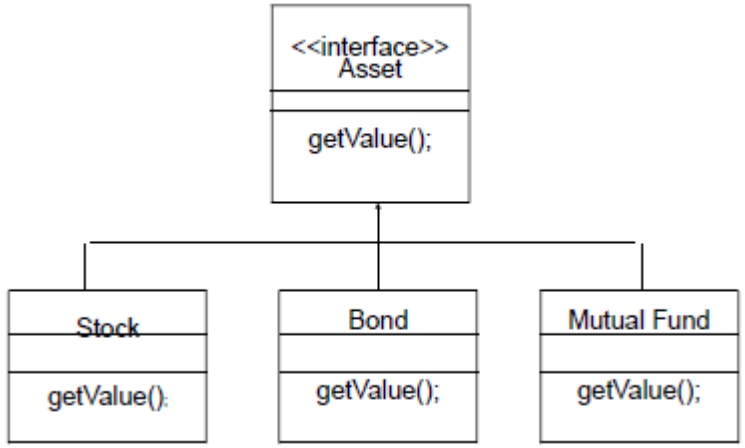
https://cw.fel.cvut.cz/wiki/courses/a7b36omo/start

Co je polymorfismus?

- Obecně se polymorfismem rozumí schopnost vyskytovat se v mnoha formách.
- Polymorfismus v Java:
  - Schopnost reference chovat se dle skutečně odkazovaného objektu.
  - Výběr metody, která se volá probíhá za běhu programu podle toho na který objekt reference odkazuje.
- V Java máme dostupný jak podtypový polymorfismus, tak parametrický.
  - Podtypový polymorfismus je realizován pomocí rozhraní a dědičnosti tříd.
  - Parametrický polymorfismus je realizován pomocí generik.
- Podtypový polymorfismus vychází z principu **subsumpce**.

Polymorfismus

- Je to způsob jak skrýt více implementací za jedno rozhraní.
- Umožňuje aby stejná zpráva byla obsloužena různě v závislosti na konkrétním objektu.



Příklad

- Mějme abstraktní třídu **Shape**, polymorfismus umožňuje programátorovi definovat různé metody **area** ve zděděných třídách **Circle**, **Rectangle**, ….
- Zavolání metody **area** na referenci typu **Shape** vrátí správný výsledek s ohledem na skutečně odkazovaný objekt.

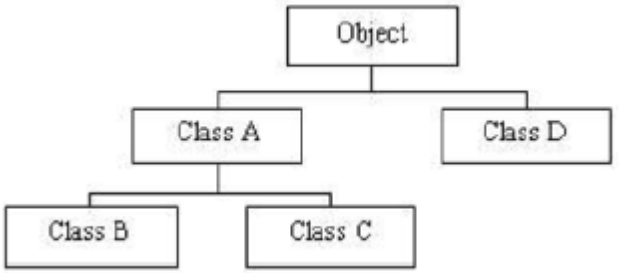
Generalizace

Forma asociace, kde třída sdílí strukturu anebo chování jiných/jiné tříd/třídy.

- Definuje hierarchii.
  - Jednoduchá dědičnost
  - Vícenásobná dědičnost
- Je to vztah typu **kind of**.

Dědičnost

- Mechanismus jak více specifické třídy přejímají chování a strukturu obecnějších tříd.
- Třída dědí atributy, operace a vazby.
- V Java jsou všechny třídy zděděné z třídy ***Object***.
- V Java je k dispozici jednoduchá dědičnost.
- Hierarchie tříd se znázorňuje obvykle stromem.



Dědičnost

- Výhoda dědičnosti v OOP - Znovupoužitelnost
- Chování (metody) definované v nadtřídě je dostupné ve všech podtřídách.
- Není nutné funkčnost metod definovat znovu.
- Podtřídy mohou implementaci metod měnit a přidávat metody nové.
- V Java používáme pro specifikaci dědičnosti klíčové slovo ***extends*** u třídy, která dědí.

Příklad

```
public class Person {
        protected String name;
        protected String address;
        /** Default constructor  */
        public Person(){
                System.out.println(“Inside Person:Constructor”);
                name = ""; address = "";
        }
}
public class Student extends Person {
        public Student(){
                System.out.println(“Inside Student:Constructor”);
        }
}
```

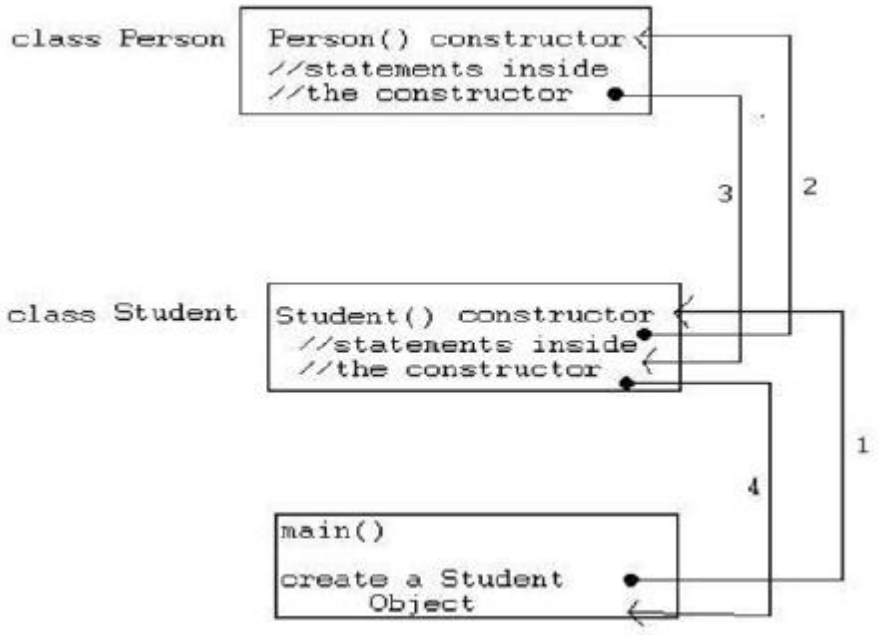
Příklad (pokr.)

- Metoda main:

```
public static void main( String[] args ){
        Student anna = new Student();
};
```

- Výstup programu:

```
Inside Person:Constructor
Inside Student:Constructor
```



Klíčové slovo “super”

- Podtřída může explicitně zavolat konstruktor nejbližší nadtřídy pomocí klíčového slova ***super***.

```
public Student(){
        super( "SomeName", "SomeAddress" );
        System.out.println("Inside Student:Constructor");
}
```

- Několik věcí na které je třeba myslet při používání klíčového slova super pro volání konstruktoru nadtřídy:
  - *super()* musí být jako první příkaz konstruktoru.
  - *super()* může být použito pouze v konstruktoru.
  - Ve stejném konstruktoru nemůžeme použít *this()* a *super()* zároveň.

Klíčové slovo “super” (pokr.)

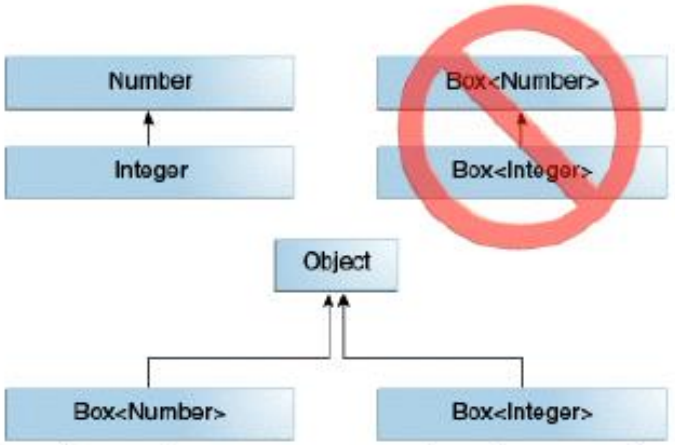
- Další použití je pokud chceme přistupovat ke členským proměnným nadtřídy.

```
public Student() {
      super.name = “somename”;
      super.address = “some address”;
      super.s();
}
```

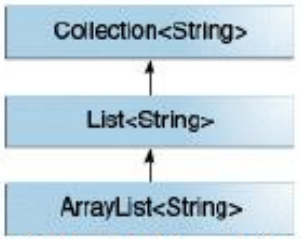
Generické typy

```
public interface Pair<K, V> {
        public K getKey();
        public V getValue();
}
public class OrderedPair<K, V> implements Pair<K, V> {
        private K key;
        private V value;
        public OrderedPair(K key, V value) {
                this.key = key;
                this.value = value;
        }
        public K getKey() { return key; }
        public V getValue() { return value; }
}
```

Generické typy a podtypy



`Box<Integer>` is not a subtype of `Box<Number>` even though `Integer` is a subtype of `Number`.



A sample `Collections` hierarchy

Literatura

- http://fi.muny.cz/data/PB006/PB006_slides2009.pdf
- https://edux.feld.cvut.cz/courses/X36OBP/_media/lectures/02-subtyping.pdf

The End