PB162 Programovanie v jazyku Java I

4. cviko

Jakub Smadiš

7. marca 2022

Rovnaké zápisy

Rovnaké zápisy

Defaultná hodnota `double`:

```
double realNumber;
```

Rovnaké zápisy

Defaultná hodnota `double`:

```
double realNumber;
double realNumber = 0.0;
```

Rovnaké zápisy

Defaultná hodnota double:

```
double realNumber;
double realNumber = 0.0;
```

Defaultná hodnota objektovej premennej:

```
Dog max;
```

Rovnaké zápisy

Defaultná hodnota double:

```
double realNumber;
double realNumber = 0.0;
```

Defaultná hodnota objektovej premennej:

```
Dog max;
Dog max = null;
```

Rovnaké zápisy

Defaultná hodnota double:

```
double realNumber;
double realNumber = 0.0;
```

Defaultná hodnota objektovej premennej:

```
Dog max;
Dog max = null;
```

Pri výpise objektu sa použije metóda:

```
System.out.println(object);
```

Rovnaké zápisy

Defaultná hodnota double:

```
double realNumber;
double realNumber = 0.0;
```

Defaultná hodnota objektovej premennej:

```
Dog max;
Dog max = null;
```

Pri výpise objektu sa použije metóda:

```
System.out.println(object);
System.out.println(object.toString());
```

Rovnaké zápisy

Defaultná hodnota double:

```
double realNumber;
double realNumber = 0.0;
```

Defaultná hodnota objektovej premennej:

```
Dog max;
Dog max = null;
```

Pri výpise objektu sa použije metóda:

```
System.out.println(object);
System.out.println(object.toString());
```

Ak trieda Dog nemá žiaden konštruktor,

Rovnaké zápisy

Defaultná hodnota double:

```
double realNumber;
double realNumber = 0.0;
```

Defaultná hodnota objektovej premennej:

```
Dog max;
Dog max = null;
```

Pri výpise objektu sa použije metóda:

```
System.out.println(object);
System.out.println(object.toString());
```

Ak trieda Dog nemá žiaden konštruktor, vytvorí sa defaultný:

Rovnaké zápisy

Defaultná hodnota double:

```
double realNumber;
double realNumber = 0.0;
```

Defaultná hodnota objektovej premennej:

```
Dog max;
Dog max = null;
```

Pri výpise objektu sa použije metóda:

```
System.out.println(object);
System.out.println(object.toString());
```

Ak trieda Dog nemá žiaden konštruktor, vytvorí sa defaultný:

```
public Dog() {
}
```

Konštruktor s parametrami alebo bez?

1. Konštruktor bez parametrov

```
public Dog() { } // v triede Dog

Dog staryDunco = new Dog(); // v inej triede
staryDunco.setName("Dunco");
staryDunco.setAge(13);
```

2. Konštruktor s parametrami

```
public Dog(String ourName, int ourAge) {
  name = ourName;
  age = ourAge; // privatne atributy su name a age
}

Dog staryDunco = new Dog("Dunco", 13);
```

Má vôbec niekedy zmysel používať settery?

Konštruktor s parametrami alebo bez?

1. Konštruktor bez parametrov

```
public Dog() { } // v triede Dog

Dog staryDunco = new Dog(); // v inej triede
staryDunco.setName("Dunco");
staryDunco.setAge(13);
```

2. Konštruktor s parametrami

```
public Dog(String ourName, int ourAge) {
  name = ourName;
  age = ourAge; // privatne atributy su name a age
}

Dog staryDunco = new Dog("Dunco", 13);
```

Má vôbec niekedy zmysel používať settery?

- ak chceme hodnoty neskôr zmeniť (premenujeme psa)
- niektoré parametry sú voliteľné (pes nemá špeciálne schopnosti, no môžme mu nejakú pridať)

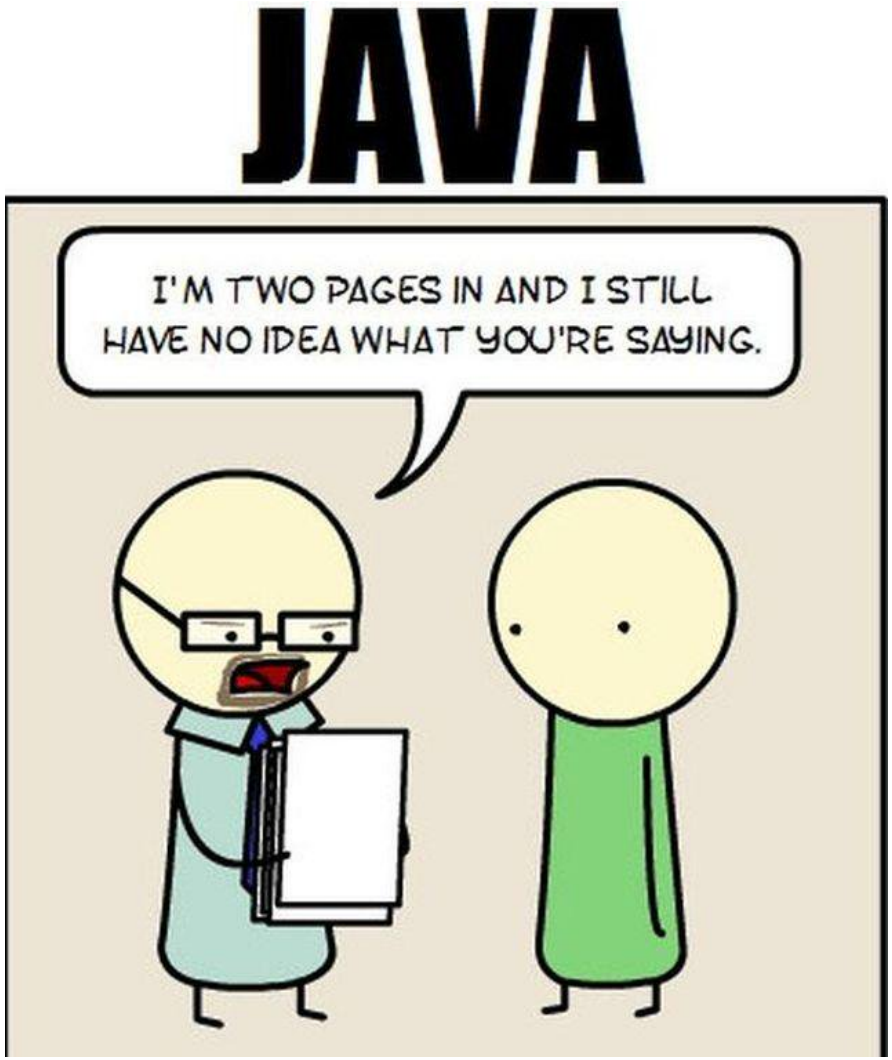
Výnimky

Pri nezvyčajných (výnimočných) situáciach Java vyhadzuje výnimky.
Výnimky budeme preberať neskôr, zatiaľ v skratke:

- nastávajú pri situáciach, ktoré by nemali nastať (delenie nulou)
- výnimka spôsobí pád programu
- `null.toString()` vyhodí `NullPointerException`

Java je ukecaná

Java je ukecaná



Konvencie v Jave

- balíčky (packages)

Konvencie v Jave

- balíčky (packages)

  com.sun.eng, cz.muni.fi

Konvencie v Jave

- balíčky (packages)

  com.sun.eng, cz.muni.fi
- triedy (classes)

Konvencie v Jave

- balíčky (packages)

  com.sun.eng, cz.muni.fi
- triedy (classes)

  Dog, VeryBigDog, Object, String

Konvencie v Jave

- balíčky (packages)

  com.sun.eng, cz.muni.fi
- triedy (classes)

  Dog, VeryBigDog, Object, String
- premenné (variables)

Konvencie v Jave

- balíčky (packages)

  com.sun.eng, cz.muni.fi
- triedy (classes)

  Dog, VeryBigDog, Object, String
- premenné (variables)

  int i, Dog maxipesFik

Konvencie v Jave

- balíčky (packages)

  com.sun.eng, cz.muni.fi
- triedy (classes)

  Dog, VeryBigDog, Object, String
- premenné (variables)

  int i, Dog maxipesFik
- metódy (methods)

Konvencie v Jave

- balíčky (packages)

  com.sun.eng, cz.muni.fi
- triedy (classes)

  Dog, VeryBigDog, Object, String
- premenné (variables)

  int i, Dog maxipesFik
- metódy (methods)

  Vertex getVertex(), void runFast(Dog runningDog)

Konvencie v Jave

- balíčky (packages)

  com.sun.eng, cz.muni.fi
- triedy (classes)

  Dog, VeryBigDog, Object, String
- premenné (variables)

  int i, Dog maxipesFik
- metódy (methods)

  Vertex getVertex(), void runFast(Dog runningDog)
- konštanty (constants)

Konvencie v Jave

- balíčky (packages)

  com.sun.eng, cz.muni.fi
- triedy (classes)

  Dog, VeryBigDog, Object, String
- premenné (variables)

  int i, Dog maxipesFik
- metódy (methods)

  Vertex getVertex(), void runFast(Dog runningDog)
- konštanty (constants)

  static final int MAX_WIDTH = 99

Konvencie v Jave

- balíčky (packages)

  com.sun.eng, cz.muni.fi
- triedy (classes)

  Dog, VeryBigDog, Object, String
- premenné (variables)

  int i, Dog maxipesFik
- metódy (methods)

  Vertex getVertex(), void runFast(Dog runningDog)
- konštanty (constants)

  static final int MAX_WIDTH = 99

Podrobnejšie informácie nájdete na stránke Oracle.

Kľúčové slovo this

Máme nasledovný kód:

```
public class Dog {
  private int age; // global variable
  ...
  public int countAge() {
    int age;  // local variable
    ...
    print(age); // which one is printed?
  }
```

Kľúčové slovo this

Máme nasledovný kód:

```
public class Dog {
  private int age; // global variable
  ...
  public int countAge() {
    int age;  // local variable
    ...
    print(age); // which one is printed?
  }
```

- Ako zavolať lokálnu premennú?

Kľúčové slovo this

Máme nasledovný kód:

```
public class Dog {
  private int age; // global variable
  ...
  public int countAge() {
    int age;  // local variable
    ...
    print(age); // which one is printed?
  }
```

- Ako zavolať lokálnu premennú?

  ```
  age
  ```

Kľúčové slovo this

Máme nasledovný kód:

```
public class Dog {
  private int age; // global variable
  ...
  public int countAge() {
    int age;  // local variable
    ...
    print(age); // which one is printed?
  }
```

- Ako zavolať lokálnu premennú?

  ```
  age
  ```

- Ako zavolať globálnu premennú?

Kľúčové slovo this

Máme nasledovný kód:

```
public class Dog {
  private int age; // global variable
  ...
  public int countAge() {
    int age;  // local variable
    ...
    print(age); // which one is printed?
  }
```

- Ako zavolať lokálnu premennú?

  ```
  age
  ```

- Ako zavolať globálnu premennú?

  ```
  this.age
  ```

Preťaženie konštruktorov

Constructor/method overloading

- rovnaký názov, rôzne parametre
- nemusí ich byť len rôzny počet

```
class Dog {
  public Dog(int age) { }
  public Dog() {
    Dog(4);
  }
```

Preťaženie konštruktorov

Constructor/method overloading

- rovnaký názov, rôzne parametre
- nemusí ich byť len rôzny počet

```
class Dog {
  public Dog(int age) { }
  public Dog() {
    Dog(4);
  }
```

Volanie konštruktoru v konštruktore takto nefunguje!
Treba použiť známe kľúčové slovo.

Preťaženie konštruktorov

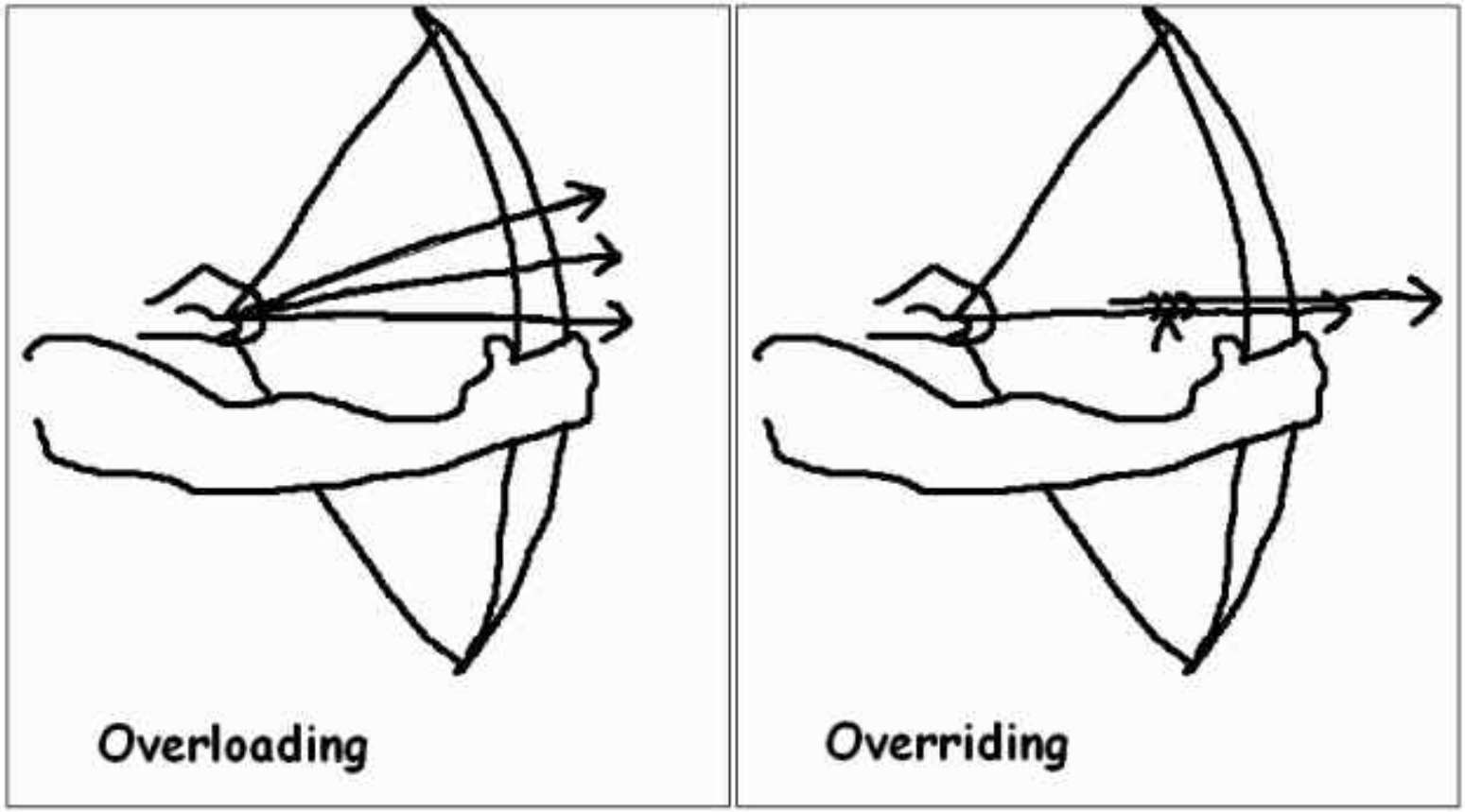
Constructor/method overloading

- rovnaký názov, rôzne parametre
- nemusí ich byť len rôzny počet

```
class Dog {
  public Dog(int age) { }
  public Dog() {
    this(4);
  }
```

Volanie konštruktoru v konštruktore takto nefunguje!
Treba použiť známe kľúčové slovo.

Overloading & overriding



Hint na záver

`void divide(int depth)` je na pár riadkov:

- skontrolujem podmienky
- rozdelím trojuholník
- zavolám rozdelenie trojuholníka na menších podtrojuholníkoch