PB162 Programování v jazyce Java

3. cvičenie

dedičnosť *(inheritance)*

- 2. pilier OOP
- spôsob vyjadrenia vzťahu nadtyp–podtyp = generalizácia–špecializácia
- vzťah **ISA**
- subtype **is a** (special case of) supertype
- dnes
  - trieda Object a jej metódy
  - rozhrania (*interfaces*)

trieda Object

- predok všetkých tried
- každy objekt **ISA** object
- každá trieda je potomok triedy Object
- zdedené metódy
  - equals – relácia ekvivalencia
  - hashCode – hash (odtlačok, *digest*) objektu
  - toString – info o objekte

rozhrania *(interfaces)*

- prvok **dedičnosti** *(inheritance)*
  - rozhrania vyjadrujú čiste funkčný aspekt tohto vzťahu
- vyjadrujú „sľubovanú“ funkcionalitu

  = povinné metódy
- **implementujúca** trieda sa k nej „hlási“

rozhrania (*interfaces*)

- syntax
  - trieda A definujúca rozhranie
    - public intreface A
  - trieda B implementujúca rozhranie A
    - public class B implements A
- výhody
  - skrytie implementačných detailov
  - a zároveň definícia (dohoda) komunikačného *rozhrania*

rozhranie ≠ rozhranie

- *rozhranie*
  - vopred daný/dohodnutý spôsob komunikácie
  - blízke významu slova **protokol**
  - umožňuje delenie zložitosti na jednoduchšie podcelky
- rozhranie ako prvok jazyka Java
  - budeme nazývať slovom **interface**
- interface slúži na definíciu *rozhrania*

interface

- definuje názvy metód, ich parametre a návratové hodnoty
  - hlavičky metód
- implementujúca trieda určuje konkrétnu implementáciu týchto metód
  - hlavičkám doplní telo

interface

- výhody
  - vytvorenie nového typu
  - môže byť parametrom/návratovou hodnotou metód v iných triedach/rozhraniach
  - nevyhovujúcu implementáciu možno nahradiť lepšou zmenou jedného riadku kódu
- príklady
  - rozhranie Comparable
  - vyhľadávací strom SearchTree

rozhranie Comparable

- definuje prirodzené lineárne usporiadanie
- lineárne usporiadanie
  - reflexívna, antisymetrická a tranzitívna relácia
- jediná metóda compareTo()
- o1.compareTo(o2) vracia

  -1 ak o1 < o2

  0 ak o1.equals(o2)

  1 ak o1 > o2

rozhranie Comparable

- môžeme definovať pre akúkoľvek triedu, pre ktorej prvky existuje prirodzené usporiadanie
  - (čísla podľa veľkosti, reťazce lexikograficky...)
- univerzálny
  - vyhľadávací strom
  - radiaci algoritmus
  - ...

pre akýkoľvek objektový typ

SearchTree

- binárny vyhľadávací strom
- ohodnotené uzly
- pre každý uzol platí, že v ľavom podstrome sú hodnoty menšie ako v uzle, v pravom väčšie
- čas pridania hodnoty do stromu, dopytu na prítomnosť, odobrania hodnoty je **logaritmická** pre vyvážený strom
- vyvažovaním sa zaoberať nebudeme

SearchTree – implementácia

- využíva rozhranie Comparable na definíciu usporiadania
- jediná trieda Node
  - ohodnotenie typu Comparable
  - pravý a ľavý podstrom
    - odkaz na koreň typu Node
  - metódy addSubnode, hasSubnode, removeSubnode

SearchTree ako interface

- rozhranie definuje funkcionalitu
  - pridanie, odobratie, zistenie príslušnosti
- naša trieda – UnbalancedSearchTree implements SearchTree
- AvlTree, RedBlackTree...
- domáca úloha
  - AvlTree alebo RedBlackTree
  - nepovinná, bonus. body
  - doporučujem aspoň skúsiť