Polynómy

Motivácia: V Jave nájdeme množstvo hotových tried na všetky možné účely. Avšak asi tvorcovia Javy nemali až tak radi matematiku. V Jave je totiž relatívne málo tried podporujúcich matematické objekty ...

Nedávno sa u kolegov matematikov (pri hraní sa s veľmi zložitými výpočtami) objavila potreba po nejakej triede, ktorá by umožnila symbolické výpočty s polynómami nad poľom reálnych čísel. Keďže nič také v Jave štandardne nie je, neostáva nám nič iné, ako im skúsiť pomôcť a požadovanú triedu pre nich naprogramovať.

Zadanie: Vytvorte triedu *Polynom*, ktorá:

- bude implementovať rozhranie *java.lang.Comparable* (polynómy vieme porovávať)
- bude umiestnená v balíku *sk.upjs.matematika*

Objekt triedy *Polynom* bude reprezentovať nejaký polynóm nad poľom reálnych čísel. Príkladom takého polynómu je napríklad polynóm: $2x^{45} – 4x^{23} + x^3 + 1{,}5x^2 - 5$. Ľahko vidieť, že taký polynóm je vlastne len súčtom niekoľkých členom tvaru $a.x^b$, pričom reálne číslo *a* nazývame koeficientom člena a prirodzené číslo *b* exponentom. Členy, ktorých koeficient je 0 zo zápisu polynómu zvyčajne vynechávame. Tak ako u objektov triedy *String*, nebude možné zmeniť reprezentovaný polynóm – t.j. polynóm (hodnoty koeficientov a exponentov) reprezentovaný objektom bude po celú dobu života objektu nemenný. Všetky inštančné premenné a metódy triedy s výnimkou metód uvedených v zadaní musia byť neverejné.

Požadované konštruktory triedy *Polynom*:

- `public Polynom()`
- `public Polynom(double k, int e)`
- `public Polynom(double[] koeficienty)`

Prvý konštruktor vytvorí objekt reprezentujúci polynóm 0 (presnejsie $0.x^0$). Druhý konštruktor vytvorí polynóm reprezentujúci jediný člen $k.x^e$. Posledný tretí konštruktor vytvorí polynóm na základe hodnôt v poli koeficientov. Presnejšie *i*-ty prvok v poli *koeficienty* určuje koeficient v člene s exponentom *i* (t.j. pri x^i).

Implementujte nasledovné inštančné metódy:

- `public double vyhodnot(double hodnota)` - vráti hodnotu, ktorú dostaneme dosadením hodnoty *hodnota* za premennú *x* polynómu.
- `public double getKoeficient(int exponent)` – vráti koeficient v člene s exponentom *exponent*.
- `public int getStupen()` - vráti stupeň polynómu, t.j. najväčší taký exponent, že člen s týmto exponentom má nenulový koeficient. Stupeň polynómu $0.x^0$ je *0*.

Prekryte nasledovné inštančné metódy zdedené z triedy *java.lang.Object*:

- `public String toString()` – vráti referenciu na novovytvorený reťazec vyjadrujúci zápis polynómu. Členy vypíšte tak, aby ich exponenty poli v klesajúcom poradí. Členy s nulovým exponentom nevypisujte. Ako symbol umocnenia použite znak ^. Teda x^6 vypíšte ako x^6.
- `public boolean equals(Object obj)` – vráti *true*, ak objekt *obj* reprezentuje rovnaký polynóm – t.j. ak objekt *obj* je inštanciou triedy *Polynom* a koeficienty pri všetkých členoch sú rovnaké ako pri polynóme *this*.

Implementujte nasledovné inštančné metódy:

- `public Polynom pripocitaj(Polynom p)` – vráti referenciu na novovytvorený objekt reprezentujúci polynóm, ktorý vznikne súčtom polynóm *this* a polynómu *p*.
- `public Polynom odpocitaj(Polynom p)` - vráti referenciu na novovytvorený objekt reprezentujúci polynóm, ktorý vznikne odpočítaním polynómu *p* od polynómu *this*.

Pripomeňme, že súčet 2 polynómov vypočítame sčítaním koeficientov členov s rovnakým exponentom.

Implementujte metódu vyžadovanú rozhraním *java.lang.Comparable*:

- `public int compareTo(Object o)` - vráti -1 (0, 1) ak polynóm reprezentovaný polynómom *o* je lexikograficky väčší (rovnaký, menší). Lexikografické porovnanie 2 polynómov dostaneme porovnaním koeficientov pri členoch s najväčším takým exponentom, že tieto koeficienty sú v porovnávaných polynómoch rôzne. Môžete predpokladať, že táto metóda bude vždy s parametrom, ktorý je pretypovateľný na objekt triedy *Polynom*.

Implementujte nasledovné inštančné metódy:

- `public Polynom derivuj()` – vráti referenciu na novovytvorený objekt reprezentujúci polynóm, ktorý vznikne zderivovaním polynómu
- `public Polynom integruj()` - vráti referenciu na novovytvorený objekt reprezentujúci polynóm, ktorý vznikne integrovaním polynómu

Implementujte statickú metódu triedy polynóm:

- `public static Polynom sucet(Polynom p1, Polynom p2)` – vráti referenciu na novovytvorený objekt reprezentujúci polynóm, ktorý vznikne súčtom polynómu *p1* a *p2*

Niekoľko rád ku riešeniu:

- existuje viacero spôsobov, ako interne reprezentovať polynómy:
  - To najjednoduchšie riešenie je vytvoriť si pole koeficientov. V tomto poli si budeme potom na indexe *i* pamätať koeficient pri člene s exponentom *i*. Nevýhoda tohto riešenia je, že vždy potrebujeme pole dĺžky aspoň o 1 väčšej ako je stupeň polynómu. Napríklad, za polynómom x^{6321} sa bude skrývať pole dĺžky *6322* a to je už plytvanie pamäťou.
  - Efektívnejšie riešenie je implementačne o dosť náročnejšie. Vystačíme si v ňom s 2 poľami dĺžky rovnej počtu členov s nenulovým koeficientom. V prvom poli (intov) si budeme pamätať exponenty členov a v druhom poli (doubleov) si uložíme koeficienty pri týchto členoch. Ak naviac členy budeme mať uložené usporiadané, budeme vedieť jednotlivé metódy implementovať výrazne efektívnejšie (ale opäť implementačne náročnejšie).
- spomeňte si na *instanceof*

Demo:

```
public static void main(String[] args) {
      Polynom p1 = new Polynom(3, 5);

      double[] exps = {5, 7.5, 0, 2.5, 3};
      Polynom p2 = new Polynom(exps);

      // Test jednej z chyb implementacie
      for(int i=0; i<exps.length; i++)
        exps[i] = 0;

      System.out.println("p1: " + p1);
      System.out.println("p2: " + p2);

      System.out.println("p2(x = 8) = " + p2.vyhodnot(8));
      System.out.println("Stupen p2 je " + p2.getStupen());

      System.out.println("p1 ? p2 :" + p1.compareTo(p2));

      Polynom sucet = p1.pripocitaj(p2);
      System.out.println("p1 + p2 : " + p3);
}
```