Polsemestrálny test
Zadanie

Ústav informatiky
Prírodovedecká fakulta
UPJŠ v Košiciach

Dvakrát meraj (rozmýšľaj), raz rež (programuj)

Pravidlá a informácie:

- čas na riešenie úloh je **90 minút**
- nie je dovolená žiadna (elektronická aj neelektronická) komunikácia s kýmkoľvek okrem dozoru
- nie je dovolené používať žiadne zdroje ani materiály okrem oficiálneho ťaháku
- nie je dovolené používať žiadnu inú aplikáciu než Eclipse (s výnimkou webového prehliadača pri odosielaní riešenia)
- svoje riešenia odovzdávajte cez systém Moodle (http://moodle.ics.upjs.sk/)

Upozornenie:

- Skontrolujte si, či máte k projektu pripojenú knižnicu *jpaz2.jar*.

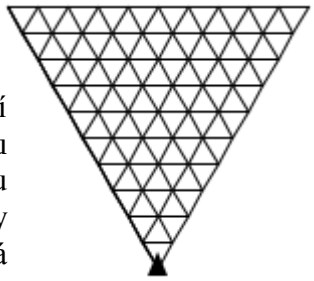
Ktoré úlohy treba riešiť:

Cieľom úloh je vytvoriť triedu *Midtermarka*, ktorá rozširuje triedu *Turtle*. Z prvej trojice úloh si **vyberte len 2 úlohy**, ktoré **budete riešiť !!!** To, ktoré úlohy ste sa rozhodli riešiť, uveďte v komentári pri odosielaní riešenia cez Moodle.

V *Časti 2* je len jedna úloha, t.j. v tejto časti nie je možný výber úloh.

Časť 1 (dve úlohy z troch)

Pyramída (10 bodov)



Do triedy *Midtermarka* pridajte metódu *pyramida*, ktorá nakreslí pyramídu z rovnostranných trojuholníkov so zadaným počtom radov a so zadanou dĺžkou strany malých rovnostranných trojuholníkov. Korytnačka sa na začiatku nachádza v jednej zo strán trojuholníka a je nasmerovaná v smere výšky pyramídy. Po nakreslení pyramídy nech je korytnačka na pozícii a je natočená v smere, ako bolo pri volaní metódy.

```
public void pyramida(int pocetRadov, double strana)
```

Rozklad (10 bodov)

Z matematiky je známe, že každé nenulové celé číslo *n* vieme jednoznačne zapísať v tvare $a.2^i$, kde a je nepárne číslo. Do triedy *Midtermarka* pridajte metódu *exponentRozkladu*, ktorá pre zadané číslo vráti exponent *i* (pri dvojke) v tomto jedinečnom rozklade – t.j. exponent najväčšej mocniny dvojky, ktorá delí zadané číslo. Napríklad pre číslo *100* má metóda vrátiť *2* lebo $100 = 25.2^2$, pre číslo 27 má vrátiť *0* pretože $27 = 27.2^0$, pre číslo *40* číslo *3* pretože $40 = 5.2^3$. Môžete predpokladať, že parametrom je kladné číslo.

Rada: Na vyriešenie tejto úlohy netreba použiť žiadne matematické finty. Jeden z prístupov môže byť postupne skúšať rôzne hodnoty *i* a vybrať najväčšiu vyhovujúcu. V tomto prípade nemá zmysel skúšať mocniny dvojky, ktoré sú väčšie ako zadané číslo. Iný prístup je skúšať deliť zadané číslo dvojkou, kým sa dá – t.j. kým nám neostane nepárne číslo.

```
public int exponentRozkladu(int cislo)
```

Spoločný prefix (10 bodov)

Prefixom reťazca nazývame ľubovoľný jeho podreťazec, ktorým tento reťazec začína. Napríklad reťazec `"skola"` má tieto prefixy: `""`, `"s"`, `"sk"`, `"sko"`, `"skol"`, `"skola"`.

Do triedy *Midtermarka* pridajte metódu *spolocnyPrefix*. Táto metóda dostane ako parametre referencie dva reťazce a vráti **najdlhší** taký reťazec (referenciu na taký reťazec), ktorý je prefixom oboch zadaných reťazcov.

Príklady:

- najdlhší spoločný prefix reťazcov `"promocia"` a `"program"` je reťazec `"pro"`
- najdlhší spoločný prefix reťazcov `"kolovratok"` a `"kolo"` je reťazec `"kolo"`
- najdlhší spoločný prefix reťazcov `"program"` a `"kolo"` je reťazec `""`

```
public String spolocnyPrefix(String r1, String r2)
```

Časť 2

Korytnačí salaš (10 bodov)

- *(3 body)* Vytvorte triedu *MidtermPane*, ktorá rozširuje triedu *WinPane*. Po vytvorení kresliacej plochy triedy *MidtermPane* nech sa v nej vytvorí 8 korytnačiek triedy *Midtermarka* (alebo triedy *Turtle*) na náhodných pozíciach.

- *(7 bodov)* Do triedy *MidtermPane* pridajte metódu *vsetkyVKosiari*, ktorá vráti, **či všetky** korytnačky v kresliacej ploche sa nachádzajú vo vnútri obdĺžnika rovnobežného s kresliacou plochou a ktorého ľavý horný roh má súradnice `(x, y)`, jeho šírka je `s` a výška `v`.

```
public boolean vsetkyVKosiari(double x, double y, double s, double v)
```

Na druhej strane nájdete oficiálny ťahák.

Oficiálny ťahák

Polsemestrálny test

Ústav informatiky
Prírodovedecká fakulta
UPJŠ v Košiciach

Dvakrát meraj (rozmýšľaj), raz rež (programuj)

Základné metódy objektov triedy String:

`int length()`
- vráti dĺžku reťazca

`char charAt(int index)`
- vráti znak na zadanom indexe v reťazci (znaky sú indexované od 0)

`boolean equals(String r)`
- vráti *true* práve vtedy, keď tento reťazec sa skladá z tej istej postupnosti znakov ako reťazec referencovaný parametrom *r*

`String trim()`
- vráti referenciu na novovytvorený reťazec vytvorený odstránením počiatočných a koncových medzier

`String toLowerCase()` resp. `String toUpperCase()`
- vráti referenciu na novovytvorený reťazec po zmene znakov v reťazci na malé (veľké) písmena

`String substring(int zacIndex, int konIndex)`
- vráti referenciu na novovytvorený reťazec obsahujúci podreťazec tvorený znakmi na indexoch *zacIndex* (vrátane) až *konIndex* (nie je zahrnutý)

`int indexOf(String podretazec)` resp. `int indexOf(char znak)`
- vráti index prvého výskytu podreťazca resp. znaku v reťazci. Ak sa v reťazci nenachádza vráti -1

Základné metódy objektov triedy Turtle:

`void center()`
- presunie korytnačku do stredu plochy, v ktorej sa nachádza (korytačka musí byť v ploche)

`void setPosition(double x, double y)`
- presunie korytnačku na pozíciu so súradnicami [x, y], čiara sa nekreslí

`void step(double dlzka)`
- spraví krok v smere natočenia zadanej dĺžky, čiara sa kreslí v závislosti od stavu kresliaceho pera

`void turn(double uhol)`
- otočí korytnačku o zadaný uhol v smere hodinových ručičiek

`void moveTo(double x, double y)`
- korytačka spraví krok do bodu na súradniciach [x, y], čiara v závislosti od kresliaceho pera

`void setDirection(double smer)`
- natočí korytnačku zadaným smerom (smer 0 je nahor, 90 doprava, atď.)

`double getDirection()`
- vráti smer aktuálneho natočenia korytnačky

`double getDirectionTowards(double x, double y)`
- vráti, aký by mala mať korytnačka sme (natočenie), aby bola smerom k bodu na súr. [x, y]

`double distanceTo(double x, double y)`
- vráti vzdialenosť korytnačky k bodu na súradniciach [x, y]

`void dot(double polomer)`
- nakreslí vyplnený kruh (farbou výplne) so zadaným polomerom a stredom v pozícii korytnačky

`void setFillColor(Color farba)`
- nastaví farbu výplne

`void setPenColor(Color farba)`
- nastaví farbu kresliaceho pera

`void penDown()`
- zapne kresliace pero

`void penUp()`
- vypne kresliace pero

Základné metódy objektov triedy WinPane (kresliaca plocha):

`void add(Turtle korytnacka)`
- pridá (referencovanú) korytnačku do kresliacej plochy

`void remove(Turtle korytnacka)`
- odoberie (referencovanú) korytnačku z kresliacej plochy

Java a polia

- prechod všetkými indexami poľa referencovaného z premennej *pole*:

```
for (int i=0; i<pole.length; i++) { … }
```

JPAZ a myšacie udalosti

```
protected void onMouseClicked(int x, int y, MouseEvent detail) {
    if ((detail.getButton() == MouseEvent.BUTTON1) &&
            detail.isControlDown()) {
        // pri zatlačení ľavého tlačidla myši
        // vo chvíli, keď je zatlačený aj Ctrl
    }
}
```

Farby

`Color.red, Color.blue, Color.green, Color.gray, Color.black` … alebo `new Color(int r, int g, int b)`, kde *r*, *g* a *b* sú celé čísla od 0 po 255.

Náhodné číslo

Vygenerovanie náhodného čísla z intervalu <0, a): `Math.random()*a`
Vygenerovanie náhodného celého čísla od 0 po n: `(int)(Math.random()*(n+1))`

Vytvorenie poľa

Vytvorenie poľa 6 celých čísel:

```
int[] pole = new int[6];
```

Vytvorenie poľa 6 celých čísel s inicializáciou hodnôt:

```
int[] pole = {3, 4, 6, 1, 2, 4};
```

Výpis poľa:

```
System.out.println(Arrays.toString(pole));
```