Polsemestrálny test
Zadanie

Ústav informatiky
Prírodovedecká fakulta
UPJŠ v Košiciach

Dvakrát meraj (rozmýšľaj), raz rež (programuj)

Pravidlá a informácie:

- čas na riešenie úloh je **80 minút**, resp. do 14:05,
- 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), monitorovací softvér musí byť spustený počas celého testu,
- porušenie pravidiel má za následok hodnotenie FX,
- svoje riešenia odovzdávajte cez systém Moodle (http://moodle.ics.upjs.sk/).

Upozornenie:

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

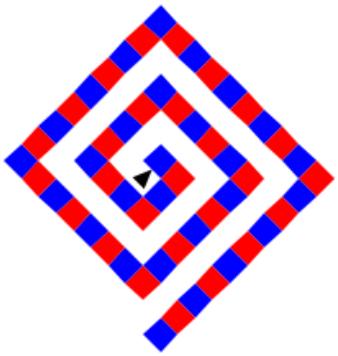
Ktoré úlohy treba riešiť:

V ***Časti 1*** je cieľom úloh 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 (ak to nie je zrejmé z odoslaného).

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

Časť 1 (dve úlohy z troch)

Špirálový múr (10 bodov)



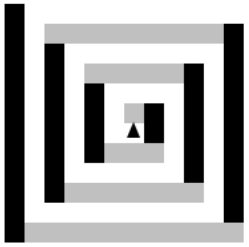
Do triedy `Midtermarka` pridajte metódu `mur`, ktorá nakreslí špirálový múr (viď obrázok s 10-timi stenami). Metóda má dva parametre `strana` a `pocetStien`. Prvý parameter `strana` určuje stranu štvorcov, z ktorých je múr vytvorený. Druhý parameter `pocetStien` určuje celkový počet stien múra. Múr je tvorený postupne sa striedajúcimi vyplnenými štvorcami červenej a modrej farby. Pomyselný stred a natočenie múra sú určené začiatočnou (a koncovou) pozíciou korytnačky a jej natočením. Po vykonaní metódy nech je korytnačka na pozícii a v smere, ako bola pred vykonaním metódy.

```
public void mur(double strana, int pocetStien)
```

Rady:

- Odporúčame si vytvoriť metódu, ktorá nakreslí vyplnený štvorec so stredom v aktuálnej pozícii korytnačky a so zadanou dĺžkou strany a farbou výplne (2 body).

```
public void vyplnenyStvorec(double strana, Color farba)
```



- Všimnite si, že celý múr možno rozložiť na steny (viď ilustračný obrázok so znázornením stien). Prvá stena sa skladá z jedného štvorca, druhá z dvoch, tretia z troch, atď. Za zjednodušenú verziu úlohy, v ktorej sa striedajú len farby stien (tak ako na ilustračnom obrázku) a ktorá nevyužíva metódu `setPenWidth`, môžete celkovo získať maximálne 7 bodov.

Pytagorejské číslo (10 = 8+2 bodov)

Do triedy `Midtermarka` pridajte metódu `pytagorejskeCislo`, ktorá pre zadané prirodzené číslo c ($c \geq 1$) vráti, či je toto číslo pytagorejské. Číslo $c \in \mathrm{N}$ nazveme pytagorejským, ak existujú také prirodzené čísla $a, b \in \mathrm{N}, a, b \geq 1$, že $c^2 = a^2 + b^2$.

```
public boolean pytagorejskeCislo(int c)
```

Hodnotenie: 2 body sa udeľujú podľa efektívnosti riešenia.
Rada: Číslo c je pytagorejským práve vtedy, keď existuje také $a \in \{1, 2, \ldots, c-1\}$, že číslo $c^2 - a^2$ je druhou mocninou nejakého prirodzeného čísla.

Zámena susedov (10 bodov)

Do triedy `Midtermarka` pridajte metódu `zamenSusedne`. Táto metóda dostane ako parameter ne`null`ovú referenciu na reťazec (objekt triedy `String`) a vráti referenciu na novovytvorený reťazec, ktorý vznikne zo zadaného reťazca zámenou susedných znakov po dvojiciach. Konkrétne vymenia sa znaky na indexoch 0 a 1, potom znaky na indexoch 2 a 3, 4 a 5, atď. Ak je reťazec nepárnej dĺžky, posledný znak ostáva na svojom mieste. Príklad:

```
zamenSusedne("Programovanie") = "rPgoaromavine"
```

```
public String zamenSusedne(String r)
```

Časť 2

Vzdialenostný histogram (10 bodov)

- *(2 body)* Vytvorte triedu `MidtermPane`, ktorá rozširuje triedu `WinPane`. Po vytvorení kresliacej plochy triedy `MidtermPane` nech sa v nej (automaticky v konštruktore, resp. „inicializačnej metóde“) vytvorí 12 korytnačiek triedy `Turtle` na náhodných pozíciách vo viditeľnej časti kresliacej plochy.

- *(8 bodov)* Nech d je nenulové kladné číslo ($d > 0$) a C je bod na súradniciach $[x, y]$. Na základe vzdialenosti od bodu C môžeme korytnačky rozdeliť do „zón“. Povieme, že korytnačka sa nachádza v i-tej zóne okolo bodu C, keď jej vzdialenosť od bodu C je v intervale $\langle i \cdot d, \mathrm{i} \cdot \mathrm{d} + \mathrm{d})$. Do triedy `MidtermPane` pridajte metódu `histogram`, ktorá vráti referenciu na pole, v ktorom na indexe i bude uložený počet korytnačiek nachádzajúcich sa v i-tej zóne okolo zadaného bodu C. Dĺžka vráteného poľa nech je najmenšia možná s tou vlastnosťou, že súčet prvkov poľa bude rovný počtu korytnačiek v ploche. Metóda `histogram` má 3 parametre: `x`, `y` - súradnice bodu C a šírku zón `d`.

  ```
  public int[] histogram(double x, double y, double d)
  ```

  Hodnotenie: 4 body za vrátenie poľa správnej dĺžky, 4 body za vrátenie správne naplneného poľa.

  Rada: Odporúčame si najprv vypočítať maximálnu vzdialenosť medzi bodom C a ľubovoľnou korytnačkou v ploche. Na základe tohto vypočítate, aké veľké pole potrebujete vrátiť.

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

`void turnTowards(double x, double y)`
- natočí korytnačku tak, aby bola natočená smerom k bodu na súradniciach [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()` resp. `void penUp()`
- zapne resp. vypne kresliace pero

`void setPenWidth(double hrubka)`
- nastaví hrúbku kresliaceho pera

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

`int getWidth()` resp. `int getHeight()`
- vráti šírku, resp. výšku 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));`
Kopírovanie prvkov poľa:
```
System.arraycopy(odkiaľ, odAkéhoIndexu, kam, odAkéhoIndexu, koľkoPolíčok);
```

Čísla

`Double.MAX_VALUE` - najväčšie číslo, ktoré možno uložiť v premennej typu double
`Double.POSSITIVE_INFINITY` – $+\infty$
`double cislo = Double.parseDouble("3.14");` - prevedie reťazec na číslo
`Math.sqrt(c)` – vyráta odmocninu zadaného čísla `c`