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**, resp. do 11:20,
- 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,
- svoje riešenia odovzdávajte cez systém Moodle (http://moodle.ics.upjs.sk/).

Upozornenie:

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

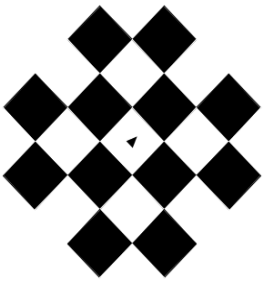
Ktoré úlohy treba riešiť:

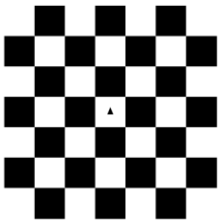
V ***Časti 1*** je cieľom úloh vytvoriť triedu `Midtermarka`, ktorá rozširuje triedu `Turtle`. Z prvej trojice úloh si **vyberte len 2 úlohy**, ktoré **budete riešiť!!!** To, ktoré úlohy ste sa rozhodli riešiť, uveďte v komentári pri odosielaní riešenia cez Moodle (ak to nie je zrejmé z odoslaného).

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

Časť 1 (dve úlohy z troch)

Šachovnica (10 bodov)





Do triedy `Midtermarka` pridajte metódu `sachovnica`, ktorá nakreslí šachovnicu ako na obrázku. Metóda má dva parametre `pocetPolicok` a `sirkaPolicka`. Prvý parameter `pocetPolicok` určuje „šírku“ a „výšku“ šachovnice (počet políčok v jednom rade a jednom stĺpci šachovnice). Druhý parameter `sirkaPolicka` určuje šírku a výšku jedného políčka. Všimnite si, že šachovnica pozostáva z vyplnených štvorcov bielej a čiernej farby. Korytnačka sa na začiatku nachádza v strede kreslenej šachovnice a je natočená v smere jedného zo stĺpcov. Po nakreslení šachovnice nech je korytnačka na pozícii a je natočená v smere, ako bola pri volaní metódy.

```
public void sachovnica(int pocetPolicok, double sirkaPolicka)
```

Rady:

- Odporúčame vytvoriť si pomocnú metódu na nakreslenie vyplneného štvorca zadanej farby:

```
public void policko(double sirka, Color farba)
```

- Nech `r` je poradové číslo riadka, v ktorom sa nachádza políčko, a `s` je poradové číslo stĺpca, v ktorom sa nachádza políčko. Všimnite si, že pre všetky políčka rovnakej farby na šachovnici platí, že výraz `r+s` má rovnakú paritu (t.j., buď sú všetky `r+s` políčok rovnakej farby párne, alebo nepárne).

Vyvážené číslo (10 bodov)

Kladné nenulové číslo nazveme vyváženým, ak súčet jeho párnych a nepárnych cifier je rovnaký. Napríklad číslo 1542 je vyvážené, pretože 1+5 = 6 = 4+2. Podobne je vyváženým aj číslo 94412, pretože 9+1 = 10 = 4+4+2. Naopak číslo 24 nie je vyvážené, pretože 2+4 = 6, no súčet nepárnych cifier je nula. Do triedy `Midtermarka` pridajte metódu `vyvazeneCislo`, ktorá vráti, či zadané číslo je vyváženým číslom.

```
public boolean vyvazeneCislo(int cislo)
```

Počet rozdielov (10 bodov)

Do triedy `Midtermarka` pridajte metódu `pocetRozdielov`. Táto metóda dostane ako parametre referencie na dva reťazce a vráti na koľkých pozíciách v reťazcoch sa ich znaky nerovnajú. Napríklad reťazce `"sova"` a `"láva"` sú rozdielne v dvoch pozíciách (prvých dvoch). Reťazce `"sova"` a `"sopečný"` sa nerovnajú v piatich pozíciách, reťazce `"škola"` a `"okolnosti"` sa nerovnajú v šiestich pozíciách, reťazce `"keltský"` a `"test"` sa nerovnajú v piatich pozíciách. Pri vyhodnocovaní, či sú písmená na zadanej pozícii rovnaké, na veľkosti písmen záleží.

```
public int pocetRozdielov(String r1, String r2)
```

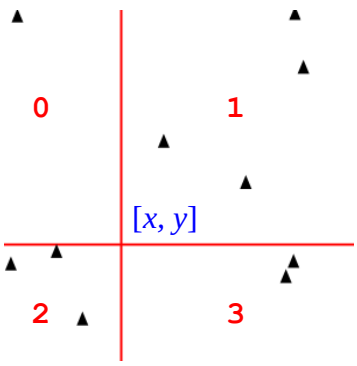
Časť 2

Korytnačky v kvadrantoch (10 bodov)

- *(3 body)* Vytvorte triedu `MidtermPane`, ktorá rozširuje triedu `WinPane`. Po vytvorení kresliacej plochy triedy `MidtermPane` nech sa v nej vytvorí 10 korytnačiek triedy `Turtle` na náhodných pozíciách vo viditeľnej časti kresliacej plochy.

- *(7 bodov)* Do triedy `MidtermPane` pridajte metódu `vKvadrantoch`, ktorá vráti, koľko korytnačiek sa nachádza v každom zo štyroch kvadrantov kresliacej plochy. Jednotlivé kvadranty sú určené navzájom kolmými osami (rovnobežnými so stranami kresliacej plochy), ktorých priesečník sa nachádza na zadaných súradniciach (parametre `x` a `y`) – tak ako je to znázornené na obrázku. Očíslovanie jednotlivých kvadrantov je vyznačené na obrázku**.** Výsledkom metódy nech je referencia na jednorozmerné pole, v ktorom na indexe **i** sa bude nachádzať počet korytnačiek v **i**-tom kvadrante. Ak korytnačka leží na niektorej z osí, nezarátavame ju do žiadneho z kvadrantov.
  V prípade na obrázku vpravo bude počet korytnačiek v kvadrantoch takýto: `[1, 4, 3, 2]`.

```
public int[] vKvadrantoch(double x, double y)
```



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()`
- 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));
```