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 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 a predmetového webu,
- nie je dovolené používať žiadnu inú aplikáciu než Eclipse (s výnimkou webového prehliadača pri odosielaní riešenia),
- porušenie pravidiel má za následok hodnotenie FX,
- svoje riešenia odovzdávajte cez systém Moodle (http://lms.ics.upjs.sk/).

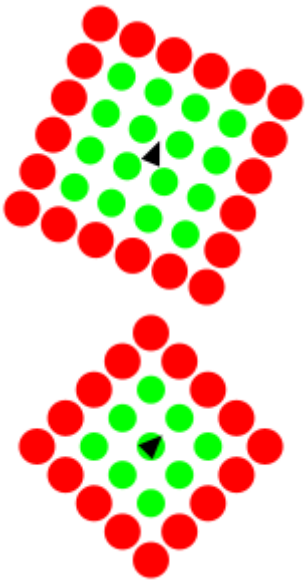
Ktoré úlohy treba riešiť:

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)

Jesenný sad (5 bodov)



Do triedy `Midtermarka` pridajte metódu `jesennySad`, ktorá nakreslí sad ako na obrázku vpravo. Metóda má jeden parametre `pocetNaStranu`. Parameter `pocetNaStranu` určuje počet stromov na strane štvorcového sadu (na obrázku vpravo má horný sad 6 stromov na jednej strane, dolný sad 5 stromy). Sad je tvorený kruhmi pričom kruhy vo vnútri sadu majú polomer 7 a zelenú farbu a kruhy po obvode majú červenú farbu a polomer 9, vzdialenosti stredov kruhov sú 20. Korytnačka sa na začiatku nachádza v strede sadu a natočená je v smere strany sadu. Po vykonaní metódy nech je korytnačka na pozícii a v smere, ako bola pred vykonaním metódy.

```
public void jesennySad(int pocetNaStranu)
```

Rada:

. Odporúčame si vytvoriť dvojicu pomocných metód. Jedna pomocná metóda nakreslí jeden riadok z vnútra sadu skladajúci sa z `pocetNaStranu` kruhov, pričom krajné majú červenú faru. Druhá pomocná metóda nakreslí horný alebo dolný riadok zo sadu skladajúci sa z `pocetNaStranu` červených kruhov.

```
public void vnutornyRiadok(int pocetNaStranu)
public void vonkajsiRiadok(int pocetNaStranu)
```

k-narcistické číslo (5 bodov)

Kladné nenulové číslo n nazveme k-narcistickým číslom, ak súčet jeho k-tých mocnín cifier je rovný číslu n. Napríklad číslo 153 je 3-narcistické, pretože $1^3+5^3+3^3 = 1+125+27 = 153$. Podobne je 3-narcistickým aj 407, pretože $4^3+0^3+7^3 = 407$ a 4-narcistickým je číslo 1634, pretože $1^4+6^4+3^4+4^4 = 1634$. Naopak číslo 24 nie je 3 narcistickým, pretože $2^3+4^3 = 72$ a číslo 153 nie je 2 narcistickým, pretože $1^2+5^2+3^2 = 35$. Do triedy Midtermarka pridajte metódu kNarcissticNumber, ktorá pre parametre n a k vráti, či n je k narcistické číslo. Pozn.: Hodnoty n a k sú zvolené tak aby nedošlo k prekročeniu rozsahu int-ov.

```
public boolean kNarcissticNumber(int n, int k)
```

Orež na palindróm (5 bodov)

Palindromickým dvojčaťom znaku z1 v reťazci nazveme znak z2, ktorý je rovnaký ako znak z1 a nachádza sa na rovnakej pozícii od konca ako sa znak z1 nachádza od začiatku. Pozn. znak môže byť svojim vlastným palindromickým dvojčaťom.

Do triedy Midtermarka pridajte metódu orezNaPalindrom. Táto metóda dostane ako parameter ne**null**ovú referenciu na reťazec (objekt triedy String) a vráti referenciu na novovytvorený reťazec (objekt triedy String), ktorý vznikne zo zadaného reťazca odstránením všetkých dvojíc znakov, ktoré nemajú palindromické dvojča.

Príklady, odstránené znaky sú zobrazené hrubým písmom:

```
orezNaPalindrom("ABXCA") = "AXA"
orezNaPalindrom("FRHJRF") = "FF"
orezNaPalindrom("ABBA") = "ABBA"
orezNaPalindrom("ADAM") = ""
orezNaPalindrom("CZDYDXC") = "CDYDC"
orezNaPalindrom("ANICKA") = "AA"
```

```
public String orezNaPalindrom(String r)
```

Rada:

Odporúčame si vytvárať reťazec zo začiatočnou časťou slova a koncovou časťou slova, ktoré spojíme až na konci. Napr. pre reťazec `CZDYDXC` máme začiatočnú časť vytváraného slova `CD` a koncovú časť `DC`, potom ich spojíme aj spolu so stredom na `CDYDC`, podobne pre `FRHJRF` máme časti `FR` a `RF`, ktorých spojením vznikne `FRRF`.

Časť 2

Korytnačie tímy (5 bodov)

- *(1 bod)* 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, pričom prvá polovica korytnačiek bude umiestená v hornej polovici kresliacej plochy a druhá polovica v dolnej časti. Vytvorené korytnačky majú náhodné natočenia a nachádzajú sa vo viditeľnej časti kresliacej plochy.

- *(4 body)* Korytnačky sa rozdeľujú do dvoch tímov. Oba tímy majú svojho kapitána (indexy kapitánov sú zadané parametrami metódy). Každá zo zvyšných korytnačiek je priradená do toho tímu, ku ktorému kapitánovi je najbližšie. Príslušnosť k svojmu tímu preukáže natočením ku kapitánovi a vykreslením čiary smerom k nemu (korytnačka po vykreslení čiary ostáva na svojom pôvodnom mieste, mení sa len jej natočenie).

  Metóda `rozdelenieDoTimov` nakresli priradenie korytnačiek, otočí každú korytnačku ku svojmu kapitánovi a vráti hodnotu `true` ak obidva tímy majú rovnaký počet korytnačiek. Pri nerovnovážnom rozdelení tímov vráti hodnotu `false`.

  ```
  public boolean rozdelenieDoTimov(int idxA, int idxB)
  ```

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 directionTowards(double x, double y)`
- vráti smer, pri ktorom by bola korytnačka 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 openPolygon()`
- zapne sledovanie ohraničovania oblasti ktorá bude vyplnená pri `void closePolygon()`

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.POSITIVE_INFINITY` – +∞
`double cislo = Double.parseDouble("3.14");` - prevedie reťazec na číslo
`Math.sqrt(c)` – vyráta odmocninu zadaného čísla `c`

Znaky

`Character.isUpperCase(z)` – vráti, či znak `z` predstavuje veľké písmeno
`Character.toUpperCase(z)` – vráti znak, ktorý vznikne zo `z` zmenou na veľké písmeno