# Polsemestrálny test
## Zadanie

Ústav informatiky
Prírodovedecká fakulta
UPJŠ v Košiciach

**Dvakrát meraj (rozmýšľaj), raz rež (programuj)**

**Pravidlá a informácie:**

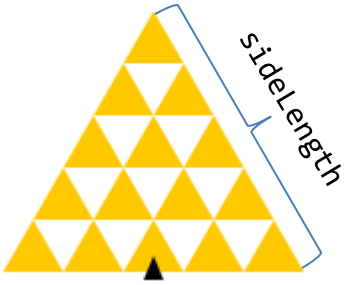
- čas na riešenie úloh je **80 minút**, resp. do 9:30,
- nie je dovolená žiadna (elektronická ani neelektronická) komunikácia s kýmkoľvek okrem dozoru,
- nie je dovolené používať žiadne zdroje ani materiály okrem oficiálneho ťaháka a predmetového webu vrátane gitlabu,
- nie je dovolené používať žiadnu inú aplikáciu než IntelliJ (s výnimkou webového prehliadača pri odosielaní riešenia),
- porušenie pravidiel má za následok hodnotenie FX,
- svoje riešenia odovzdávajte cez systém Moodle (https://moodle.science.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)




### Mon (10 bodov)

Japonský klan Hōjō bol klan samurajov v 13. a 14. storočí, ktorý mal svoj „mon“, teda symbol, ktorým bol ich klan identifikovateľný. Tento symbol sa nazýva „mitsudomoe“. Vašou úlohou bude vykresliť tento symbol v rôznych veľkostiach.

Do triedy `Midtermarka` pridajte metódu `mon`, ktorá nakreslí rozšírenú verziu „mon“, ako na obrázkoch pvravo. Metóda má dva parametre `sideLength` a `levels` (dĺžka strany a poschodia). Parameter `sideLength` určuje dĺžku každej zo strán `mon` a parameter `levels` určuje počet poschodí `mon` (na obrázkoch sú korytnačkou nakreslené `mon` s 5 a 4 poschodiami). `mon` je tvorený vyplnenými rovnostrannými trojuholníkmi s dĺžkou strany `sideLength /levels`, pričom sa používa žltá farba výplne (yellow). Korytnačka sa na začiatku nachádza v strede jednej zo strán `mon` a natočená je v smere výšky `mon`. Po vykonaní metódy nech je korytnačka na pozícii a v smere, ako bola pred vykonaním metódy.

```
public void mon (double sideLength, int levels)
```

Rada:

Odporúčame si vytvoriť pomocnú metódu, ktorá nakreslí vyplnený trojuholník so zadanou dĺžkou strany a s počiatočnou a koncovou pozíciou ako je naznačené na obrázku.

```
public void filledTriangle(double size, int levels)
```

Taktiež odporúčame si vytvoriť metódu, ktorá nakreslí jedno poschodie `mon` skladajúce sa z `n` trojuholníkov so zadanou dĺžkou strany, pričom odporúčaná začiatočná a koncová pozícia sú naznačené na obrázku.

```
public void oneLevel(double size, int n)
```

**Na druhej strane papiera nájdete oficiálny ťahák.**

### Goldbach (10 bodov)

Goldbachova hypotéza tvrdí, že každé párne celé číslo väčšie ako 2 sa dá vyjadriť ako súčet dvoch prvočísel (je overené že platí pre všetky párne čísla menšie ako 4*10^18, teda to pokryje všetky `int`-y). Takýchto súčtov pre jedno číslo môže existovať viacero. Do triedy `Midtermarka` pridajte metódu `goldbach`. Táto metóda dostane ako parameter kladné párne celé číslo `n > 2` a nájde taký pár `p` a `q`, že `p + q = n`, obe sú prvočísla a väčšie z nich je maximálne možné. Metóda vráti väčšie z nájdených čísel `p` a `q`. Pozn.: Neočakávame efektívne riešenie. Odporúčame si vytvoriť pomocnú metódu `isPrime`.

```
goldbach(16) = 13      lebo 16 = 3 + 13 = 5 + 11 a max(3, 13) > max(5, 11)
goldbach(22) = 19      lebo 22 = 3 + 19 = 5 + 17 = 11 + 11 a
                       max(3, 19) > max(5, 17) > max(11,11)
goldbach(12) = 7       lebo 12 = 5 + 7
```

```
public int goldbach(int n)
```

### Faktor (10 bodov)

Do triedy `Midtermarka` pridajte metódu `subsequence`. Táto metóda dostane ako parameter ne`null`ové referencie na dva reťazce `s1` a `s2` (objekty triedy `String`). Metóda vráti či sa reťazec `s1` nachádza v reťazci `s2`, pričom znaky `s1` sa nemusia nachádzať v reťazci `s2` bezprostredne za sebou, len sa musí zachovať ich poradie.

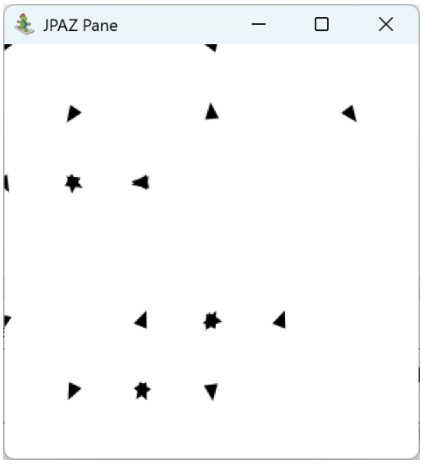
Príklady:

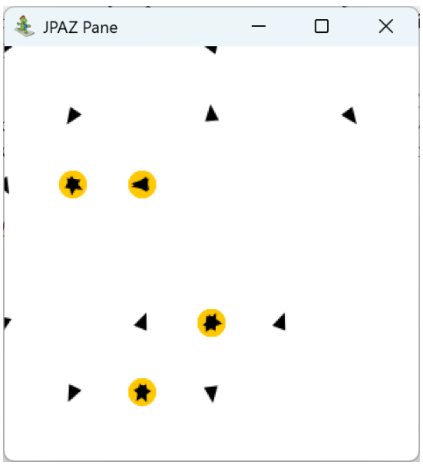
```
subsequence("PZa", "PAZ1a") = true,
subsequence("UPJŠ", "Univerzita P.J. Šafárika") = true,
subsequence("ellooW", "Hello World") = false,
subsequence("aaaa", "aaa") = false,
subsequence("aaa", "aaaa") = true,
subsequence("aHoj3", "aHoj3") = true.
```

```
public boolean subsequence(String s1, String s2)
```

## Časť 2

### Korytnačky v ploche (10 bodov)




- (3 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 20 korytnačiek triedy `Turtle`. Korytnačky sú umiestnené na náhodných pozíciách vo viditeľnej časti kresliacej plochy, pričom ich x aj y súradnica je násobkom čísla 50. Korytnačky sú náhodne natočené.

- (7 bodov) Do triedy `MidtermPane` pridajte metódu `overlap`. Táto metóda zistí, či sa (aspoň) dve korytnačky nachádzajú na rovnakých súradniciach. Ak áno, vykreslí sa okolo nich kruh s polomerom 10.

```
public void overlap()
```

# 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