1. Domácí úloha 07

Základní informace:

- **Účel:** rozšíření existujících tříd, dědění implementace
- **Kostra:** `07_RozsireniADedicnost.zip`
- **Odevzdávaný soubor aplikace:** `07_RozsireniADedicnost.jar`
- **Odevzdávané soubory UML zabalené do JAR:** `07_uml.jar`

Zadání:

- rozšiřte stávající funkčnosti při zachování původních u tříd `Rande` a `Par`
- připravte třídu `Superman`
- do Portálu odevzdáte JAR soubor celého projektu
- rozšiřte UML diagram tříd

Postup řešení:

- stáhněte si soubor `07_RozsireniADedicnost.zip`, rozbalte jej - NEotvírejte projekt v BlueJ
- do rozbaleného adresáře nakopírujte soubory `Osoba.java`, `Rozmer.java`, `Pohlavi.java`, `IMeritelny.java`, `IZvyrazneny.java`, `Zvyraznovac.java`, `Rande` a `Par`, které jste odevzdávali v minulém DU
  - třídu `Hlavni` nebudeme potřebovat
- v BlueJ otevřete projekt `07_RozsireniADedicnost`
- doplňte a upravte třídu `Rande`
  - přidejte konstruktor se signaturou a kontraktem:

```
/**********************************************************
 * převezme instance muže a ženy a aktualizuje domácí pozice
 * zajistí zobrazení muže a ženy
 *
 * @param muz existující muž chystající se na rande
 * @param zena existující žena chystající se na rande
 */
public Rande(Osoba muz, Osoba zena) {
```

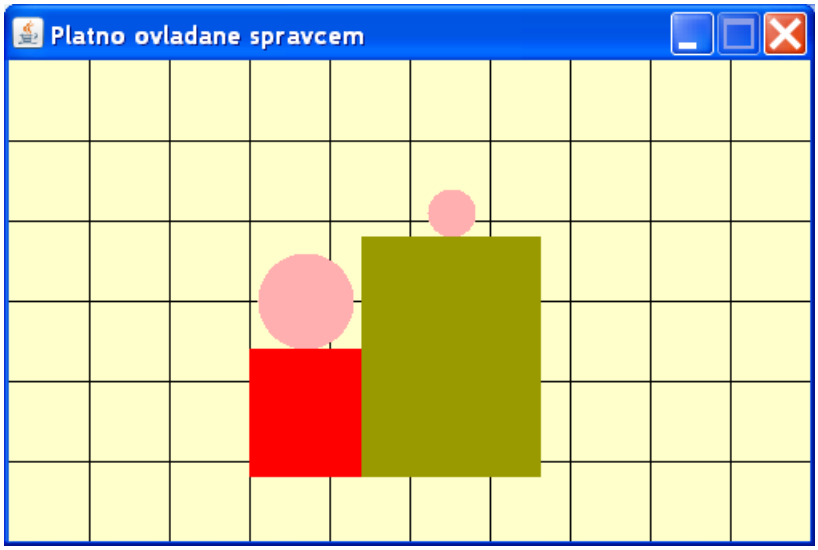
  - přidejte metodu:

```
/**
 * Y posun muže, pokud je žena vyšší, jinak 0
 * @return posun v pixelech
```

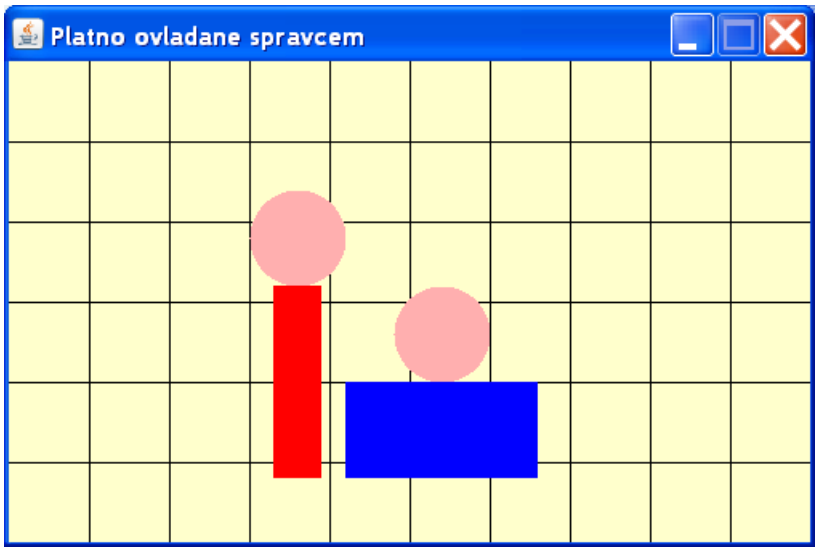
```
  */
 public int yPosunMuze() {
```

a podobnou metodu `yPosunZeny()`

- s využitím metod `yPosunMuze()` a `yPosunZeny()` upravte metodu `jdouNaRande()` tak, aby různě vysoké osoby měly při schůzce tělo stejně vysoko
  - dopředu vypočtěte finální souřadnice ženy i muže tak, aby na rande šli rovnou na své správné místo, tzn. nepřesouvali se poté, co se setkají na úrovni vrcholků svých hlav a zjistí, že jsou různě vysocí
  - pomocí testu z minulého DU *testSetkaniUprostred()* ověřte, že funkčnost z minulého DU zůstala zachována
  - správnou funkci rozšíření ověřte pomocí *testSetkaniMalaGeneral()*
    - **Poznámka**: Pro generála je občas plátno trochu malé, což nijak neřešte.
    - **Nápověda**: Pokud se vám při ladění budou zdát rychlosti přesunů v testech malé, můžete si je v testovací třídě zvýšit.

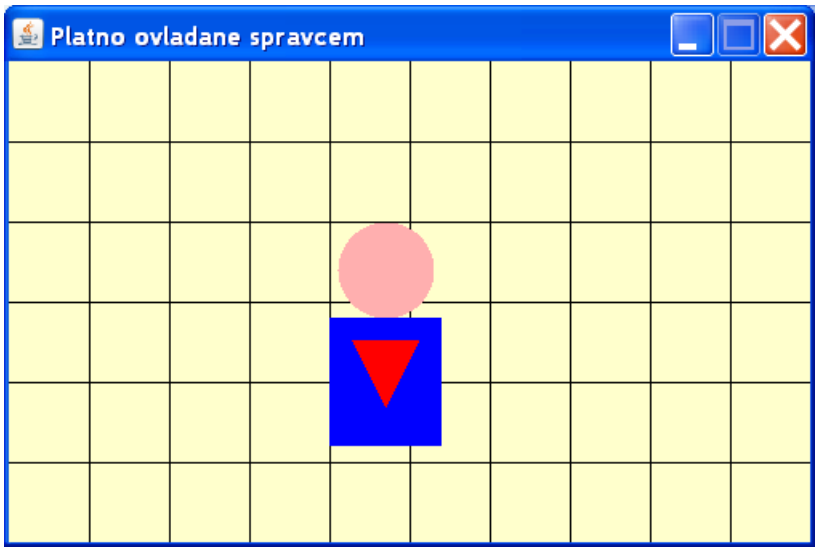


  - správnou funkci rozšíření ověřte pomocí *testSetkaniVysokaTlusty()*



  - zde je vidět systémový problém, kdy žena má štíhlejší tělo než je šířka její hlavy, takže pár nemůže být těsně vedle sebe
  - tento problém nemá dobré řešení
    - pokud by se sešel pár dvou štíhlých lidí, nesměly by se jim překrývat hlavy
    - budeme nadále předpokládat, že u běžných osob je tělo širší než hlava

- doplňte a upravte třídu `Par`
  - doplňte instanční konstanty `int yPosunMuze` a `int yPosunZeny`
  - zmíněné konstanty nastavte v konstruktoru
  - upravte metody `getPozice()` a `setPozice()` tak, aby využívaly zmíněné konstanty
  - správnou funkci rozšíření ověřte pomocí testů z DU-06 *testJdouSpolu(), testPokracujiSpolu(), testCeleRande()*
    - tyto testy musejí dávat stejné výsledky jako v DU-06, tzn. naše úpravy nezhoršily původní stav;
- upravte třídu `Rande`
  - spusťte *testCeleRandeGeneral()* a věnujte pozornost závěrečné domácí pozici ženy
  - ve třídě `Rande` vyřešte problém úpravou metody `jdouDomu()`
  - správnou funkci opravy ověřte pomocí testů *testCeleRandeGeneral()* a *testCeleRande()* z DU-06
- připravte třídu `Superman` která bude dědit implementaci od třídy `Osoba` - Superman se vyznačuje modrým oděvem a znakem na prsou, který v našem případě bude pro zjednodušení představován jen červeným trojúhelníkem



  - ve třídě `Osoba` změňte přístupové právo `private` na `protected` u atributů (umožní to jejich přímé využití ve třídě potomka):
    - `IMPL_VELIKOST_HLAVY`
    - `telo`

- další pokyny se vztahují jen k vytvářené třídě `Superman`
- definujte statickou třídní konstantu `POMER_ZNAK_TELO = 3.0/5.0`, která udává poměr šířek znaku a těla
- definujte instanční konstanty, které budou nastaveny v konstruktoru
  - `Trojuhelnik znak`
  - `int posunZnaku` - posun znaku vůči tělu - bude stejný v X i Y souřadnici
- vytvořte konstruktor se signaturou a kontraktem

```
  /*************************************************************
   * vytvoří instanci libovolného rozměru na zadané pozici
   * tělo je vždy modré, standardních rozměrů Osoby
   * znak je vždy červený orientovaný vrcholem dolů
   * znak je v poměru k tělu a je odsazen o {@code posunZnaku}
   * vůči X i Y souřadnici těla
   *
   * @param pozice pozice zobrazení
   * @param velikostHlavy velikost hlavy, podle které se vytvoří poměrově ▶
osoba
   */
  public Superman(Pozice pozice, int velikostHlavy) {
```

  - kostruktor nejprve pomocí `super()` vyvolá vhodný konstruktor `Osoba()`
  - pak ze šířky těla vypočte šířku znaku, která bude současně i výškou znaku
  - konstanta `posunZnaku` se vypočte tak, že znak musí být vycentrován vůči tělu - stejnou hodnotu pak použijte i pro Y-posun znaku
  - vytvořte instanci znaku pomocí konstruktoru

```
public Trojuhelnik( Pozice pozice, Rozmer rozmer, Barva barva, Smer8 smer )
```

- vytvořte metodu `public Trojuhelnik getZnak()`, která bude použita pro testovací účely
- správnou funkci implementace ověřte pomocí *testKompletniKonstruktor()*, který před prvním použitím odkomentujte
  - všimněte si, že třída `Superman` zdědila ze třídy `Osoba` implementaci rozhraní `IKresleny`, takže lze osobu vykreslit bez implementace metody `nakresli()`
  - bohužel však chybí vykreslení znaku, proto je nutné překrýt metodu

```
  @Override
  public void nakresli(Kreslitko kreslitko) {
```

  - v metodě nejprve pomocí `super.nakresli(kreslitko)` vyvolejte tutéž metodu z rodičovské třídy a poté přidejte nakreslení znaku

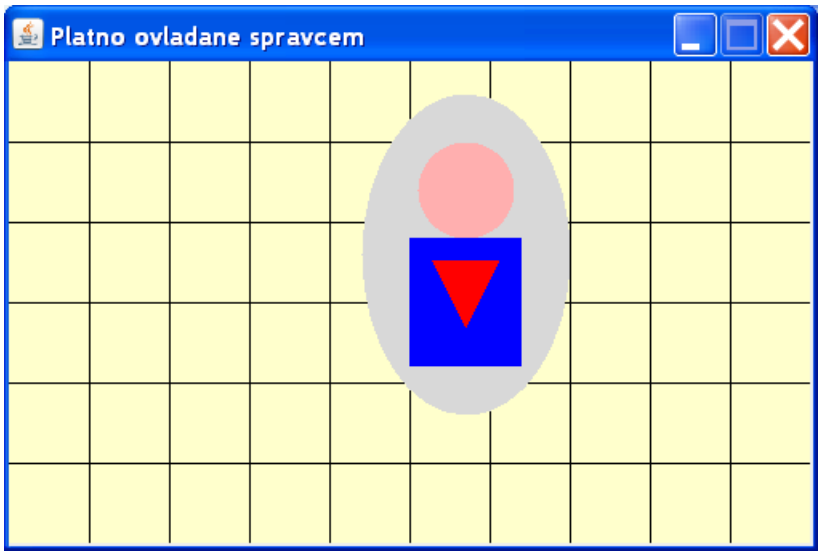
- protože metoda `zobraz()` je děděná ze třídy `Osoba`, není nutné ji psát (překrývat) - třída `Superman` je speciální případ třídy `Osoba` a z pohledu správce plátna se vykreslují stejně

- správnou funkci implementace ověřte pomocí testů *testKompletniKonstruktor()* a *testVelkySuperman()*, které před prvním použitím odkomentujte
- vytvořte konstruktor, který bude pomocí `this()` využívat funkčnost předchozího konstruktoru

```
/*******************************************************
 * vytvoří instanci standardního rozměru na zadané pozici
 *
 * @param pozice pozice zobrazení
 */
public Superman(Pozice pozice) {
```

  - velikost hlavy bude `IMPL_VELIKOST_HLAVY`, která je zděděná za třídy `Osoba`
  - správnou funkci implementace ověřte pomocí *testKonstruktorPozice()*, který před prvním použitím odkomentujte
- vytvořte bezparametrický konstruktor, který vytvoří instanci supermana v levém horním rohu plátna
  - správnou funkci implementace ověřte pomocí *testBezparKonstruktor()*, který před prvním použitím odkomentujte
- spusťte *testPresun()*, který před prvním použitím odkomentujte
  - ve výsledku testu je vidět, že je třeba překrýt některé metody z rozhraní `IPosuvny`
  - konkrétně se jedná jen o jednu metodu
  - analyzujte, o kterou metodu se jedná, a pak ji překryjte - opět jako první příkaz využijte pomocí `super.` volání metody z rodičovské třídy a dalším příkazem pak upravte novou pozici znaku
  - správnou funkci implementace ověřte pomocí *testPresun()*,
- překryjte metodu `void setPozice(Pozice p)` a otestujte ji pomocí *testSetPozicePoz()*, který před prvním použitím odkomentujte
- odkomentujte *testSetPoziceXY()* s spusťte jej (Poznámka: Tento test by měl projít na první pokus.)

- pomocí *testZvyrazneni()*, který před prvním použitím odkomentujte, ověřte, že `Superman` zdědil z `Osoba` i tuto schopnost



- pomocí testu ze třídy `TestRande` *testCeleRandeSuperman()*, který před prvním použitím odkomentujte, ověřte, že `Superman` zdědil z `Osoby` i tuto schopnost
- všechny vytvořené a upravované třídy prověřte pomocí PMD a odstraňte případné problémy
- na závěr otestujte pomocí Duck-testů celou svoji práci a odstraňte případné problémy
- celý projekt již známým způsobem zabalte do JAR souboru `07_RozsireniADedicnost.jar`, který budete odevzdávat
- rozšiřte UML diagram tříd z minulého DÚ o novou třídu `Superman`
  - pravděpodobně budete muset změnit rozmístění některých minule nakreslených tříd, nikoliv však jejich dřívější vazby
  - zakreslete správnou vazbu dědičnosti ke třídě `Osoba`

    ### Warning

    Vyberte z nabídky správnou vazbu a natáhněte ji ve směru její šipky.

    Pokud se pokusíte obrátit textovou úpravou směr vazby (šipky) v pravém dolním boxu, validátor tuto vazbu vyhodnotí jako chybnou.

  - protože třídy `Rande` a `Par` přímo nevyužívají třídu `Superman`, neměly by mezi třídami přibýt žádné další vazby
  - nezapomeňte soubor uložit pod jménem začínajícím `07` a v poznámce změnit číslo DU na 07
  - výsledek uložte do souboru `.uxf` a také exportujte jako PNG soubor (ten si zobrazte a ujistěte se, zda obsahuje všechnu informaci)
    - jména souborů budou `07_A11B0987P.uxf` a `07_A11B0987P.png` - každý samozřejmě použije své osobní číslo
    - oba tyto soubory zabalíte do souboru `07_uml.jar` příkazem

```
jar cMf 07_uml.jar 07_*.uxf 07_*.png
```

- tento soubor budete odevzdávat do **Blok 17-OOP-UML**