# 1. Domácí úloha 03

**Základní informace:**

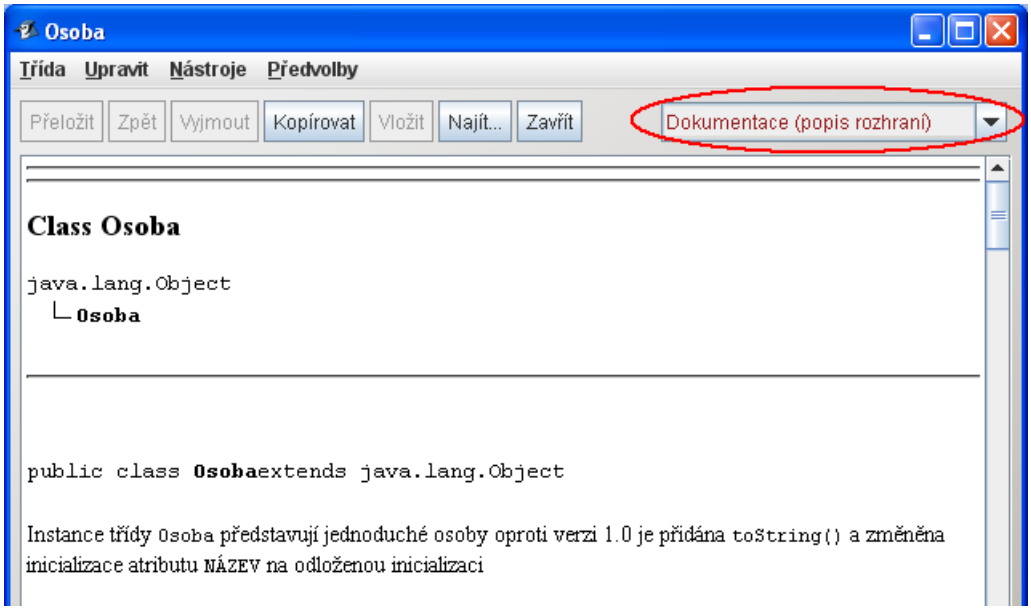
- **Účel:** procvičení práce s metodami třídy `Object`; odložená inicializace; dokumentování třídy pomocí Javadoc; test kvality kódu pomocí PMD; příprava vlastních testů; JAR soubor
- **Kostra:** `03_VylepseniTridyOsoba.zip`
- **Odevzdávaný soubor/JAR:** `03_VylepseniTridyOsoba.jar`

*Zadání:*

- upravte třídu `Osoba`, kterou jste vytvářeli v minulém DU
- upravte a doplňte třídu `TestOsoby`, která je předpřipravena v tomto projektu
- do Portálu odevzdáte JAR soubor celého projektu

**Postup řešení:**

- stáhněte si soubor `03_VylepseniTridyOsoba.zip`, rozbalte jej - NEotvírejte projekt v BlueJ
- do rozbaleného adresáře nakopírujte soubor `Osoba.java`, který jste odevzdávali v minulém DU
- v BlueJ otevřete projekt `03_VylepseniTridyOsoba`
- doplňte Javadoc dokumentaci třídě `Osoba`, všem jejím atributům, konstruktorům a metodám
  - použijte i značku `{@code }`
  - výsledek ověřte zobrazením pohledu *Dokumentace (popis rozhraní)* třídy `Osoba`, např.:




- ukázka odložené inicializace [02-str14] - **Poznámka:** odložená inicializace zde nemá praktický význam, protože inicializace atributu `NÁZEV` byla triviální - odložená inicializace je použita jen pro ukázku této techniky
  - u atributu `NÁZEV` odstraňte klíčové slovo `final` a v deklaraci jej inicializujte na `null`

- metodu `getNázev()` změňte tak, aby odloženě inicializovala atribut `NÁZEV`
- k získání názvu třídy použijte metody `getClass()` a `getSimpleName()` třídy `Object` [02-str33]
- správnost metody ověřte testem *StihlaVysokaNazev* ze třídy `TestOsoby`

■ metoda `toString()`

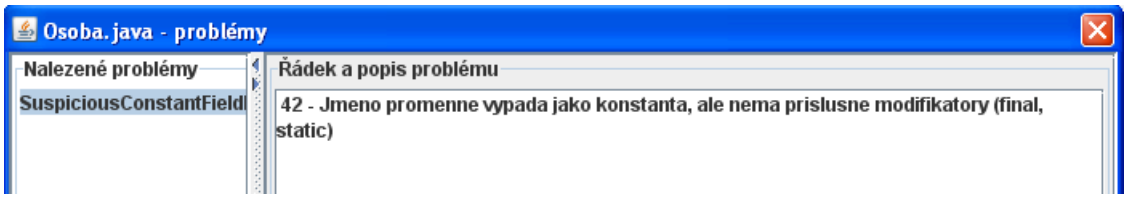
- překryjte metodu `toString()` tak, aby vracela např. řetězec:

```
Osoba_1: x=0, y=0, vyska=180, sirka=60, barva tela=hneda
```

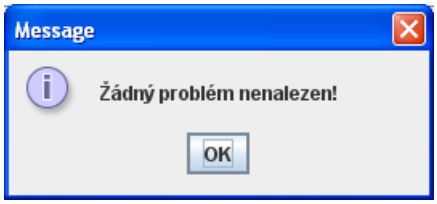
- nezapomeňte použít anotaci `@Override`
- správnost metody ověřte testem *StihlaVysokaToString* ze třídy `TestOsoby`
  - ♦ pokud test neprojde, zkuste nejprve restartovat virtuální stroj a pak test opakovat

■ nezapomeňte napsat / aktualizovat Javadoc komentáře metodám `getNazev()` a `toString()`

■ zkontrolujte kód třídy `Osoba` pomocí PMD a odstraňte případné prohřešky - kontrolujte pouze třídu `Osoba`, nikoliv všechny třídy - za ty nejste zodpovědní




- PMD vypisuje číslo řádku zdrojového kódu s chybou (zde 42) - není to číslo chyby
- v editoru zdrojových kódů lze zapnout zobrazení čísel řádků (*Nástroje / Nastavení / Editor / Zobrazovat čísla řádků*)
- reportovaný prohřešek je v tom, že `NAZEV` již není konstanta, proto by neměl být velkými písmeny
- závěrečné hlášení z PMD musí vypadat takto:




■ úprava třídy `TestOsoby` - další změny budou pouze v této třídě

- na začátek třídy doplňte údaje do `@author`
- v této třídě se téměř shodně opakuje kód testů pro `osDiteChlapec`, `osDiteDivka`, `osBeznyMuz`, `osBeznaZena`, `osNoName` a `osUrostlyGeneral`
  - ♦ Poznámka: Skutečně zde chybí původní `testStihlaVysoka()`, který testoval ještě něco navíc - nepřidávejte jej.
- opakování kódu vždy ukazuje na nedokonalost analýzy, kterou opravte

- připravte metodu `testJedneOsoby()` se:
  - signaturou

```
private void testJedneOsoby(Osoba osoba, int x, int y, int sirka, int ▶
vyska, Barva barva)
```

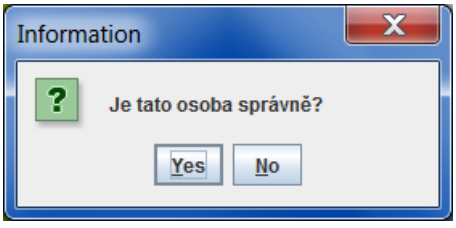
  - a kontraktem

```
/**
 * Provede základní test jedné osoby
 * testuje skutečné hodnoty dané osoby a to: souřadnice, výšku, šířku a ▶
barvu těla
 * proti očekávaným hodnotám
 * po skončení testu zobrazí dotaz pro uživatele na správnost testu
 *
 * @param osoba testovaná osoba
 * @param x očekávaná x-souřadnice
 * @param y očekávaná y-souřadnice
 * @param sirka očekávaná šířka
 * @param vyska očekávaná výška
 * @param barva očekávaná barva
 */
```

  - Poznámka: Barvu těla testujte pouze jednou a to pomocí volání metody `getBarvaTela()`
- dotaz pro uživatele na správnost testu zajistěte pomocí metody `IO.souhlas()`, jejíž kontrakt si najděte v dokumentaci ke třídě `IO` z tohoto projektu
  - návratová hodnota této metody je typu `boolean`, takže do testů se dá začlenit pomocí

```
assertEquals("Nepotvrzená správnost ", true, IO.souhlas("Je tato osoba ▶
správně?"));
```

  - tento příkaz po spuštění vytvoří potvrzovací dialog




- připravte metodu se signaturou

```
@Test
public void testRuznychOsob()
```

  - do které sloučíte všechny testy v zakomentovaných metodách `testDiteChlapec()` až `testUrostlyGeneral()`
  - vždy se nejprve vytvoří instance dané osoby (zkopírujete z původního testu) a pak ji otestujete metodou `testJedneOsoby()`

- např. celý původní (nyní zakomentovaný) test

```
// @Test
// public void testDiteChlapec()
// {
//   osDiteChlapec = new Osoba(250, 0, 30, Barva.MODRA);
//   assertEquals("Chybna x-souradnice: ", 250, osDiteChlapec.getX());
//   assertEquals("Chybna y-souradnice: ", 0, osDiteChlapec.getY());
//   assertEquals("Chybna sirka: ", 35, osDiteChlapec.getSirka());
//   assertEquals("Chybna vyska: ", 70, osDiteChlapec.getVyska());
//   assertEquals("Chybna barva: ", Barva.MODRA, ►
osDiteChlapec.getBarvaTela());
//   assertEquals("Chybná barva: ", Barva.MODRA, ►
osDiteChlapec.getTelo().getBarva());
// } ►
```

bude v nové verzi přepsán do pouze dvou řádků nově vytvářené metody `testRuznychOsob()` a bude vypadat jako:

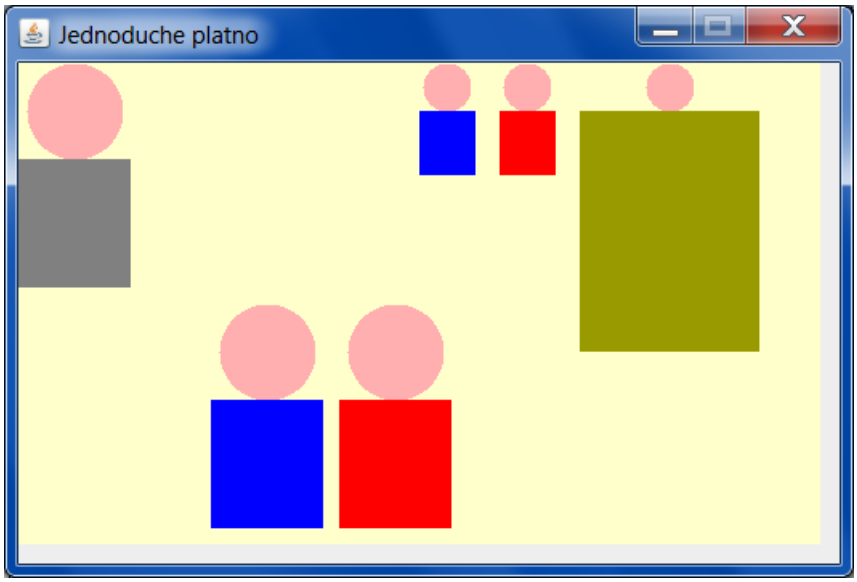
```
@Test
public void testRuznychOsob()
{
  osDiteChlapec = new Osoba(250, 0, 30, Barva.MODRA);
  testJedneOsoby(osDiteChlapec, 250, 0, 35, 70, Barva.MODRA);
```

- uvědomte si, že test má smysl pouze, když testujeme skutečné hodnoty proti očekávaným hodnotám (v naprosté většině oproti konstantám); následující ukázka testu je koncepčně chybná - testuje skutečné hodnoty proti skutečným hodnotám, takže vždy projde úspěšně

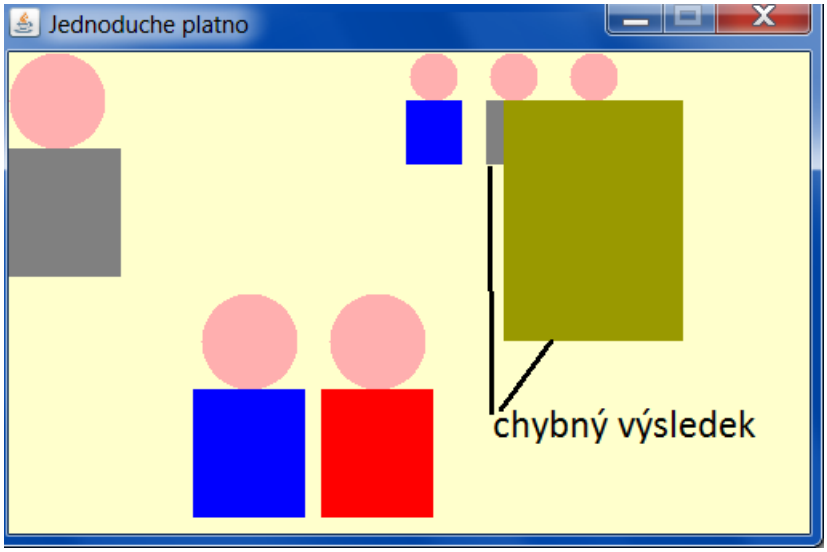
```
// ukázka chybného testu
@Test
public void testRuznychOsob()
{
  osDiteChlapec = new Osoba(250, 0, 30, Barva.MODRA);
  // logická CHYBA
  testJedneOsoby(osDiteChlapec,250,0,osDiteChlapec.getSirka(),
                 osDiteChlapec.getVyska(),osDiteChlapec.getBarvaTela());
```

- pokud je v zakomentované verzi `testDiteDivka()` provedena změna barvy těla, musí být provedena i v nové verzi testu
  - u tohoto testu již nepoužívejte čekání před změnou barvy těla dívky (`IO.cekej(500);`), protože čekání je již provedeno pomocí potvrzovacího dialogu

- výsledek testu bude



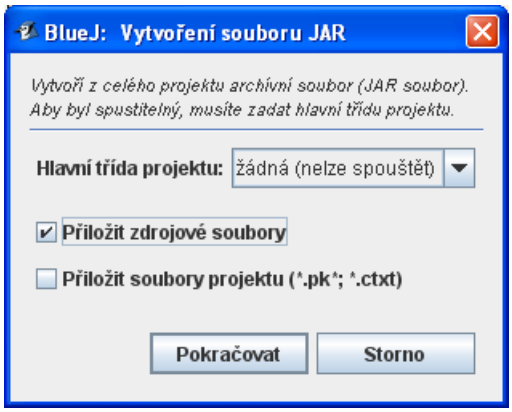

- vidíte-li jiný výsledek, (např. zde šedá dívka, špatně umístěný generál), opravte svoji třídu `Osoba`, případně doplňte test



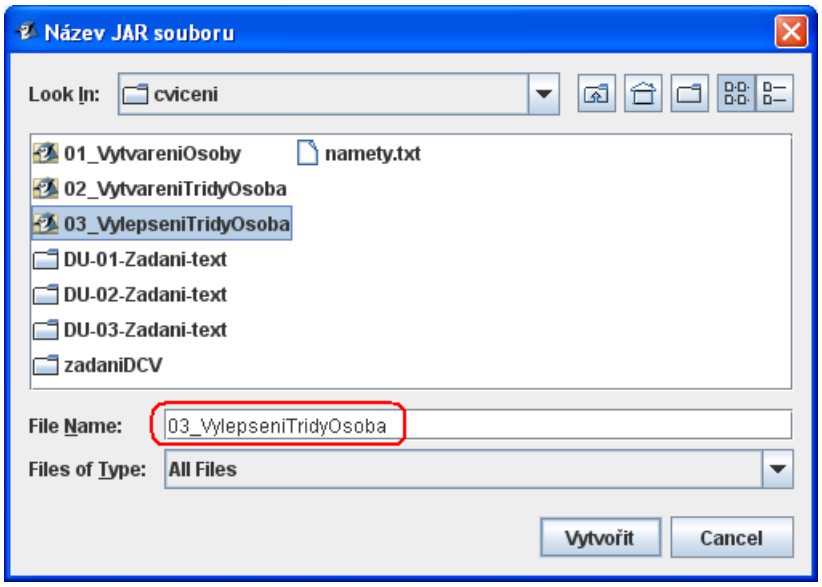

- to, že jste postupovali podle předchozího návodu (tj. vytvořili metodu `testJedneOsoby()`) ověříte testem *ExistenceTestuJedneOsoby* ze třídy `TestOsoby`
  - ♦ příslušnou metodu před použitím odkomentujte
- to, že jste v metodě `testRuznychOsob()`) otestovali všechny požadované osoby, ověříte testem *UplnostiTestuRuznychOsob* ze třídy `TestOsoby`
  - ♦ příslušnou metodu před použitím odkomentujte
- po úspěšném průchodu všech testů zakomentujte v metodě `testJedneOsoby()` řádku, na níž se čeká na potvrzení správnosti uživatelem - to znamená, že všechny testy pak proběhnou bez čekání
  - ♦ **bez tohoto zakomentování vám validátor vrátí zprávu, že vypršel časový limit**

■ příprava odevzdávaného JAR souboru

- příkazem *Projekt / Vytvořit soubor JAR* dostanete - zaškrtněte volbu *Přiložit zdrojové soubory*




- název vytvářeného JAR souboru zvolte `03_VylepseniTridyOsoba`




- vytvořený soubor `03_VylepseniTridyOsoba.jar` budete odevzdávat