# 1. Domácí úloha 05

**Základní informace:**

- **Účel:** využití aktivního plátna, rozhraní, návrhový vzor Služebník, začátky UML
- **Kostra:** `05_PlynulePosuvy.zip`
- **Odevzdávaný soubor aplikace:** `05_PlynulePosuvy.jar`
- **Odevzdávané soubory UML zabalené do JAR:** `05_uml.jar`

**Zadání:**

- upravte třídu `Osoba`, kterou jste vytvářeli v minulém DU, pro použití na aktivním plátně
- kromě jednoduchého odkomentování těl testovacích metod nijak neměňte ani neupravujte předpřipravenou třídu `TestOsoby`
- připravte rozhraní `IMeritelny` a `IZvyrazneny`
- připravte třídu `Zvyraznovac`
- do Portálu odevzdáte JAR soubor celého projektu
- dosud vytvořené třídy zdokumentujte pomocí UML diagramu tříd

**Postup řešení:**

- stáhněte si soubor `05_PlynulePosuvy.zip`, rozbalte jej - NEotvírejte projekt v BlueJ
- do rozbaleného adresáře nakopírujte soubory `Osoba.java`, `Rozmer.java` a `Pohlavi.java`, které jste odevzdávali v minulém DU
- v BlueJ otevřete projekt `05_PlynulePosuvy`
  - oproti již známým třídám přibyly další třídy a rozhraní, kterých si zatím nemusíte všímat
- upravte třídu `Osoba` tak, že bude umět využívat možnosti nového správce plátna
  - do hlavičky přidejte implementaci rozhraní `IKresleny`

```
public class Osoba implements IKresleny {
```

  - přidejte statický konstantní atribut

```
SpravcePlatna SP = SpravcePlatna.getInstance();
```

  - změňte původní metodu `nakresli()` tak, aby implementovala rozhraní `IKresleny`

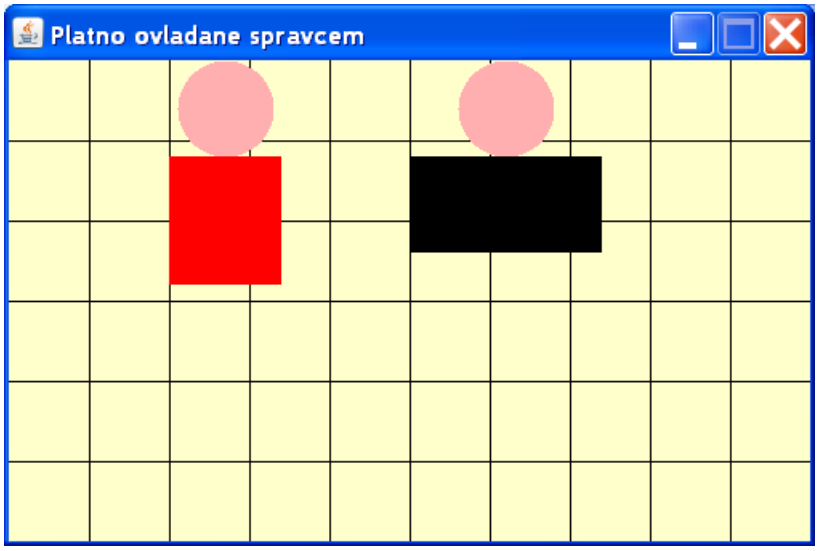
```
/**
 * vykresli instanci
 */
@Override
```

```
public void nakresli(Kreslitko kreslitko) {
  hlava.nakresli(kreslitko);
  telo.nakresli(kreslitko);
}
```

- přidejte metodu pro zaregistrování instance u `SpravcePlatna`

```
/*************************************************************
 * Prihlasi instanci u aktivniho platna do jeho spravy.
 */
public void zobraz() {
  SP.pridej(this);
}
```

- po těchto úpravách by mělo být možné třídu `Osoba` přeložit a spustit testy z dřívějších DU
  - ♦ u testu *testProhodPozice()* si všimněte, že oproti minulé DU je již zobrazení obou osob správně




- upravte třídu `Osoba` tak, aby bylo možné její instance plynule přesouvat
  - stačí jen v hlavičce třídy dodat, že implementuje rozhraní `IPosuvny`
  - vlastní implementace metod rozhraní již byla provedena v předchozím DU
  - možnost plynulého posunu ověřte pomocí *testPresouvani()*, kterou před prvním použitím odkomentujte
  - v testu si všimněte, že pro přesun využívá metody `presunO()` a `presunNa()`

### Warning

Po spuštění testu se občas stává, že se vykreslí jen rámeček plátna a program “zamrzne”. Pak pomůže restartování virtuálního stroje, případně i restartování celého BlueJ.

- připravte rozhraní `IMeritelny`, které bude mít metody `getVyska()`, `getSirka()` a `getRozmer()`
- upravte třídu `Osoba` tak, aby implementovala toto rozhraní
  - všechny metody rozhraní jsou již ve třídě `Osoba` implementovány, takže stačí pouze upravit hlavičku třídy

- správnou funkci implementovaného rozhraní ověřte pomocí *testRozmeru()*, kterou před prvním použitím odkomentujte

- připravte značkovací rozhraní `IZvyrazneny`, se implementací:

```
public interface IZvyrazneny extends IKresleny, IPosuvny, IMeritelny {
}
```

  - protože se jedná o značkovací rozhraní, nemá opravdu žádné metody
  - využívá dědičnosti rozhraní - podrobnosti budou uvedeny v další přednášce

- upravte třídu `Osoba` tak, aby implementovala toto rozhraní
  - stačí jen změnit hlavičku třídy -- všechny potřebné metody jsou již implementovány

- připravte třídu `Zvyraznovac`, která je podle návrhového vzoru Služebník a má následující kontrakt:

```
/**********************************************************
 * Instance třídy {@code Zvyraznovac} představují Služebníka,
 * který dokáže vykresit obdélník ohraničující kreslený objekt
 * na pozadí tohoto objektu.
 * Objekt musí být instancí třídy implementující rozhraní
 * {@link IZvyrazneny}.
```

  - přidejte statický konstantní atribut

```
SpravcePlatna SP = SpravcePlatna.getInstance();
```

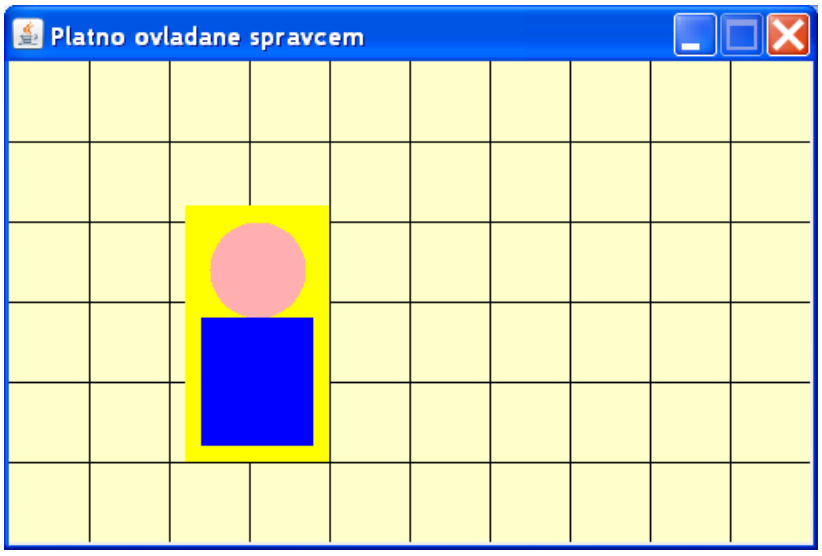
  - zaveďte statický konstantní atribut `IMPLICITNE_PRIDANO = 10`, jehož kontrakt je: “počet pixelů standardně přidaných na každou stranu zvýrazňovaného obrazce”
  - zaveďte instanční atribut `pridano` s kontraktem: “počet pixelů, které se budou skutečně přidávat na každou stranu zvýrazňovaného obrazce”
  - vytvořte konstruktory se signaturami `Zvyraznovac()` a `Zvyraznovac(int pridano)`
  - implementujte metodu se signaturou:

```
public Obdelnik zvyrazniPozadi(IZvyrazneny objekt, Barva barva)
```

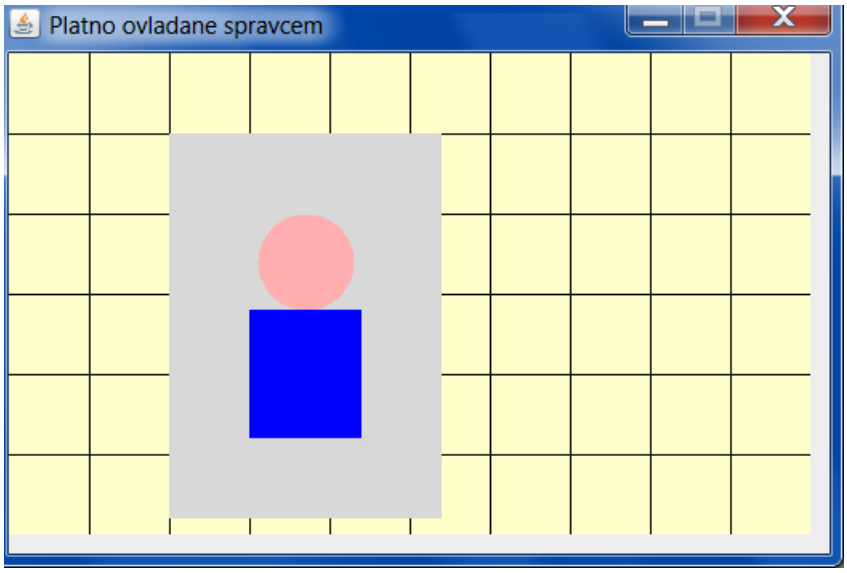
    a kontraktem:

```
  /**
   * Kreslený objekt zvýrazní tím, že danou barvou vykreslí na pozadí objektu
   * ohraničující obdélník, jehož rozměry jsou na všechny strany zvětšeny ▶
oproti
   * rozměrům zvýrazňovaného objektu.
   *
   * @param objekt zvýrazňovaný objekt
   * @param barva barva zvýrazňujícího obdélníka napozadí
   * @return zvýrazňující obdélník
   */
```

- metoda musí získat ze zvýrazňovaného objektu `Pozici` a `Rozmer` a všechny jejich hodnoty upravit o požadované zvětšení
- zvýrazňující obdélník je třeba umístit na plátno dospodu - příslušnou metodu `pridejDospod()` ověřte v dokumentaci třídy `SpravcePlatna`
- vrácená reference na zvýrazňující obdélník bude použita pro testovací účely a později bude použita pro odstranění zvýrazňování
- funkčnost implementace ověřte pomocí *testZvyrazneniPozadi()*, kterou před prvním použitím odkomentujte




- správnou funkci konstruktoru `Zvyraznovac(int pridano)` otestujte pomocí *testZvyrazneniPozadiVelkymObdelnikem()*



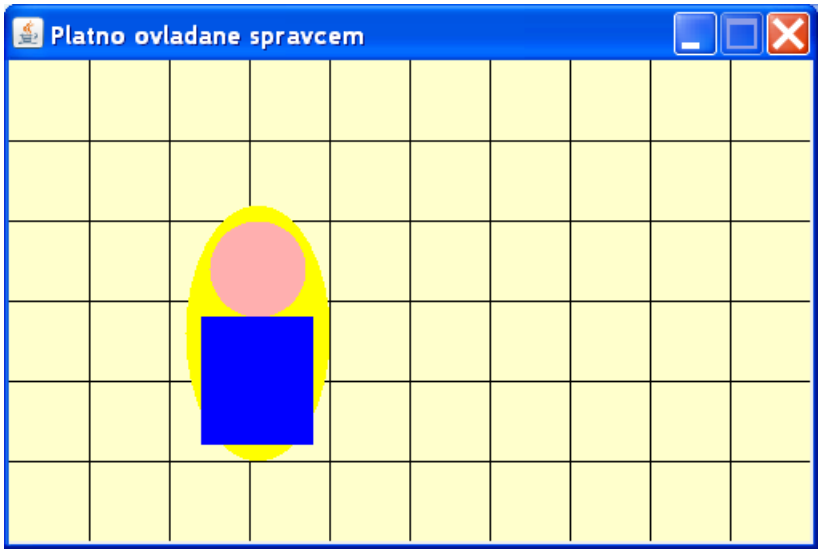

- implementujte metodu se signaturou:

```
public Elipsa zvyrazniPozadiElipsou(IZvyrazneny objekt, Barva barva)
```

a velmi podobným kontraktem jako má metoda `zvyrazniPozadi()` - liší se pouze v tom, že se zvýrazňuje elipsou

- metoda je téměř úplnou kopií metody `zvyrazniPozadi()`

- funkčnost implementace ověřte pomocí *testZvyrazneniPozadiElipsou()*, kterou před prvním použitím odkomentujte




- protože se kód v metodách `zvyrazniPozadi()` a `zvyrazniPozadiElipsou()` opakuje, proveďte **reinženýring** (tj. opravu) kódu
  - prozkoumejte kontrakt Přepravky `Oblast` - nachází se již kompletně hotová v tomto projektu
  - připravte metodu se signaturou

```
public Oblast zjistiOblastZvyrazneni(IZvyrazneny objekt)
```

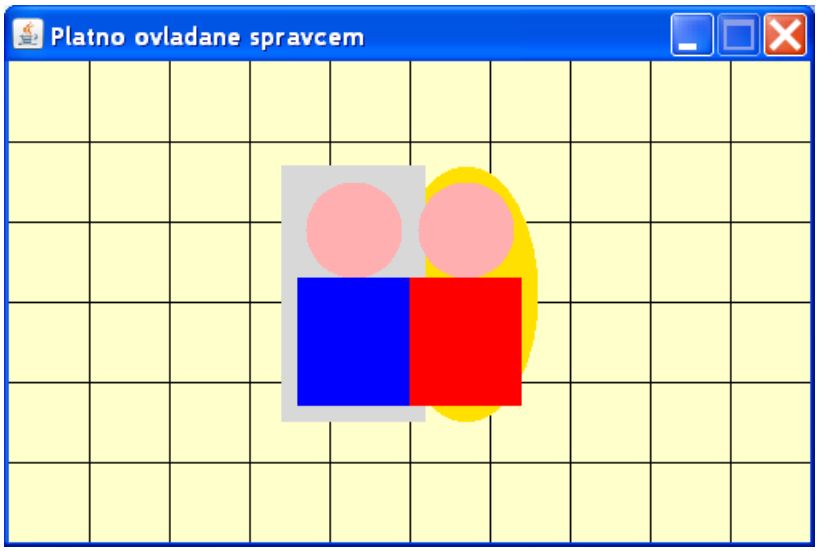
    a kontraktem

```
/**
 * Vrátí Oblast zvýrazňovaného objektu zvětšenou na všechny strany
 *
 * @param objekt zvýrazňovaný objekt
 * @return oblast, ve které bude později vykreslen zvýrazňující tvar
 */
```

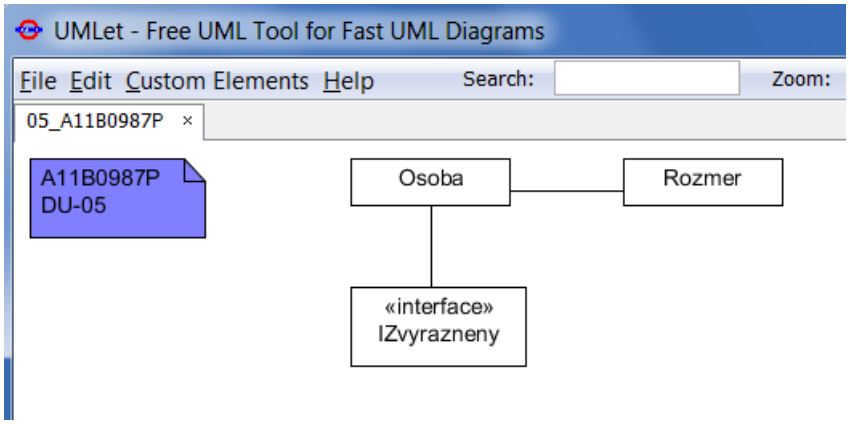
  - **Poznámka**: Metoda je pomocná, proto by měla být `private` - `public` je jen z toho důvodu, aby ji bylo možné z vnějšku otestovat
  - tělo této metody bude tvořit společná (tzn. zcela stejná) část kódu z metod `zvyrazniPozadi()` a `zvyrazniPozadiElipsou()`
  - funkčnost implementace ověřte pomocí *testZvyraznenOblasti()*, kterou před prvním použitím odkomentujte
    - zde se nic nevykresluje na plátno, důležité jsou výsledky testů
  - upravte metody `zvyrazniPozadi()` a `zvyrazniPozadiElipsou()` tak, aby využívaly metodu `zjistiOblastZvyrazneni()`
    - obě metody se tím výrazně zkrátí
    - u `Obdelnik` i `Elipsa` již existuje jeden přetížený konstruktor, který používá `Oblast`

- funkčnost nové implementace obou metod ověřte pomocí *testZvyrazneniPozadi()* a *testZvyrazneni-PozadiElipsou()*

- spusťte *testPresunuAZvyrazneni()*, který před prvním použitím odkomentujte

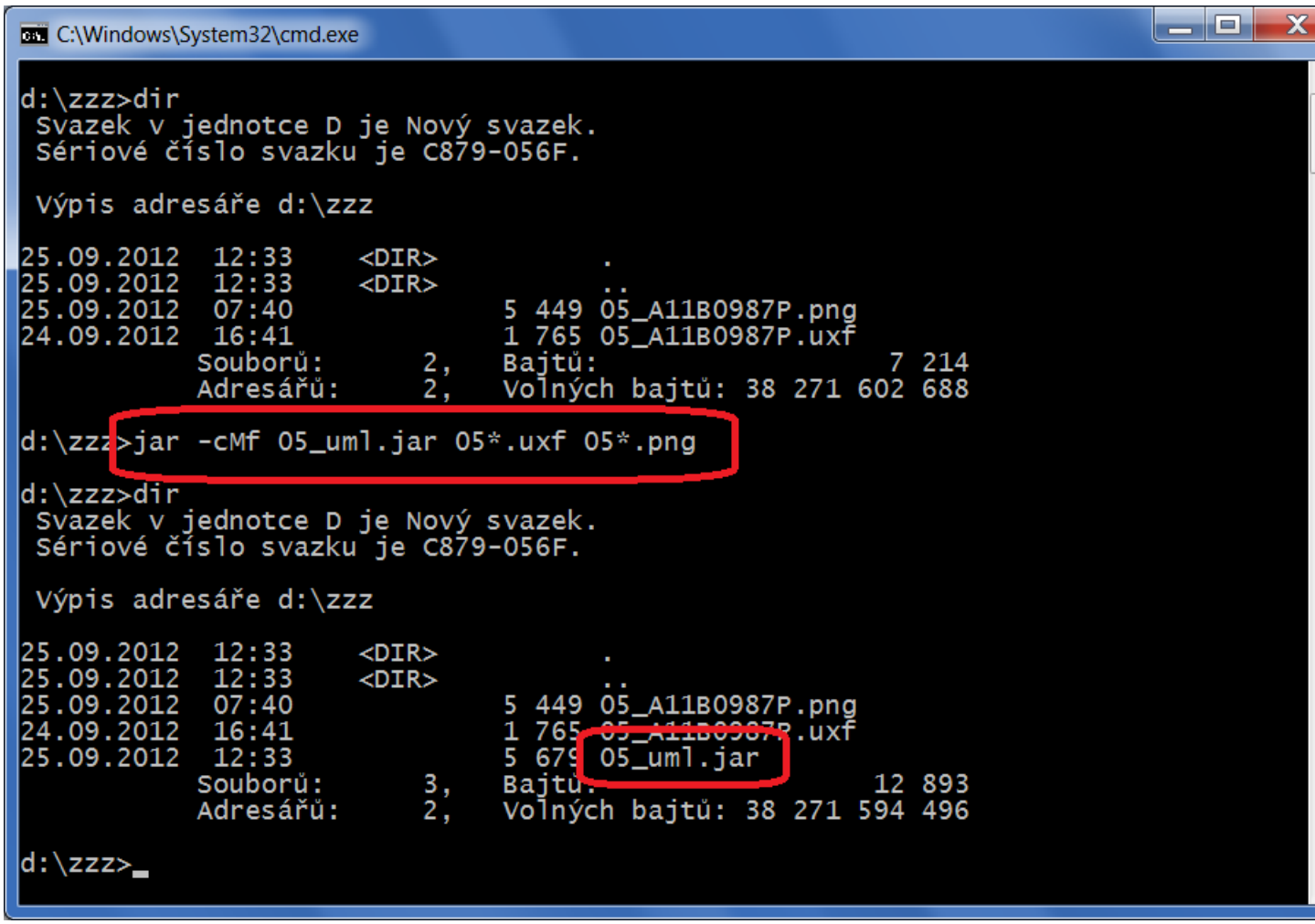



- všechny vytvořené a upravované třídy prověřte pomocí PMD a odstraňte případné problémy
- na závěr otestujte pomocí Duck-testů celou svoji práci a odstraňte případné problémy
- celý projekt již známým způsobem zabalte do JAR souboru `05_PlynulePosuvy.jar`, který budete odevzdávat
- UML diagram tříd pomocí nástroje UMLet
  - do levého horního rohu dejte poznámku, ve které bude vaše osobní číslo a číslo DU, tj. DU-05
  - nakreslete diagram tříd, rozhraní a výčtových typů, které jste dosud vytvořili, tzn. pouze: `Osoba`, `Pohlavi`, `Rozmer`, `IZvyrazneny`, `Zvyraznovac`, `IMeritelny`
    - nekreslete třídy, které jste sami nevytvářeli, např. `Obdelnik`,atp.
    - nekreslete třídy testů, např. `TestOsoby`
  - v diagramu tříd použijte zjednodušené schéma třídy (tj. pouze s názvem třídy, bez atributů a metod)
    - u rozhraní a výčtových typů nezapomeňte uvést příslušné stereotypy
  - objekty prozatím spojte pouze asociačními vazbami (plná čára bez popisů), např.:



- zaměřte se na přehledné rozmístění objektů na nákresu
- výsledek uložte do souboru `.uxf` (budete jej odevzdávat a dále s ním pracovat) a také exportujte jako PNG soubor
  - jména souborů budou `05_A11B0987P.uxf` a `05_A11B0987P.png` - každý samozřejmě použije své osobní číslo
  - oba tyto soubory zabalíte do souboru `05_uml.jar` příkazem

```
jar cMf 05_uml.jar 05_*.uxf 05_*.png
```




  - soubor `05_uml.jar` budete odevzdávat do **Blok 15-OOP-UML**
    - soubor PNG nebude validován, soubor UXF ano
- cílem akce je naučit se základním způsobem ovládat nástroj UMLet a připravit si základ diagramu tříd pro další rozšiřování a zpřesňování

### Note

Nenechávejte si diagram tříd automaticky vygenerovat službami poskytovanými např. Eclipse (a přepisovat jej pak do UMLetu)! Je třeba, abyste diagram tříd vytvářeli sami a přemýšleli při tom.