1. Projekt oks-04

Základní informace:

- **Účel:** prakticky se seznámit s technikou využití *mock* objektů
- **Kostra:** `oks-data-04.zip`
- **Odevzdávaný soubor:** `oks-04.jar`

Zadání:

- otestujte doménovou třídu JUnit testy s využitím *mock* objektů

Popis vstupních dat:

- stáhněte si soubor `oks-data-04.zip` a rozbalte jej
  - všechny soubory jsou v kódování UTF-8 bez BOM
- v adresáři `src/oks04` se nachází zdrojové `.java` soubory, které jsou velmi podobné těm z projektu `oks-01` (drobné změny jsou z důvodu lepšího testování pomocí *mock* objektů)
  - k výrazné změně došlo pouze u třídy `Generator`
    - ta je poměrně hodně přeprogramovaná (ale filosofie zpracování zůstala stejná), aby byly jednotlivé metody lépe testovatelné
    - zásadní změny se týkají vypuštění atributů (zneužívaných jako “globální proměnné”) a jejich nahrazení formálními parametry a návratovými hodnotami jednotlivých metod
    - dále je dodržována zásada, že metoda má dělat jen jednu činnost, která je v jejím názvu
      - to např. znamená, že pokud se v `oks01.Generator` vyskytuje metoda `pripravSkupinyARoztridDoNich()`, pak porušuje toto pravidlo, protože dělá dvě věci a tudíž se hůře testuje
    - pro zajímavost si porovnejte obsahy `oks01.Generator` a `oks04.Generator` a subjektivně zvažte čitelnost/pochopitelnost kódu
- v adresáři `test/oks04` se nachází zdrojové `.java` soubory, které budou použity pro testování
  - `GeneratorTest.java` - kostra třídy testů, kterou budete doplňovat
  - `TestovaciData.java` - kostra třídy, pomocí níž budete připravovat testovací data pro *mock* objekt
- v adresáři `data` se nachází datový soubor
  - adresář obsahuje následující soubor:
    - `priklady-oks-04-1.txt` - vstupní data, totožná s daty z projektu `oks-01`
      - budete je používat pouze pro inspiraci naplnění třídy `TestovaciData`
- v adresáři `kontrola` se nachází kontrolní výsledky

- adresář obsahuje následující podadresáře a soubor:
  - `oks04` - adresář s testovanými `.class` soubory
  - `vysledek` - adresář se vzorovým souborem výsledku `vzor-vysledky-testu-04.txt`
  - `kontrola.bat` - překlad testovacích tříd a spuštění kontrolního programu

■ v adresáři `lib` se nachází knihovny JUnit 5, Mockito a další potřebné knihovny

- adresář obsahuje 17 `.jar` souborů, které přidáte do *ClassPath* v IntelliJ
  - použijete postup známý z dřívějších projektů

> **Note**
>
> Pro řešení úlohy nejsou všechny `.jar` soubory zapotřebí, ale jsou nutné pro závěrečnou kontrolu. Výčet jen těch, které nutně potřebujete pro řešení, by byl matoucí.

Postup řešení:

■ v IntelliJ založte nový projekt `oks-prj-04`

- nastavte celému projektu kódování UTF-8

  File / Settings / Editor / File encodings / Project encoding: UTF-8

- do adresáře `src` projektu přetáhněte celý adresář `src/oks04`
  - tento adresář ani soubory v něm nebudete nijak měnit
- vytvořte adresář `test` a pomocí kontextového menu a **Mark Directory as / Test Sources Root** jej nastavte jako adresář testů
  - přetáhněte do něj celý adresář `test/oks04`
  - většinu aktivit budete provádět v tomto adresáři

■ třídu `Generator` nelze přeložit, protože není dostupná třída, která poskytuje data

- tuto situaci vyřešíte provizorně (pro účely testování) tak, že v `src/oks04` vytvoříte *bussines interface* `ICteniDat`, které bude mít jednu metodu `nactiVsechnaOsobniCisla()`

■ prohlédněte si třídu `TestovaciData` a do její metody `getTestovaciData()` podle vzoru doplňte ze souboru `priklady-oks-04-1.txt` dostatečné (nenulové) množství “načítaných” kontrolních dat

- pozor na to, aby studenti byli z FAV, která je většinou testována

■ doplňte kostru třídy `GeneratorTest`, ve které:

- v metodě `setUp()`:
  - neměňte řádku

```
testovaciData = TestovaciData.getTestovaciData();
```

ta dává možnost nastavit seznam dat i externě (a nezávisle na čtení souboru), což bude ve validátoru využito pro další ověřování funkčnosti třídy `GeneratorTest`

- ♦ připravíte *mock* objekt simulující čtení dat
- ♦ vytvoříte testovaný objekt třídy `Generator`
- ♦ vyvoláte nad ním jeho metodu `zpracovani()`, přičemž samozřejmě nezáleží na jménu načítaného datového souboru
  - – jako druhý skutečný parametr zadejte `"fav"`
  - – tato metoda vrací `List<List<OsobniCislo>>`, kteroužto hodnotu uložte do příslušného atributu - budete ji potřebovat pro testy

- v metodě `tearDown()`:
  - ♦ ukončete práci s *mock* objektem voláním metody `verify()`
- dále připravte tři testy
  - ♦ `getSeznamTypuStudia_bakalari()`
  - ♦ `getSeznamTypuStudia_navazujici()`
  - ♦ `getSeznamTypuStudia_doktorandi()`
  - ♦ tyto metody:
    - – budou mít velmi jednoduché tělo - vždy dva příkazy
    - – budou ověřovat, zda počet načtených platných osobních čísel v konkrétní skupině souhlasí
      - k tomu využívají skutečnosti, že v metodě `setUp()` jsou nastaveny `seznamyTypuStudia`
      - v metodě `assertEquals()` bude samozřejmě konkrétní počet platných osobních čísel, které jste v dané skupině připravili ve třídě `TestovaciData`
        - ■ tento počet může být odlišný od řešení vašich spolužáků, podstatné je, že test musí projít
        - ■ na validátoru pak bude použita jiná třída `TestovaciData`
    - – bezpodmínečně dodržte názvy metod
- dále připravte testovací metodu `vytvorSeznamChybnychFormatu()`
  - ♦ zde si uvědomte, že v metodě `setUp()`, která je volána na začátku každé (i této) metody, se nastaví atribut `testovaciData`
  - ♦ nevyužívá se tak mockování, ale přímo atribut `testovaciData` nastavený třídou `TestovaciData`
    - – je to ukázka, že při testování s využitím mockování můžeme dle potřeby paralelně použít i jiné způsoby přípravy dat

■ třídu `GeneratorTest` spusťte v IntelliJ jako JUnit test

- pokud všechny čtyři testy neprojdou, odstraňte příčinu problémů

- práci v IntelliJ končíte v okamžiku, kdy projdou všechny testy třídy `GeneratorTest`

Kontrola úplnosti řešení:

- adresář `test` z IntelliJ překopírujte někam do svého pomocného adresáře, např. `D:\zzz`
  - dále sem překopírujte obsah adresáře `kontrola/oks04` a soubor `kontrola.bat`
  - dále sem překopírujte obsah adresáře `lib`
- spusťte soubor `kontrola.bat` nebo z něj použijte příslušné příkazy
  - Pozor - pokud pracujete v Linuxu, bude zřejmě nutné v nastavení *classpath* změnit oddělovač souborů z ;

    ```
    -cp apiguardian-api-1.0.0.jar;junit-jupiter-api-5.3.2.jar ...
    ```

    na :

    ```
    -cp apiguardian-api-1.0.0.jar:junit-jupiter-api-5.3.2.jar ...
    ```

- porovnejte obsahy souborů `vysledky-testu-04.txt` a `vysledek/vzor-vysledky-testu-04.txt`
  - pokud nebudou identické, tak ani neodevzdávejte a hledejte příčinu rozdílu
- vzniklý soubor `vysledky.txt` můžete vizuálně porovnat s obsahem třídy `TestovaciData`

Příprava souborů k odevzdání:

- v adresáři, ve kterém byla prováděna kontrola, použijte příkaz:

  ```
  jar cMf oks-04.jar test
  ```

- výsledkem bude soubor `oks-04.jar`, který budete odevzdávat
  - opravdu není spustitelný, neobsahuje `.class` soubory ani Javadoc
  - jeho velikost by měla být do 10 KB - pokud je větší, je v něm něco navíc
- po úspěšném odevzdání na Portále, proklikněte výsledné OK a, prosím, vyplňte v tabulce časovou náročnost této úlohy