

# 1. Projekt oks-05

Základní informace:

- **Účel:** prakticky se seznámit se strukturálními testy a zjišťováním možností pokrytí kódu
- **Kostrá:** oks-data-05.zip
- **Odevzdávaný soubor:** oks-05.jar

Zadání:

- strukturálními testy zajistíte 100% pokrytí řádek, větví a cyklomatické složitosti u třídy `OsobniCislo`

Popis vstupních dat:

- stáhněte si soubor `oks-data-05.zip` a rozbalte jej
  - všechny soubory jsou v kódování UTF-8 bez BOM
- v adresáři `src/oks05` se nachází zdrojové `.java` soubory, které jsou velmi podobné těm z projektu `oks-01`
  - adresář obsahuje již známé soubory - ani jeden z nich nebudete měnit:
    - ♦ `Konstanty.java` - konstanty pro třídu `OsobniCislo`
    - ♦ `TypStudia.java` - výčtový typ pro třídu `OsobniCislo`
    - ♦ `OsobniCislo.java` - třída, kterou budete pokrývat testy
    - ♦ `Hlavni.java` - třída aplikace, kterou spustíte
- v adresáři `data` se nachází soubor **pro inspiraci**, jak mohou vypadat testovací data
  - adresář obsahuje následující soubor:
    - ♦ `priklady-oks-05-1.txt` - vstupní data, totožná s daty z projektu `oks-01`
- v adresáři `kontrola` se nachází soubory pro porovnání s výsledky získanými kontrolou
  - adresář obsahuje následující podadresář a soubor:
    - ♦ `oks05` - adresář s testovanými `.class` soubory
    - ♦ `kontrola.bat` - překlad a spuštění kontrolního programu
- v adresáři `lib` se nacházejí JAR knihovny JUnit 5
  - adresář obsahuje sedm `.jar` souborů

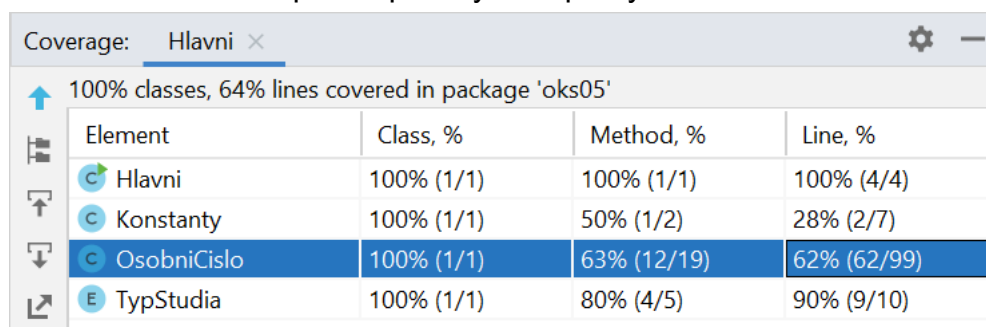
Postup řešení:

- v IntelliJ založte nový projekt `oks-prj-05`
  - nastavte celému projektu kódování UTF-8

- do adresáře `src` projektu přetáhněte celý adresář `src/oks05`
  - ♦ tento adresář ani soubory v něm nebudete nijak měnit ani doplňovat
- vytvořte adresář `test` a pomocí kontextového menu a **Mark Directory as / Test Sources Root** jej nastavte jako adresář testů
  - ♦ všechny aktivity budete provádět v tomto adresáři

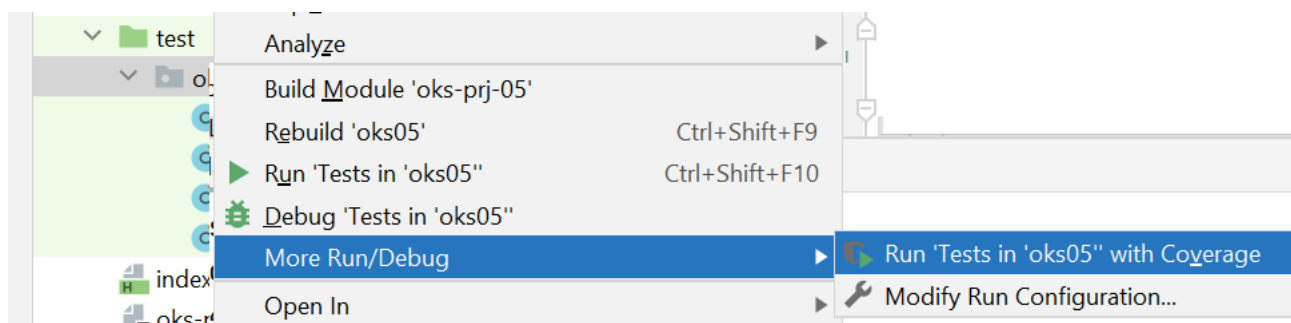
#### ■ spusťte třídu `Hlavni` s analýzou pokrytí

- na konzoli se vypíše vygenerované osobní číslo
- ve třídě `OsobniCislo` se zobrazí překvapivě vysoké pokrytí

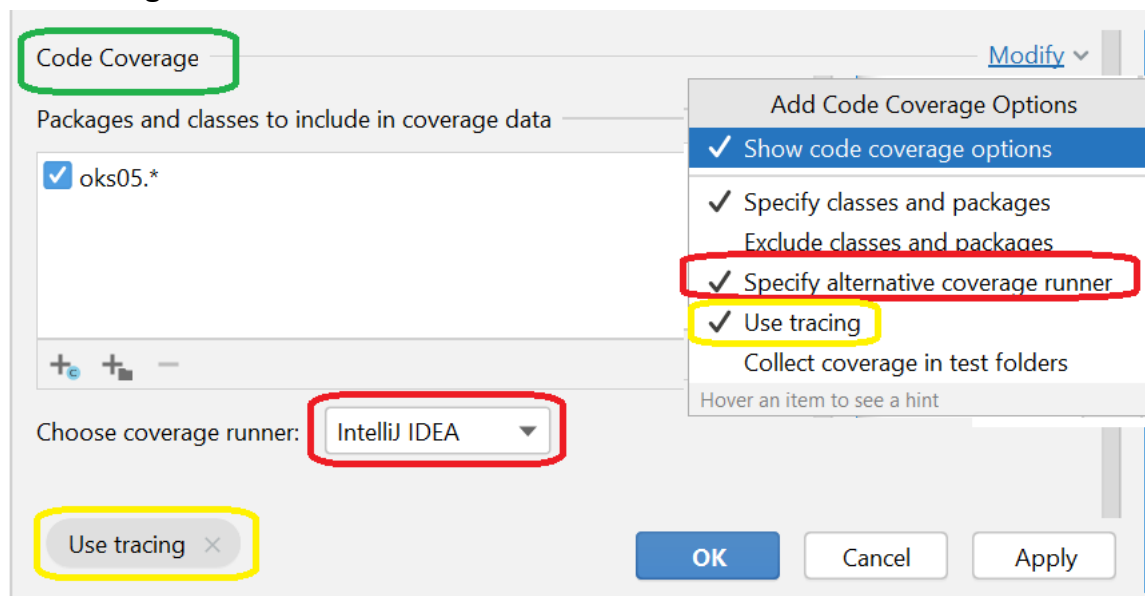


Element	Class, %	Method, %	Line, %
Hlavni	100% (1/1)	100% (1/1)	100% (4/4)
Konstanty	100% (1/1)	50% (1/2)	28% (2/7)
OsobniCislo	100% (1/1)	63% (12/19)	62% (62/99)
TypStudia	100% (1/1)	80% (4/5)	90% (9/10)

- ♦ příkazy v konstruktoru a inicializované atributy budou zelené
  - ♦ některé metody volané (i nepřímo) z konstruktoru budou mít některé příkazy též zelené
  - ♦ getry budou červené
  - výsledky ukazují, že analyzátor pokrytí pracuje
- #### ■ v `test/oks05` vytvořte třídu `OsobniCisloTest_PozitivniPolotovar`, která bude zajišťovat (pouze) **pozitivní** testy:
- nechte si předvygenerovat testy pro všechny metody ze třídy `OsobniCislo` včetně konstruktoru, ale bez testu metody `generujOsobniCislo()`
  - v metodě `setUp()`:
    - ♦ vytvořte testovaný objekt třídy `OsobniCislo` se správnými parametry (jako ve třídě `Hlavni`)
      - použijte pouze konstruktor (nepoužívejte následné volání `generujOsobniCislo()` jak je to ve třídě `Hlavni`)
  - třída `OsobniCislo` je bez chyb, tj. můžete se spolehnout na správnou funkci `getru`
  - spusťte testy z `test/oks05` s pokrytím



- v **Run / Edit Configuration** nastave:



- ♦ podle návodu v přednáškách spusťte několikrát testy a dělejte pokusy se zobrazováním pokrytím, jeho vypínáním apod.
- pak ve třídě `OsobniCisloTest_PozitivniPolotovar` doplňujte systematicky jednotlivá těla předgenerovaných vybraných testovacích metod (píšete pozitivní testy!) a průběžně sledujte zvyšující se pokrytí
  - ♦ těla budou velmi jednoduchá - jeden až dva příkazy
    - testovaná instance třídy `OsobniCislo` byla vytvořena v metodě `setUp()` - viz výše
  - ♦ uvědomte si, že testujete polotovar osobního čísla - místo čísla pořadí je řetězec `xxxx`
  - ♦ všechny příkazy `assertXY()` pište včetně chybového hlášení, např.:
 

```
assertEquals(očekáváno, skutečnost, "Chyba: první je menší");
```
  - ♦ nezaměřujte se jen na sledování pokrytí, důležité je i to, že testy procházejí
    - nejdříve vždy spusťte právě vytvářený JUnit test a ihned zkontrolujte, zda dává výsledek dle vašich představ
  - ♦ některé metody budete muset otestovat více pozitivními testy
    - pak dodržujte dřívější konvenci v pojmenovávání testů - odlišení pořadovými čísly, např.:

```
void compareTo_1() { ...  
void compareTo_2() { ...
```

- ♦ nezapomeňte, že v této třídě testů jsou pouze pozitivní testy předpřipraveného osobního čísla
- ♦ nepište bezhlavě JUnit testy všech předvygenerovaných metod (typicky metody `zpracujXY()`)
  - mnohé tyto metody jsou pokryty už jen tím, že jsou volány z jiné metody, což zjistíte snadno pohledem na obarvený zdrojový kód `OsobniCislo`
  - předvygenerované metody, které nebude nutné implementovat, přesunete do třídy `OsobniCisloTest_Negativni`
- vytvořte třídu `OsobniCisloTest_Negativni`, která bude zajišťovat testy v případě chybných vstupů:
  - do této třídy přesuňte všechny neimplementované metody ze třídy `OsobniCisloTest_PozitivniPolotovar`
    - ♦ většinu z nich ale nebudete implementovat, protože budou pokryty nepřímo negativními testy metody `naplnAtributy()`
    - ♦ ovšem jen s těmito testy nevystačíte, budete muset napsat i negativní verze již existujících pozitivních testů (např. `testToString()`)

## Note

Tato metoda se nemůže jmenovat `toString()`, protože by byla zaměnitelná se standardní metodou `toString()`.

- budete postupovat systematicky a budete psát negativní testy tak, aby byly pokryty všechny dosud nepokryté řádky a podmínky
- opět pracujte postupně, kdy nejdříve napíšete a spustíte příslušný JUnit test, který musí projít
  - ♦ v tomto projektu by se neměl vyskytnout žádný JUnit test, který selže
    - to by znamenalo, že ve třídě `OsobniCislo` je zapomenutá chyba
- pokud projde JUnit test, spustěte analýzu pokrytí a zjistěte, jak tento test vylepšil pokrytí
- nepoužívejte metodu `setUp()`:
  - ♦ testovaný objekt třídy `OsobniCislo` s příslušnými chybami vytvořte vždy “na míru” v testovací metodě, např.:

```
@Test  
void zpracujRokNastupu_1() {  
    OsobniCislo oc = new OsobniCislo("");  
    oc.zpracujRokNastupu("12345");  
}
```

- při doplňování jednotlivých ostatních předgenerovaných metod si uvědomte, že některé metody bude již zbytečné znovu testovat
  - ♦ již byly pokryty testy z “pozitivní” třídy

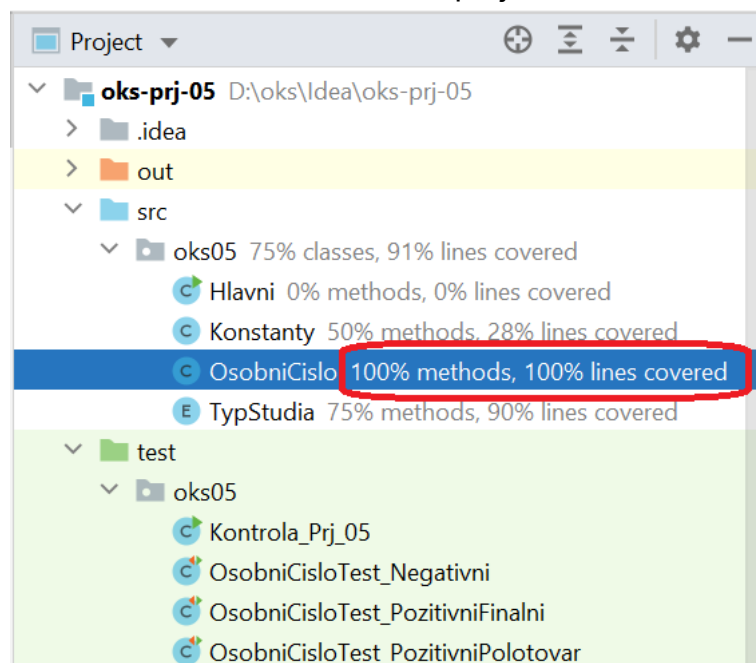
- na závěr vytvořte třídu `OsobniCisloTest_PozitivniFinalni`, která vznikne jako kopie třídy `OsobniCisloTest_PozitivniPolotovar`
  - bude zajišťovat testy v případě dokončení vytváření osobního čísla
  - v metodě `setUp()` dodejte volání metody `generujOsobniCislo()`
  - pak ponechte pouze těla metod `XY()`, které pracují s finální podobou osobního čísla
    - ♦ ty změňte tak, aby bylo dosaženo 100% pokrytí
    - ♦ všechny ostatní testovací metody vymažte
- budete pravděpodobně potřebovat nejméně 20 testů dohromady ve všech třech testovacích třídách
- práce končí v okamžiku, kdy:
  - z pohledu vývojáře
    - ♦ všechny řádky zdrojového kódu třídy `OsobniCislo` jsou pouze zelené (ne žluté, či červené)
  - z pohledu manažera - metody, řádky i větve jsou pokryty na 100 %

Coverage: oks05 in oks-prj-05

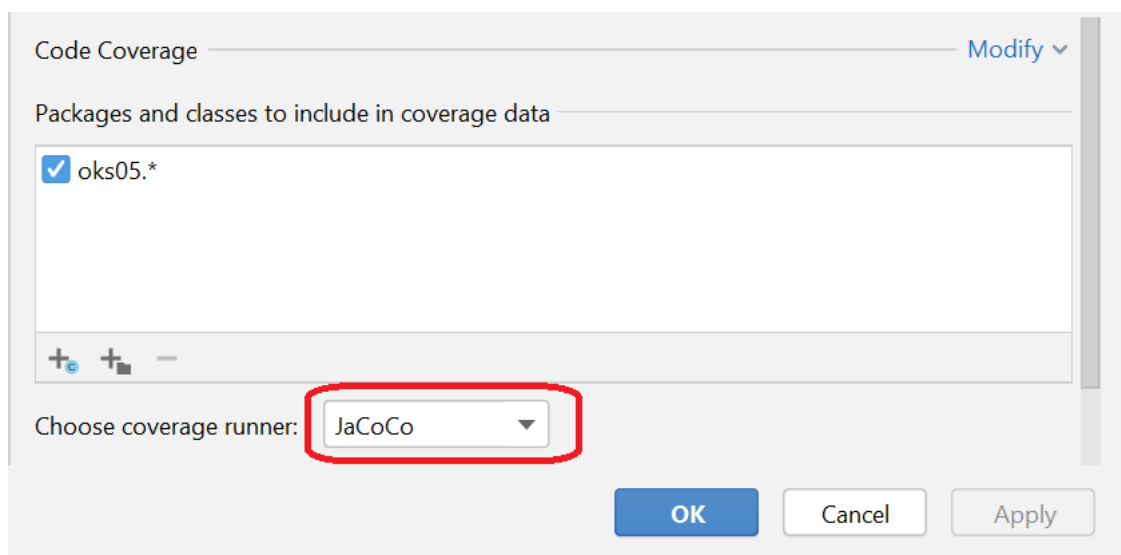
75% classes, 91% lines covered in package 'oks05'

Element	Class, %	Method, %	Line, %	Branch, %
Hlavni	0% (0/1)	0% (0/1)	0% (0/4)	100% (0/0)
Konstanty	100% (1/1)	50% (1/2)	28% (2/7)	0% (0/4)
<b>OsobniCislo</b>	<b>100% (1/1)</b>	<b>100% (19/19)</b>	<b>100% (99/99)</b>	<b>100% (42/42)</b>
TypStudia	100% (1/1)	75% (3/4)	90% (9/10)	100% (0/0)

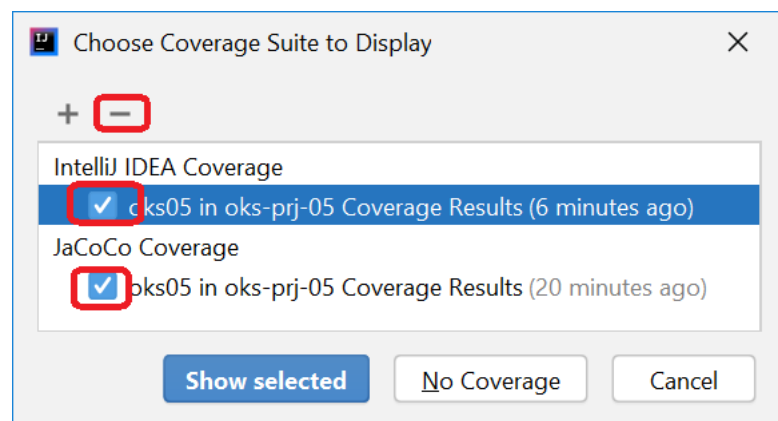
- stejná informace je vidět i vlevo u seznamu souborů v projektu



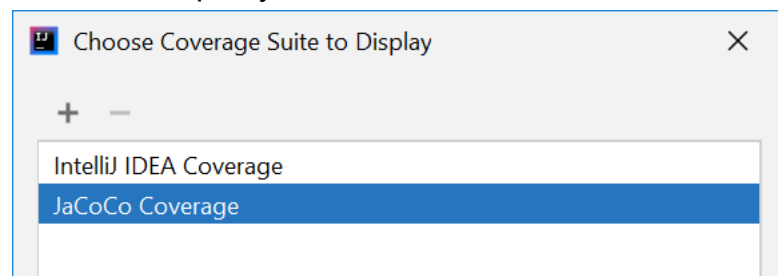
- jako poslední sérii akcí v IntelliJ provedete:
  - přepnete na měření pokrytí pomocí JaCoCo



- **Run / Show Coverage Data** pro jistotu vymažete všechna předchozí měření, abyste nevyexportovali nějaké staré pokrytí



- dostanete prázdný seznam měření pokrytí



- spustíte celou analýzu pokrytí zcela stejně (ale jiným analyzátořem) ještě jednou (dostanete trochu odlišná čísla řádek) a výsledek vyexportujete do HTML

Coverage: oks05 in oks-prj-05

75% classes, 91% lines covered in package 'oks05'

Element	Class, %	Method, %	Line, %	Branch, %
Hlavni	0% (0/1)	0% (0/1)	0% (0/4)	100% (0/0)
Konstanty	100% (1/1)	50% (1/2)	28% (2/7)	0% (0/4)
OsobniCislo	100% (1/1)	100% (19/19)	100% (99/99)	100% (42/42)
TypStudia	100% (1/1)	75% (3/4)	90% (9/10)	100% (0/0)

- v HTML zkontrolujete 100% pokrytí

## OsobniCislo

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
● <a href="#">naplnAtributy(String)</a>	<div></div>	100%	<div></div>	100%	0	7	0	20	0	1
● <a href="#">zpracujFakulta(String)</a>	<div></div>	100%	<div></div>	100%	0	3	0	7	0	1
● <a href="#">zpracujTypStudia(String)</a>	<div></div>	100%	<div></div>	100%	0	3	0	7	0	1
● <a href="#">zpracujFormaStudia(String)</a>	<div></div>	100%	<div></div>	100%	0	3	0	7	0	1
● <a href="#">OsobniCislo(String)</a>	<div></div>	100%		n/a	0	1	0	6	0	1
● <a href="#">zpracujJmeno(String)</a>	<div></div>	100%	<div></div>	100%	0	2	0	8	0	1
● <a href="#">zpracujRokNastupu(String)</a>	<div></div>	100%	<div></div>	100%	0	3	0	13	0	1
● <a href="#">compareTo(OsobniCislo)</a>	<div></div>	100%	<div></div>	100%	0	2	0	4	0	1
● <a href="#">toString()</a>	<div></div>	100%	<div></div>	100%	0	2	0	4	0	1
● <a href="#">generujOsobniCislo(String)</a>	<div></div>	100%		n/a	0	1	0	2	0	1
● <a href="#">isPlatneOsobniCislo()</a>	<div></div>	100%	<div></div>	100%	0	3	0	4	0	1
● <a href="#">zpracujPrijmeni(String)</a>	<div></div>	100%	<div></div>	100%	0	2	0	5	0	1
● <a href="#">zpracujNepovinne(String)</a>	<div></div>	100%	<div></div>	100%	0	2	0	4	0	1
● <a href="#">static {...}</a>	<div></div>	100%		n/a	0	1	0	1	0	1
● <a href="#">chybnyRokNastupu()</a>	<div></div>	100%		n/a	0	1	0	3	0	1
● <a href="#">getOsobniCislo()</a>	<div></div>	100%		n/a	0	1	0	1	0	1
● <a href="#">isPlatnyFormat()</a>	<div></div>	100%		n/a	0	1	0	1	0	1
● <a href="#">getTypStudia()</a>	<div></div>	100%		n/a	0	1	0	1	0	1
● <a href="#">getFakulta()</a>	<div></div>	100%		n/a	0	1	0	1	0	1
Total	0 of 387	100%	0 of 42	100%	0	40	0	99	0	19

- na disku v adresáři, kam byly vyexportovány výsledky, naleznete soubor `oks05/index.html` (nikoliv pouze `index.html`)

- tento soubor budete též odevzdávat

## Note

Ze zájmu můžete porovnat třídy `oks01.OsobniCislo` a `oks05.OsobniCislo`. Třída `oks01.OsobniCislo` obsahovala několik logických chyb, na které jsem přišel až důkladným testováním pomocí strukturálních testů. Je to poměrně přesvědčivý důkaz o jejich užitečnosti.

Kontrola úplnosti řešení:

- adresář `test` z IntelliJ překopírujte někam do svého pomocného adresáře, např. `D:\zzz`
  - dále sem překopírujte obsah adresáře `kontrola/oks05` 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 `;`;

```
-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 ...
```

- vypíše se např.:

```
Vysledek vseh testu: true<br />
Pocet spustenych testu: 22<br />
Pocet testu, ktere selhaly: 0<br />
Pokryti: [OsobniCislo,100%,100%,0,40,0,99,0,19,0,1]<br />
Uplne pokryti: true<br />
```

- pokud jste použili vnitřní analyzátor IntelliJ (a nikoliv JaCoCo), má soubor `index.html` nesprávný obsah a vypíše se např.:

```
Vysledek vseh testu: true<br />
Pocet spustenych testu: 22<br />
Pocet testu, ktere selhaly: 0<br />
CHYBA: Nebyl použit analyzátor JaCoCo
Uplne pokryti: false<br />
```

- při malém pokrytí se vypíše např.:

```
Vysledek vseh testu: true<br />
Pocet spustenych testu: 14<br />
Pocet testu, ktere selhaly: 0<br />
Pokryti: [OsobniCislo,94%,95%,3,40,5,99,2,19,0,1]<br />
Uplne pokryti: false<br />
```

- v tomto případě se vraťte do IntelliJ a pište další testy

- při selhání testů se vypíše např.:

```
Vysledek vseh testu: false<br />
Pocet spustenych testu: 22<br />
Pocet testu, ktere selhaly: 1<br />
Pokryti: [OsobniCislo,100%,100%,0,40,0,99,0,19,0,1]<br />
Uplne pokryti: true<br />
```

- v tomto případě se vraťte do IntelliJ a opravujte existující testy

Příprava souborů k odevzdání:

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

```
jar cMf oks-05.jar test/oks05 index.html
```

- výsledkem bude soubor `oks-05.jar`, který budete odevzdávat

- opravdu není spustitelný, neobsahuje `.class` soubory ani Javadoc
- jeho velikost by měla být do 5 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