

Dokumentace kódu.

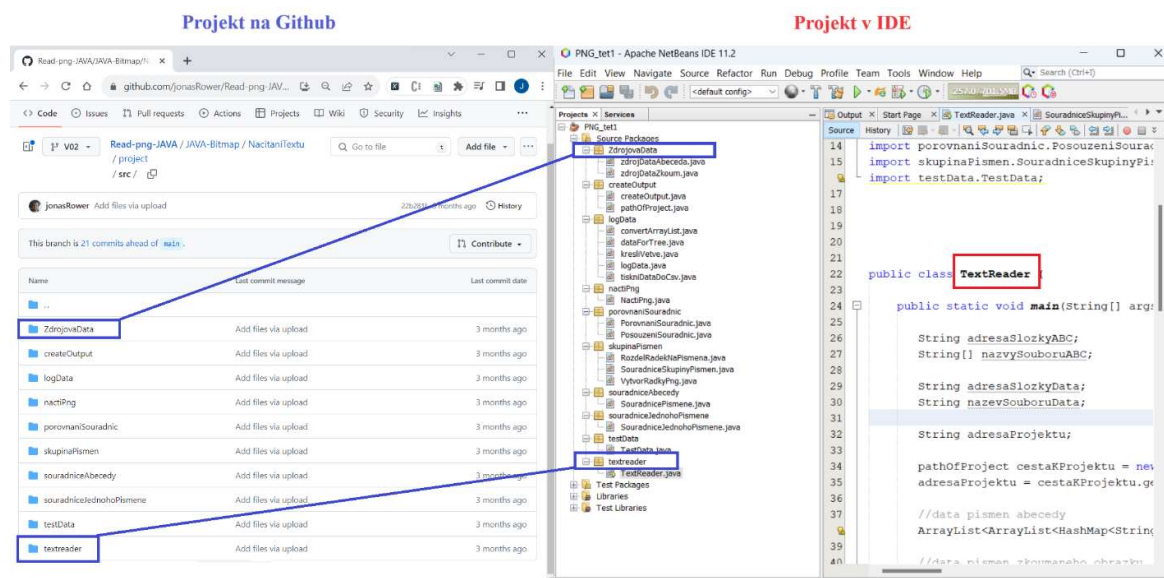
Princip zanoření první metody kódu JAVA.

1.	Data testovaného projektu.....	2
2.	Před spuštěním projektu.....	2
3.	Vyhledání první zanořené metody.....	3
4.	Vyhledání indexů od-do pro vyjmutí dílčího kódu.....	9
5.	Vkládání vnořených větví kódu.....	12

1. Data testovaného projektu.

Jedná se o zdrojový kód funkčního projektu, který je zde:

<https://github.com/jonasRower/Read-png-JAVA/tree/V02/JAVA-Bitmap/NacitaniTextu/project/src>



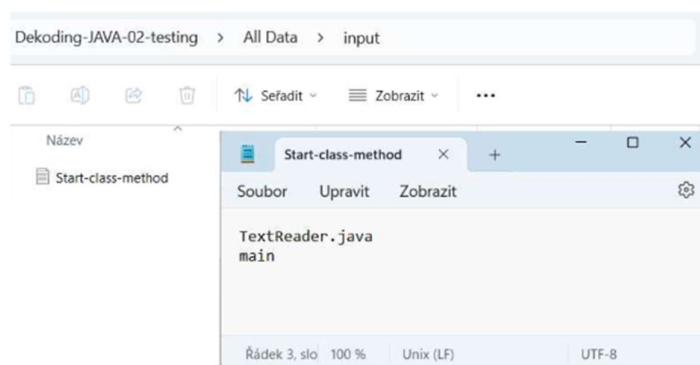
Obr. 1 – Testovací data projektu.

2. Před spuštěním projektu.

Jak uvádí dokumentace

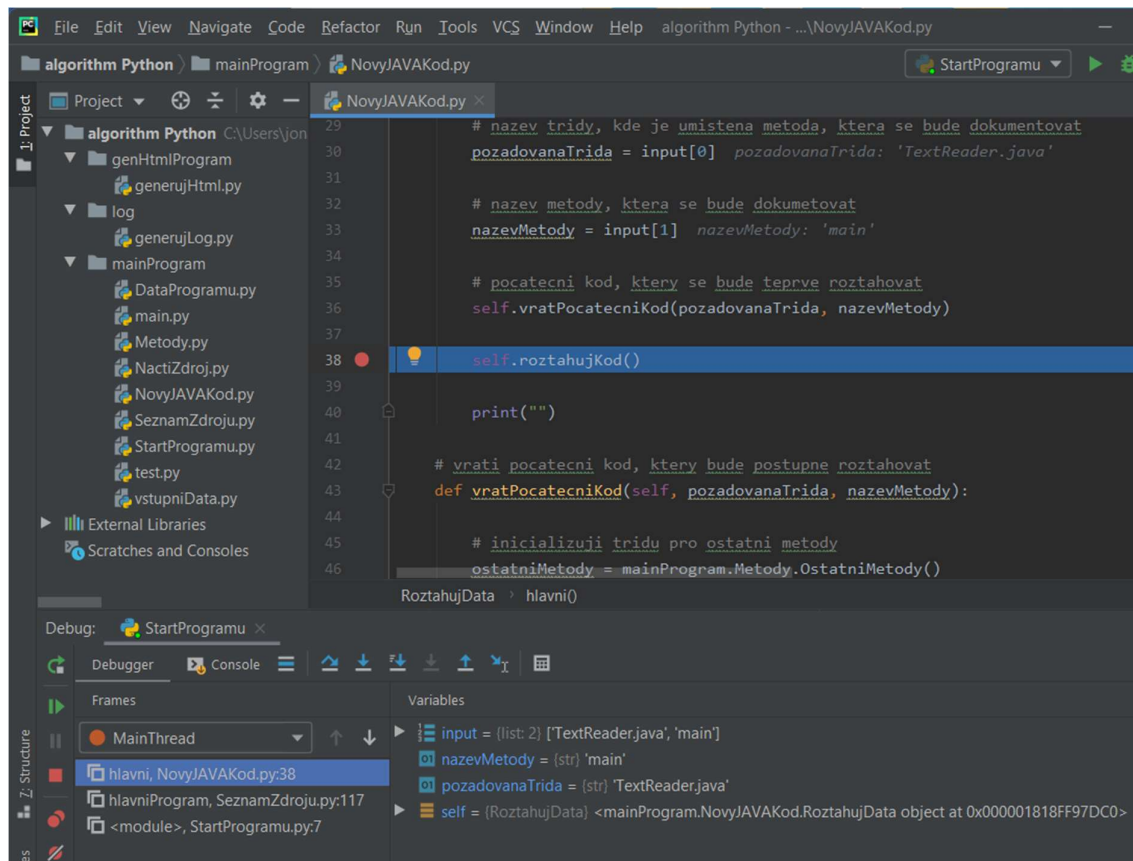
<https://github.com/jonasRower/Dekoding-JAVA/blob/V02-testing/V02-coJeNoveho.pdf>

je potřeba nastavit výchozí název metody a třídy odkud kód poběží:



Obr. 2 – Vstupní data projektu.

Kód zastavíme na breakpointu v souboru NovyJAVAKod.py:



Obr. 3 – Načtení vstupních dat do proměnných.

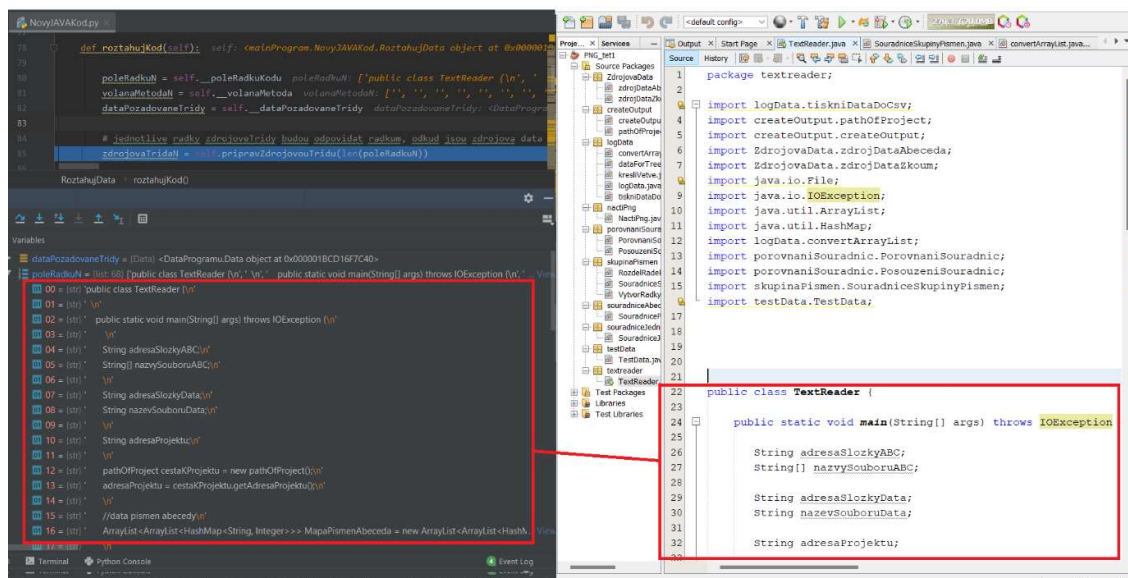
`nazevMetody` a `pozadovanaTrida` odpovídají vstupům, obr. 2.

3. Vyhledání první zanořené metody.

V tomto odstavci si popíšeme jak funguje kód pro získání zanořené dílčího kódu metody.

Pomocí klávesy F7 krojujeme a zastavíme kód zde, v třídě `roztahujKod` (NovyJAVAKod, ř. 85).

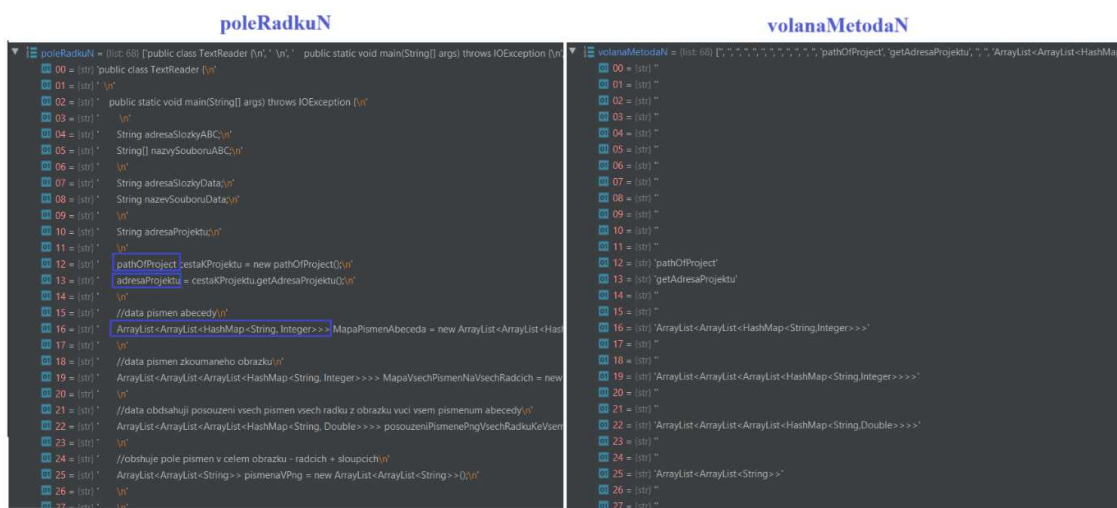
Rozvíjení zdrojového kódu do stromové struktury



Obr. 4 – Objekt **poleRadkuN**.

Zde **poleRadkuN** odpovídá zdrojovému kódu v JAVA, bez řádků import.

Následně data **volanaMetodaN** obsahuje jen ty řádky obsahující volané metody, ostatní řádky jsou prázdné.

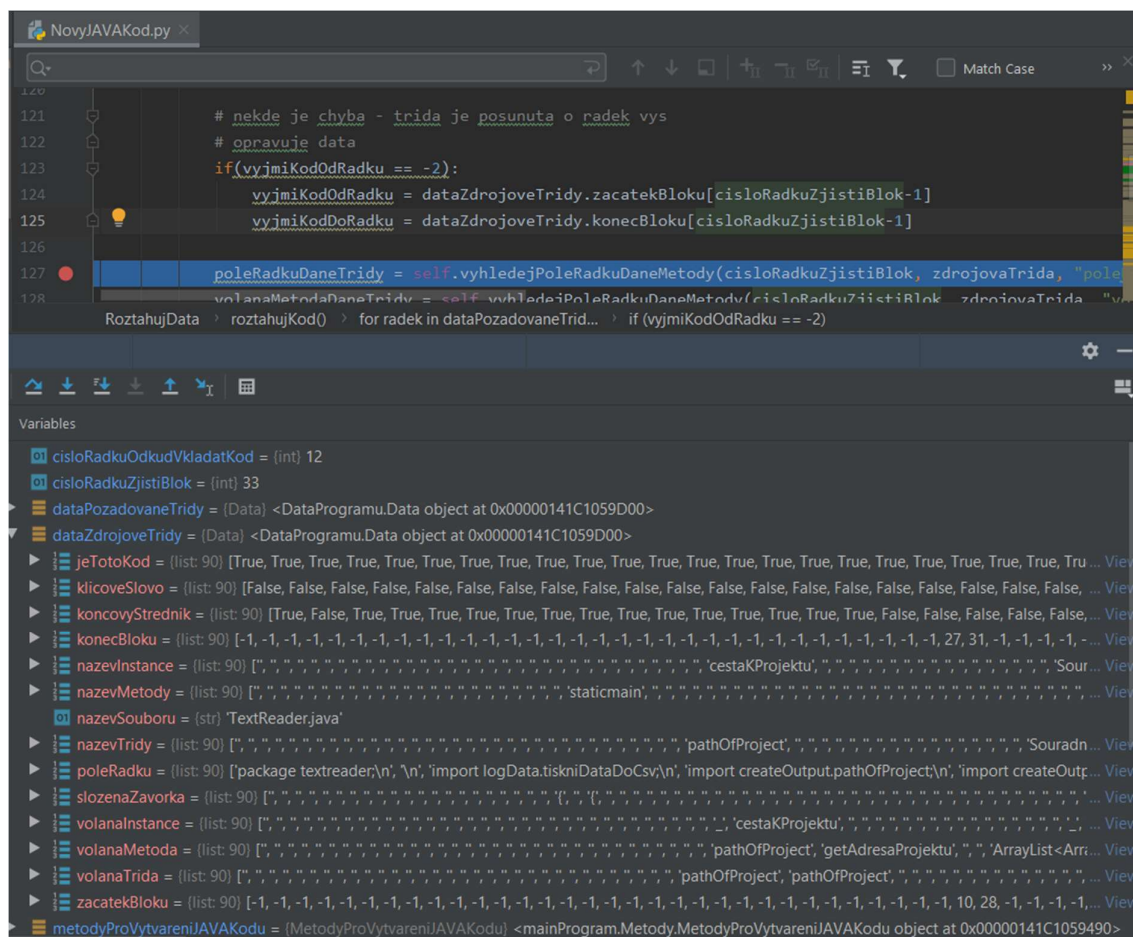


Obr. 5 – Srovnání dat **poleRadkuN** a **volanaMetodaN**.

Budeme tedy vyhledávat metodu **getAdresaProjektu()** ve třídě **pathOfProject()**.

Zastavme kód na řádku 127 v NovyJAVAKod.py.

Rozvíjení zdrojového kódu do stromové struktury



Obr. 6 – **dataZdrojoveTridy** obsahují všechny data kódu JAVY (dané třídy).

Můžeme si všimnout:

cisloRadkuOdkudVkladatKod = 12.

Jedná se o 1. řádek jiný než prázdný na obr. 5 – vpravo.

Logicky:

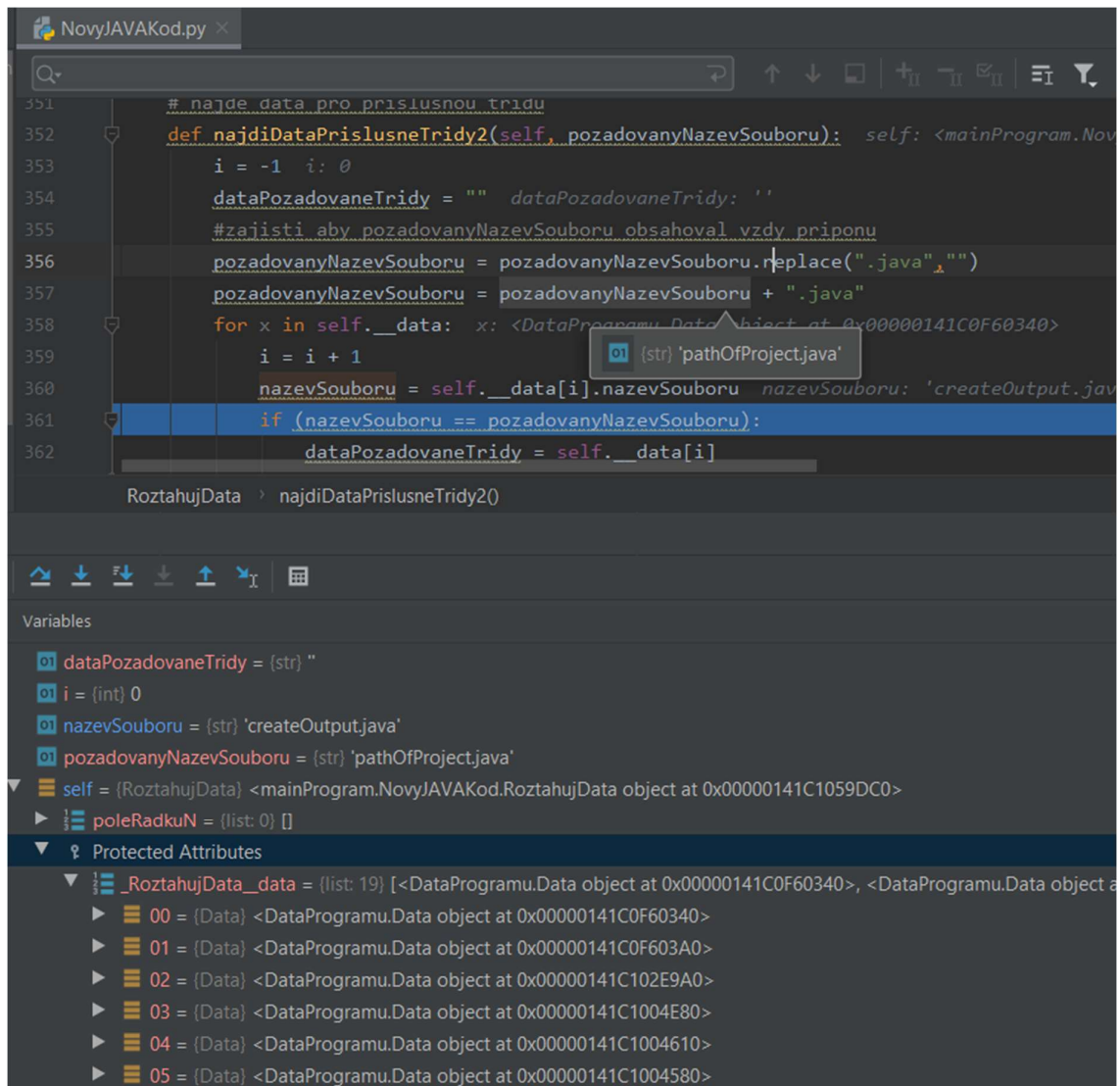
```
VolanaMetodaN[12] = pathOfObject
```

dataZdrojoveTridy jsme si již popisovali v dokumentaci zde:

<https://github.com/jonasRower/Dekoding-JAVA/blob/V02/kodDokumentace.pdf>

Obr. 8 – Obr. 19.

Následně pomocí klávesy F7 krokujeme, na řádek 361.

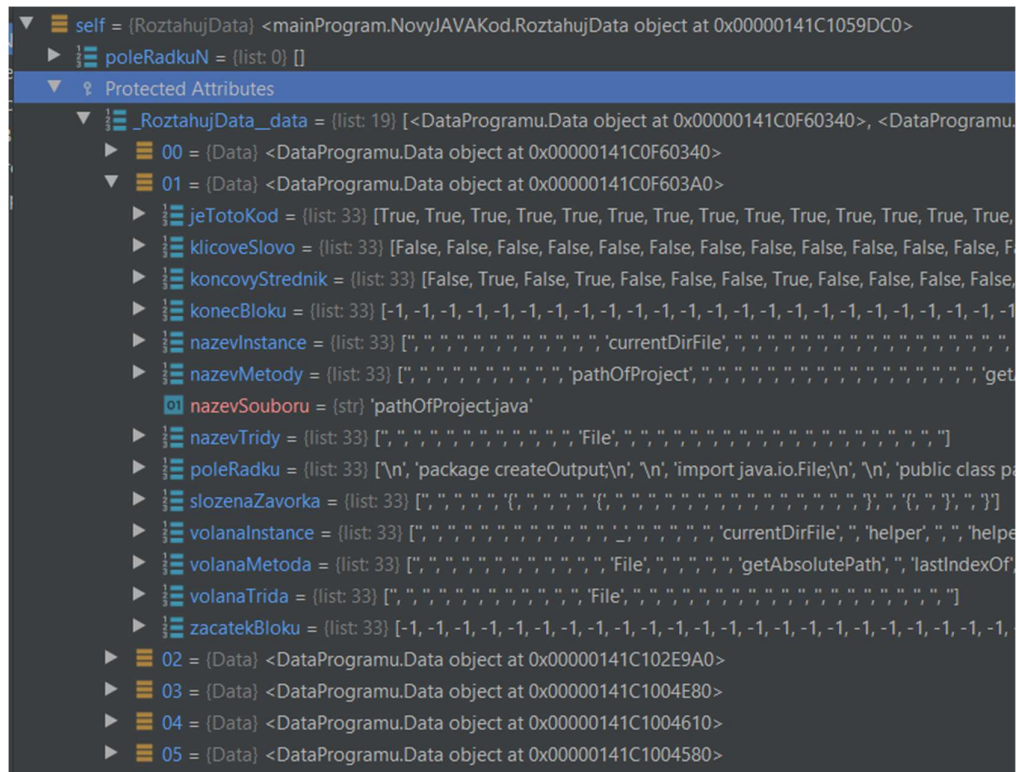


Obr. 7 – `_RoztahujData__data` obsahuje 19 pod-objektů. Každý pod-objekt obsahuje data jednoho java-souboru.

Též uvedeno v dokumentaci (Obr. 6).

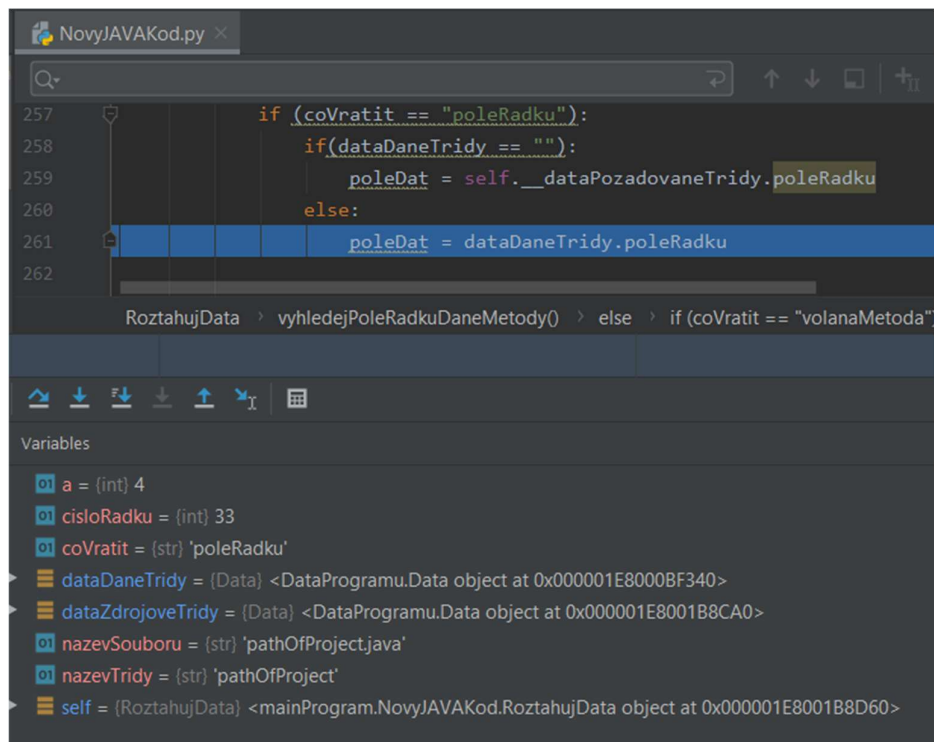
<https://github.com/jonasRower/Dekoding-JAVA/blob/V02/kodDokumentace.pdf>

`self.__data` obsahují jednotlivá data všech souborů .java. Je tedy třeba vyhledat správný soubor dat, dle `pozadovanyNazevSouboru`. Nalezneme jej s indexem `i = 1`.



Obr. 8 – Potřebnou sadu dat vyhledáváme přes název souboru.

Očekáváme návrat **poleRadku**, tudíž vložíme jej do proměnné **poleDat**.



Obr. 9 – Získání návratových dat.

Rozvíjení zdrojového kódu do stromové struktury


[illegible]

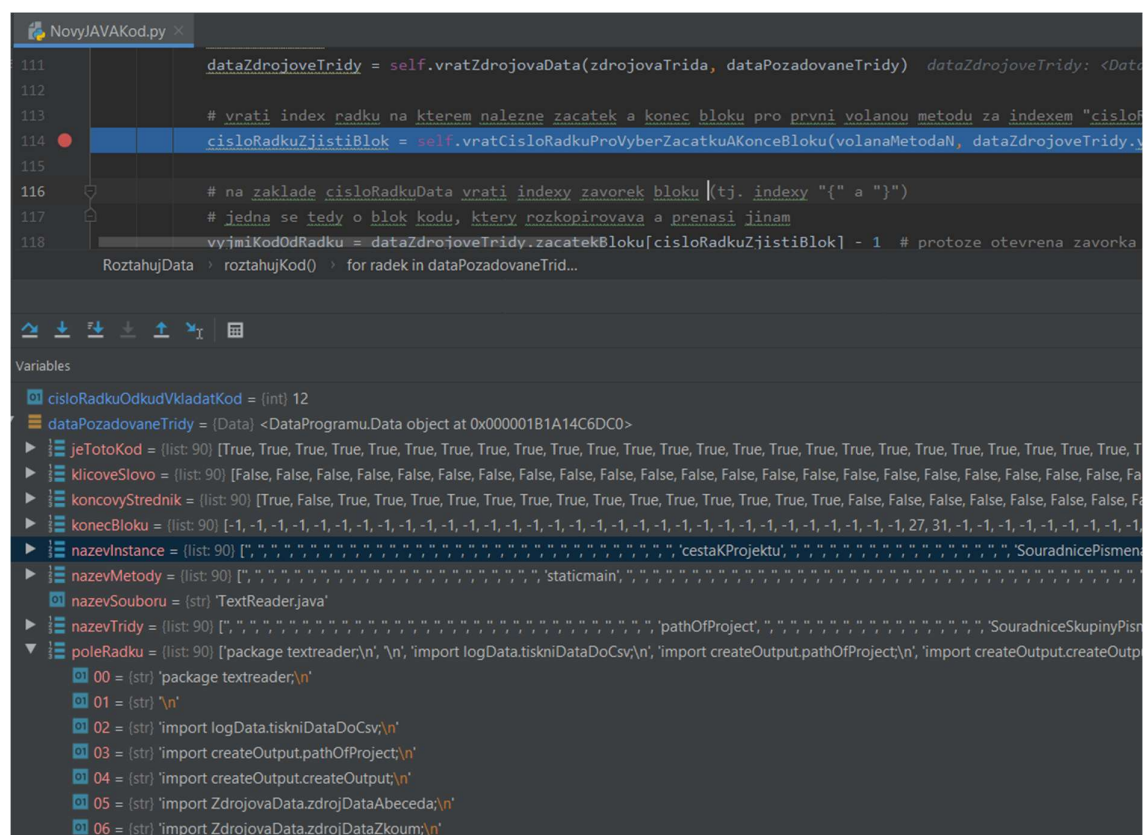
Obr. 10 – Data třídy `PathOfProject()`.
Z této třídy je třeba vyjmout požadovanou metodu.

Z obr. 5 je patrné, že očekáváme metodu pro vložení – **getAdresaProjektu()**. Požadované hodnoty jsme našli v objektu **poleRadkuDaneTridy**, na řádcích 28 – 30 (viz obr. 10). Nyní je třeba rozmezí mezi požadovanými řádky vyjmout a vložit do **poleRadkuN**.

4. Vyhledání indexů od-do pro vyjmutí dílčího kódu

Na obrázku výše jsme též zachytili proměnné **vyjmiKodOdRadku** a **vyjmiKodDoRadku**. Předpokládáme, že jejich význam je již zřejmý.

Zastavme tedy debug  a spustíme debug znovu. Zastavíme kód na řádku 114, NopvyJAVAKod.py.



Obr. 11 – Zastavme kód na ř. 114, abychom si mohli popsat parametry metody.

Zde voláme metodu:

```
cisloRadkuZjistiBlok = self.vratCisloRadkuProVyberZacatkuAKonceBlok(
    volanaMetodaN, dataZdrojoveTridy.volanaMetoda, cisloRadkuOdkudVKladatKod,
    dataPozadovaneTridy.nazevInstance)
```

kde parametry metody jsou:

volanaMetodaN – obsah dat jsme si již ukázali na obr. 5, vpravo.

dataZdrojoveTridy.volanaMetoda – obsah pole je stejné jako volanaMetodaN, pouze s rozdílem, že pole je posunuté. Rozdíl v indexaci řádků je způsoben rozdílným uvažováním začátku pole. Počet řádků, o které je pole posunuté je dáno počtem zanedbaných řádků, před názvem třídy.

28 - 7 = 21

dataZdrojoveTridy.volanaMetoda

volanaMetodaN

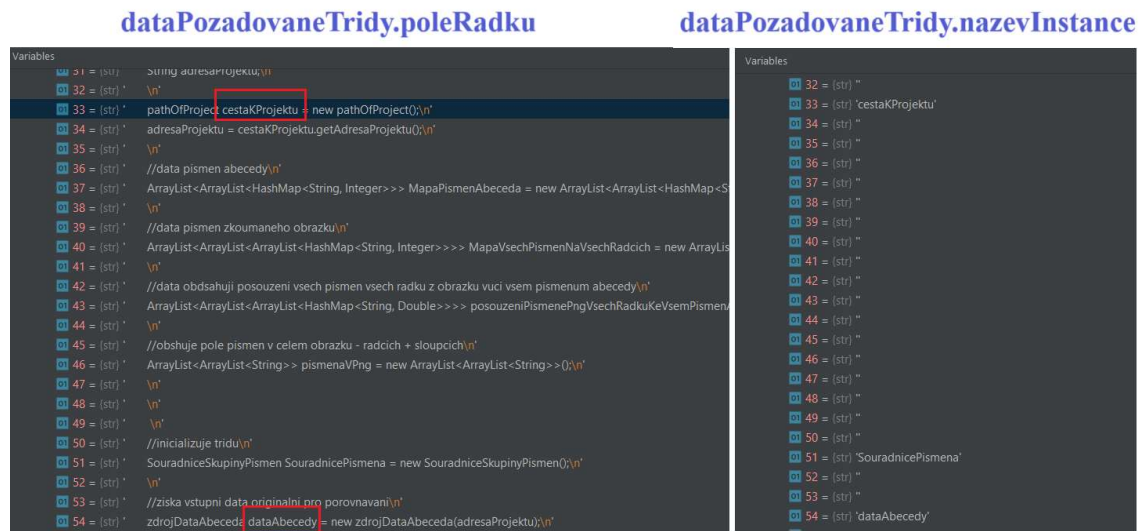
Obr. 12 – Porovnání dat :
dataZdrojoveTridy.volanaMetoda a **volanaMetodaN**

cisloRadkuOdkudVkladatKod = 12 – Jedná se o 1. řádek jiný než prázdný na obr. 5, vpravo.

dataPozadovaneTridy.nazevInstance – srovnej s **dataPozadovaneTridy.poleRadku**.

dataPozadovaneTridy.poleRadku jsou shodná s **poleRadkuN** (viz obr. 4), posunutá stejným mechanismem, jako je mezi **dataZdrojoveTridy.volanaMetoda** a **volanaMetodaN**.

Metoda **self.vratCisloRadkuProVyberZacatkuAKonceBloku** vrátí 33, jelikož hodnota „**pathOfProject**“ náleží v **dataZdrojoveTridy.volanaMetoda** právě indexu 33.



Obr. 13 – Porovnání dat:
dataPozadovaneTridy.poleRadku a dataPozadovaneTridy.nazevInstance

Posléze již jednoduše dohledat, že

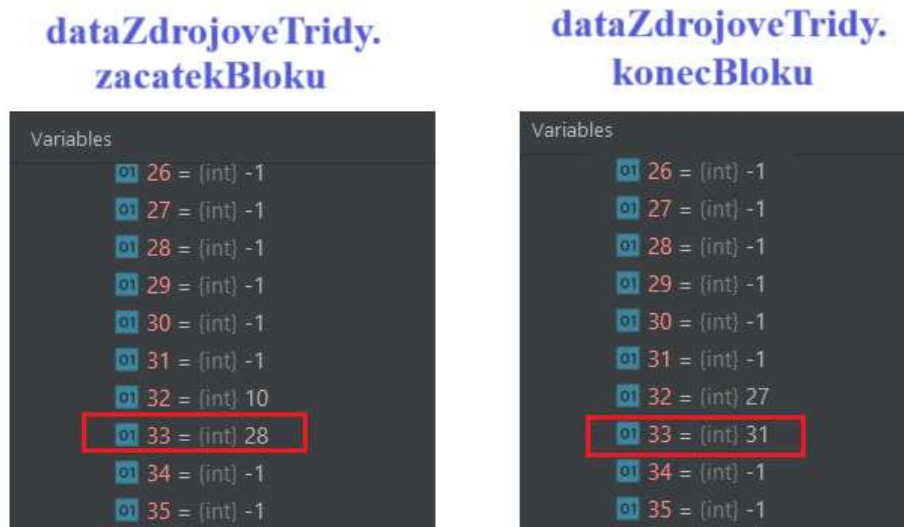
vyjmiKodOdRadku = 27 a

vyjmiKodDoRadku = 31.

Dané hodnoty dohledáme v poli:

dataZdrojoveTridy.zacatekBloku a

dataZdrojoveTridy.konecBloku.



Obr. 13 – Dohledání hodnot:
vyjmiKodOdRadku a vyjmiKodDoRadku

5. Vkládání vnořených větví kódu.

Vkládání větví zanořeného kódu se provádí v

```
poleRadkuN = metodyProVytvoreniJAVAKodu.vlozSubDataProJednotlivuPole(
    poleRadkuN, True, vyjmiKodOdRadku, vyjmiKodDoRadku, cisloRadkuOdkudVkladatKod,
    poleRadkuDaneTridy, "")
```

kde jednotlivé objekty jsme si již popsali výše.

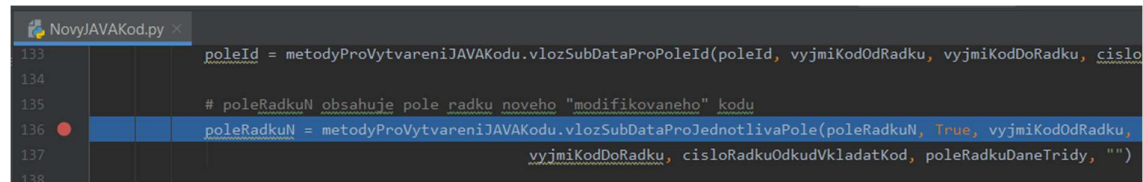
poleRadkuN – obr. 4.

vyjmiKodOdRadku = 27

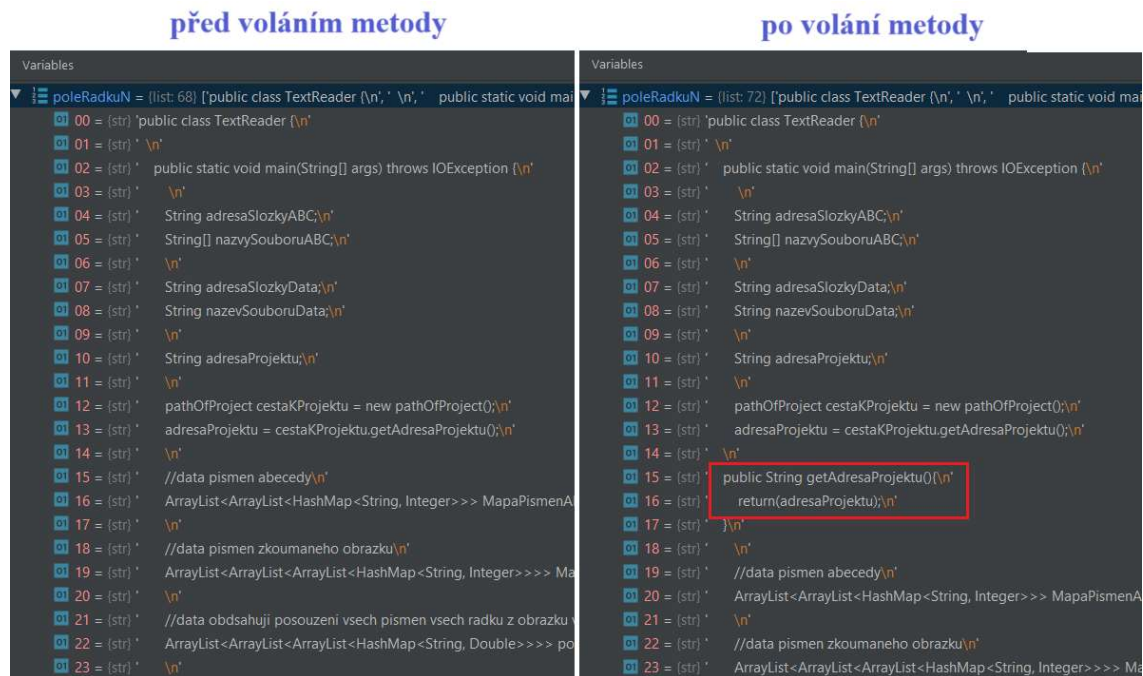
vyjmiKodDoRadku = 31

cisloRadkuOdkudVkladatKod = 12

poleRadkuDaneTridy – obr. 10



Obr. 14 – Metoda zajišťující vkládání vnořené větve kódu:



Obr. 14 – Rozdíl před a po volání metody – dle obr. 14.