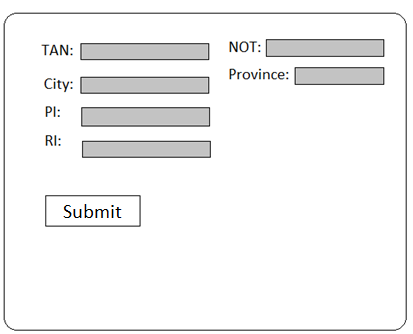
**Coding Exercise 1**



**Handle Return Data**

1. Retrieve details of set of returns
   * Option 1 – Read return data from the csv file (sampledata.csv)
   * Option 2 – Manually collect return data (see step 2)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **TAN** | **NOT** | **City** | **Province** | **PI** | **RI** |
| T53967724 | Wbsemgvwtv | Banff | AL | 428034 | 306450 |
| T35700020 | Rnskmvcarj | Laval | QC | 64347 | 673588 |
| T94067732 | Rfqjdihobe | Kamloops | BC | 482274 | 130247 |
| T92374709 | Kanbfibfnn | Lethbridge | AL | 711949 | 964297 |
| T27381278 | Umzhnmoicx | Dorval | QC | 659018 | 208771 |

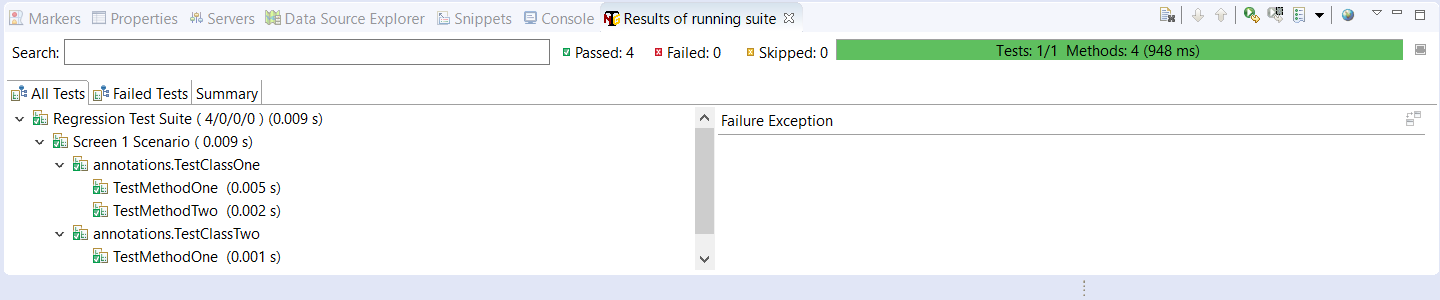
1. Add returns to collection of your choice (ArrayList, LinkedList, Map)
   * Which implementation is better for inserting a return later, searching for a return?
     + Think in context of millions of returns
2. Display return data (in console is ok)

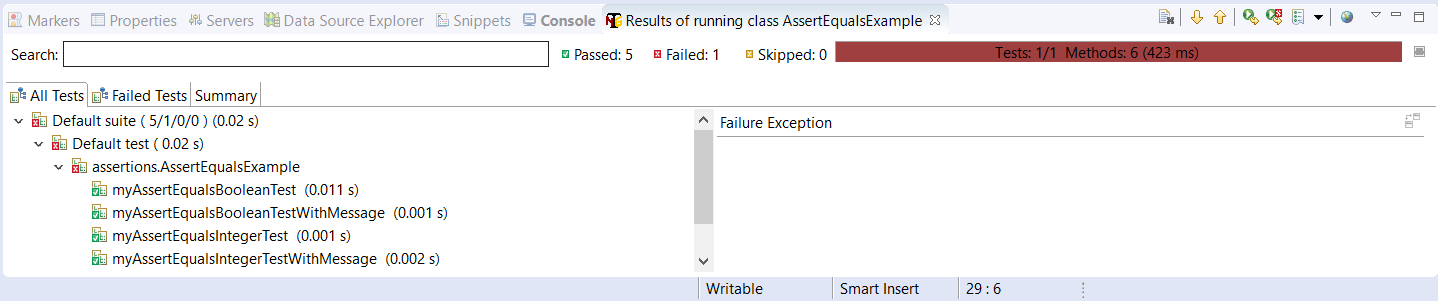
**Return Rejected (no data is being stored, only tested)**

1. If any of the amount fields are negative
   * Throw an exception (console)
2. Manually enter return data into interface
   * If a return is missing data in any field ‘Reject’ the return and display a ‘Rejected Return’ message to the user.

**Method:**

* Test Driven Development
  + Minimum 5 assertions (your choice), one that fails
    - assertTrue()
    - assertFalse()
    - assertEquals()
    - assertEquals()
    - assertNull()





**TODO**

1. Install TestNG
2. Add necessary imports

**import** org.testng.Assert;

**import** org.testng.annotations.Test;

1. Add TestNG library to build path
2. Pair Programming for at least one subtask per Sprint (suggestion 🡪 Collection implementation)
   * Pick a developer take have them observe as you implement the code/feature
   * Developer should be looking for ways to prevent bugs
   * Discussion/feedback
   * Change roles (coder – observer)
3. Peer/Code Review
   * At least two developers should review your code (provide sign-off)
   * Reviewers should be looking for code quality issues
     + Some examples:
       - Overly complicated methods
       - Empty statements
       - Dead code
       - Duplicated code
       - Passing arguments vs using static variables
   * Request a review
     + In T3 a review should be requested after about a days work or 40 lines of code, but for training purposes request a review after about 3 to 5 days of work. So once or twice per Sprint.

**Definition of Done**

* Peer Reviewed
* Unit Tested (No current code coverage percentage – Note in T3 we aim for 70%)

**MVP (Minimum Viable Product)**

* Display message for a rejected return
* Return data displayed

Bonus Work:

* Handle test case where duplicate Trust numbers exist (in reality this cannot occur)
* Save Return data entered from GUI
* Experiment collecting return data using different data structures and use generics to handle **T3** Return object type
  + Example Map <String, String>

Map<String, T3Return>