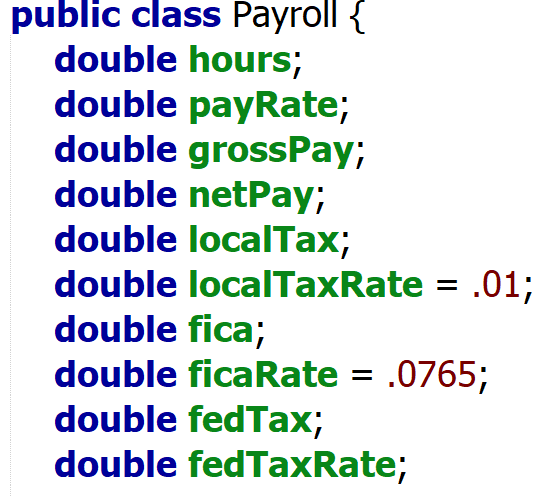
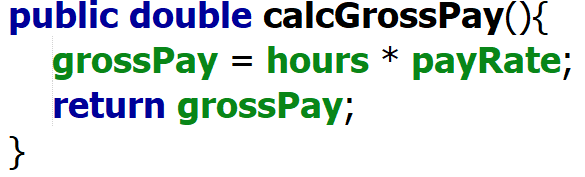
# Directions

1. Complete the following steps.
2. Screenshot where directed.
3. Submit to Blackboard.

# Scenario

You are writing a class to calculate payroll for an hourly employee. Your class will store and calculate the following:



* Create Setters and Getters for hours and payrate
* Create the default constructor that accepts no parameters
* Create the parameterized constructor that accepts hours and payrate.
* You will create methods for the remaining variables that will set the class variable and return the value. I have provided a partial example with calcGrossPay().  
  

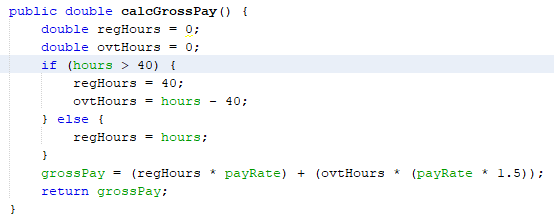
You are going to follow TDD principles to build this class. This document will step you through the process.

# grossPay

Your gross pay function will calculate gross pay. It must also include overtime hours. Sample data:

|  |  |  |
| --- | --- | --- |
| Hours | Pay Rate | Gross Pay |
| 40 | 10 | 400 |
| 45 | 10 | 475 |

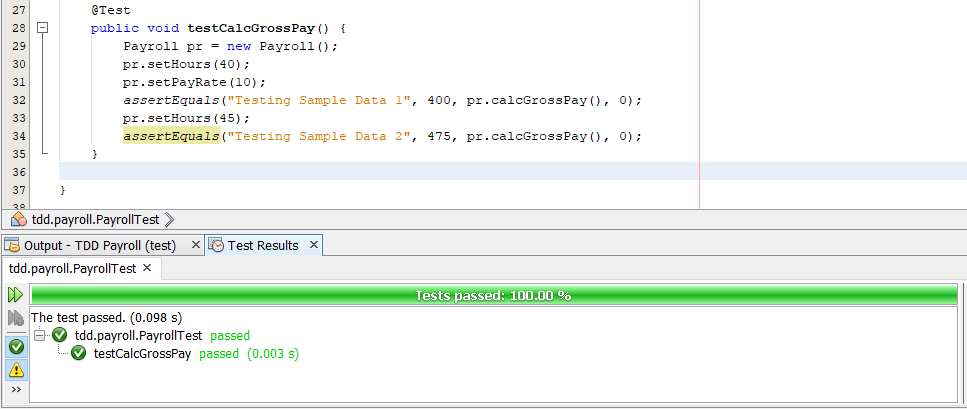
Overtime is calculated as any 40 + (hours > 40) \* 1.5 \* payrate. Example for 45 hours, you get 40 hours at 10 per hour and 5 at 10\*1.5 per hour.



***This was a fun assignment. I enjoyed completing it! A very interesting way to show the testing process.***

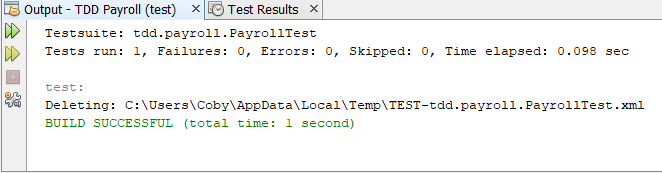
## Build the test(s)

Create JUnit tests that check for your two pieces of sample data. Screenshot your tests



## Run the test(s)

Screenshot the outputs of running your test.



## Refactor the code

Correct the calcGrossPay function until it passes the test. Screenshot your new calcGrossPay code and the output of your JUnit tests.

**It worked the first time. I will not snip the working code again. I will only populate this section if it originally fails.**

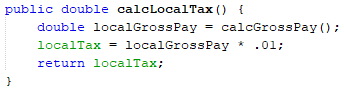
# localTax

We have set local tax to 1%. Your method must calculate the local tax. To calculate local tax, multiply the gross pay by the .01. Here is sample data

|  |  |  |
| --- | --- | --- |
| Hours | Pay Rate | Local Tax |
| 40 | 10 | 4 |

## Build the test(s)

Create JUnit tests that check for your sample data. Your calculations can be within 1 penny. (Hint, delta) Screenshot your tests.



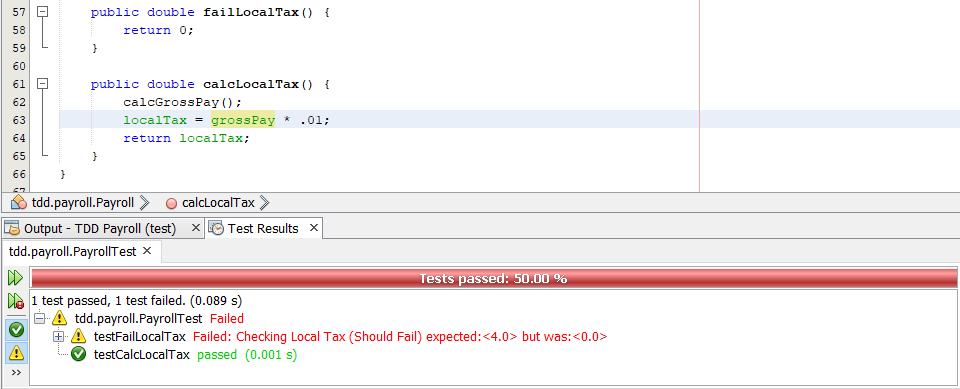
## Write a function stub

Write a simple function for calcLocalTax() that returns 0;



## Run the test(s)

Screenshot the outputs of running your test. This first run should fail.



## Refactor the code

Correct the calcLocalTax() function until it passes test. Screenshot your new calcLocalTax() code and the output of your JUnit tests.

**It worked the first time. I will not snip the working code again. I will only populate this section if it originally fails.**

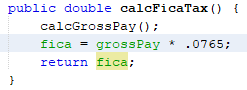
# ficaTax

FICA tax is your Social Security and Medicare tax. SS tax is 6.2% while Medicare tax is 1.45. Combined, the FICA tax is 7.65%. To calculate fica tax, multiply the gross pay by the .0765. Here is sample data

|  |  |  |
| --- | --- | --- |
| Hours | Pay Rate | FICA Tax |
| 40 | 10 | 30.60 |

## Build the test(s)

Create JUnit tests that check for your sample data. Your calculations can be within 1 penny. (Hint, delta) Screenshot your tests.



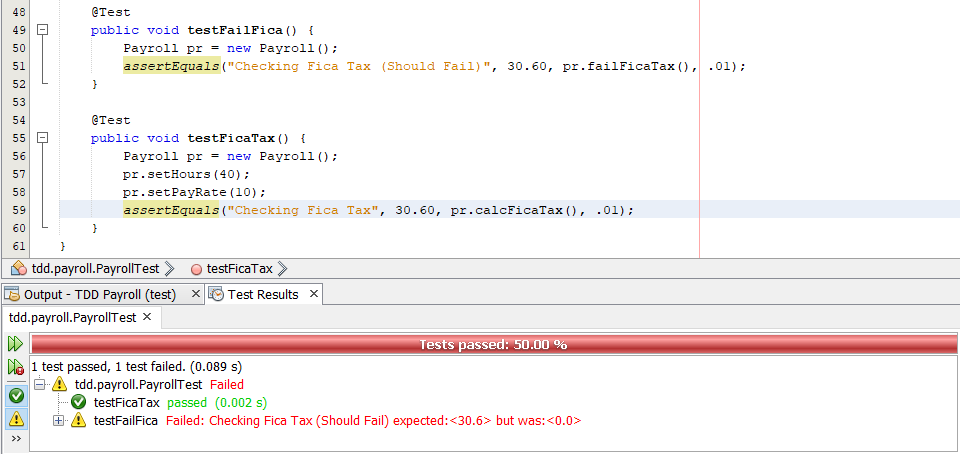
## Write a function stub

Write a simple function for calcFicaTax() that returns 0;



## Run the test(s)

Screenshot the outputs of running your test. The first run will fail.



## Refactor the code

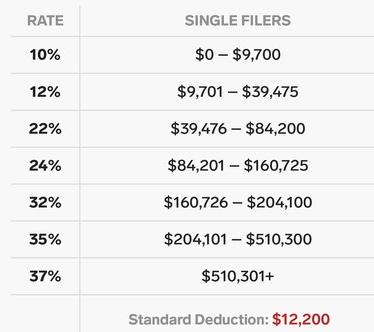
Correct the calcFicaTax() function until it passes test. Screenshot your new calcFicaTax() code and the output of your JUnit tests.

**It worked the first time. I will not snip the working code again. I will only populate this section if it originally fails.**

# fedTax[[1]](#footnote-1)

Fed tax is complicated. It is based on yearly income. Assume that your employees work 52 weeks in a year. Since the data you have is one week, you multiply the payroll by 52. You then subtract your standard deduction of 12,200. Your formula is

Now, we need to determine the FED Tax rate. Use this table.[[2]](#footnote-2) Build an if or switch structure to determine

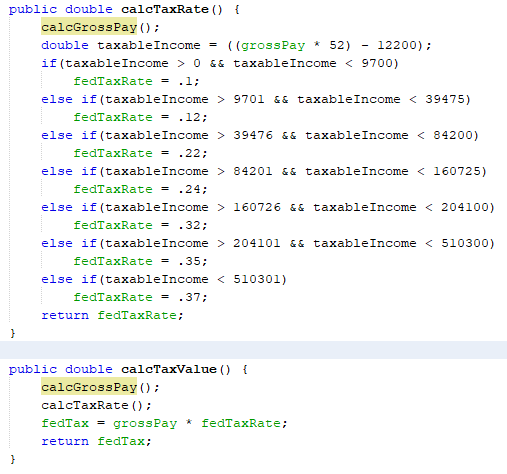


Now that you have the fed tax rate, you calculate fed tax, multiply the gross pay by the RATE. Here is sample data. Notice that you have a second row of data again.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Hours | Pay Rate | Taxable Income | FED Tax Rate | FED Tax Value |
| 40 | 10 | 8,600 | 10% | 40 |
| 40 | 50 | 91,800 | 24% | 480 |

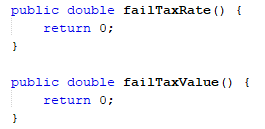
## Build the test(s)

Create JUnit tests that check for Fed Tax Rate and Fed Tax Value. Your calculations can be within 1 penny. (Hint, delta) Screenshot your tests. You would write a test that checked each value of the FED Tax Rate.



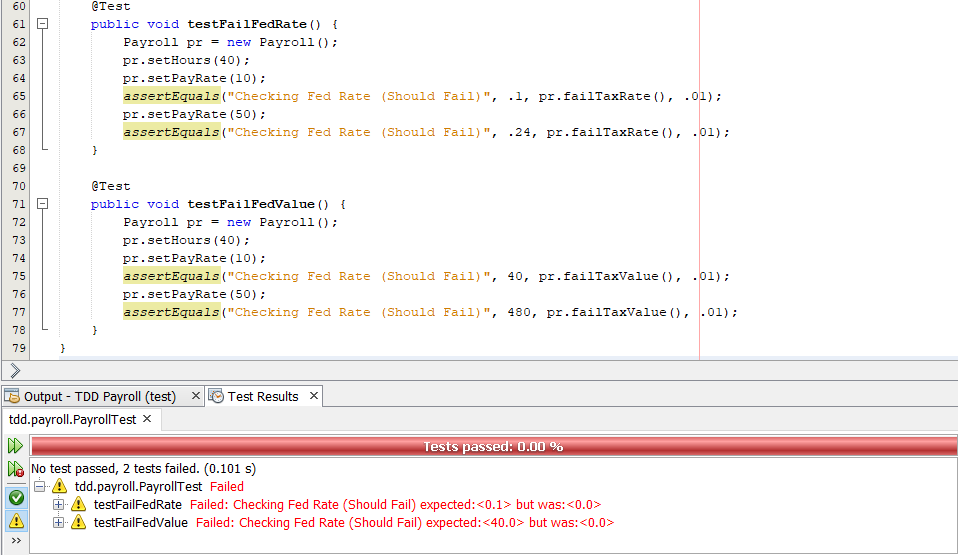
## Write a function stubs

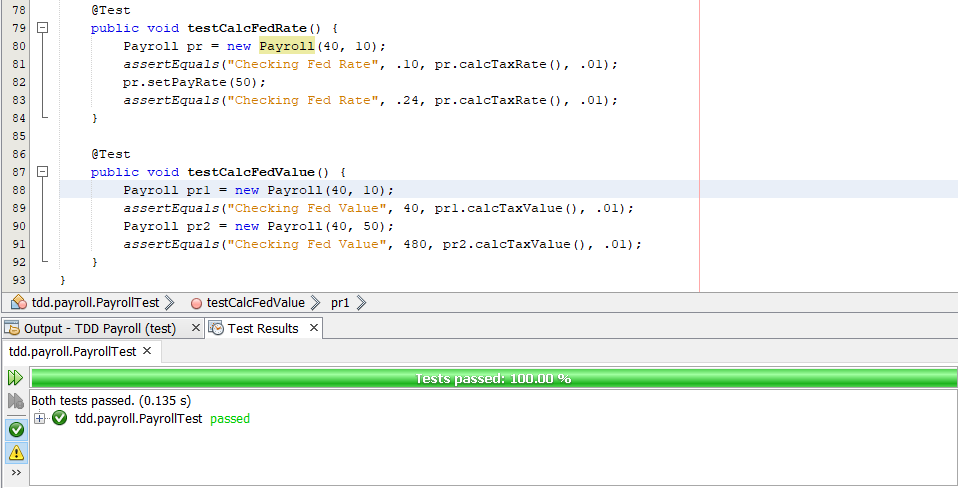
Write simple functions for calcFedTax() and calcFedTaxRate() that return 0;



## Run the test(s)

Screenshot the outputs of running your test. The first run will fail. **I separated the tests to see easier.**





## Refactor the code

Correct the calcFedTax() and calcFedTaxRate() functions until they passes their tests. Screenshot your new calcFicaTax() code and the output of your JUnit tests.

**It worked the first time. I will not snip the working code again. I will only populate this section if it originally fails.**

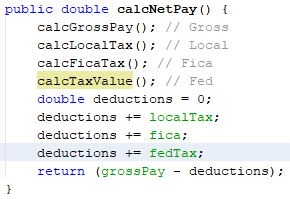
# netPay

Net pay is your gross pay – all deductions and taxes. Only one piece of sample data.

|  |  |  |
| --- | --- | --- |
| Hours | Pay Rate | Net Pay |
| 40 | 10 | 325.40 |

## Build the test(s)

Create JUnit tests that check for your sample data. Your calculations can be within 1 penny. (Hint, delta) Screenshot your tests.



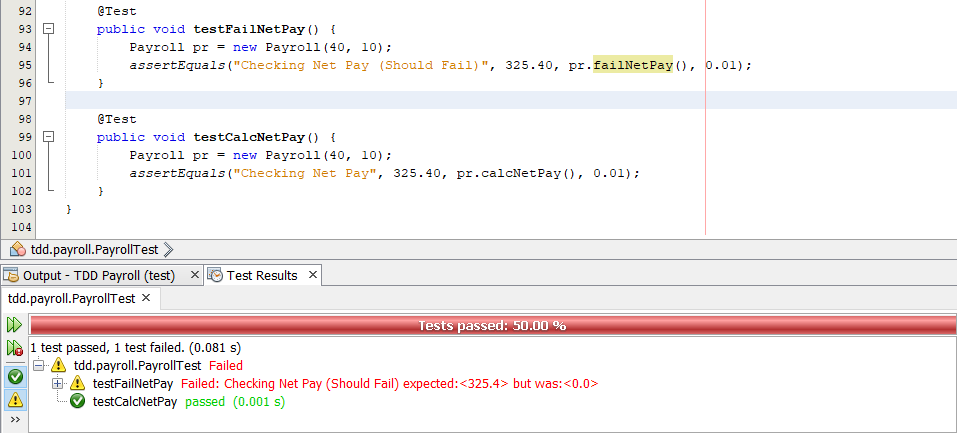
## Write a function stub

Write a simple function for calcNetPay() that returns 0;



## Run the test(s)

Screenshot the outputs of running your test. The first run will fail.



## Refactor the code

Correct the calcFicaTax() function until it passes test. Screenshot your new calcFicaTax() code and the output of your JUnit tests.

**It worked the first time. I will not snip the working code again. I will only populate this section if it originally fails.**

1. Fed tax here is assuming that you are single. I simplified the calculation a little. [↑](#footnote-ref-1)
2. <https://jellisoncpa.com/whats-new-2019-taxes-tax-brackets/> [↑](#footnote-ref-2)