|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **LEARNING PROFILE FOR ASSIGNMENT#\_1\_\_\_\_\_ AND QUESTION#\_1\_\_\_\_\_\_\_** | | | | | |
| *Name* | *:* | *Steven Morrissey* | *Due Date* | *:* |  |
| *Student ID* | *:* | *3300222* | *Submission Date* | *:* |  |

**1. Problem Statement:**

Create a class named AddressBook that has the following field names: firstName, middleName, lastName, homeAddress, businessPhone, homePhone, cellphone, skypeId, facebookId, and personalWebSite. Use appropriate data types to store the values for these fields in AddressBook objects. Create appropriate get and set methods to retrieve and assign values to these names. For example, getMiddleName(viveAddressBook) should return the middle name of the person Vive. Similarly, vive.setPersonalWebsite(url) should set the personal website of the person Vive to the specified URL object. Using the get and set methods, create a comparison method compareNames(name1, name2) that compares the first, middle, and last names of strings name1 and name2

**2. Description of the Code:**

Besides creating all the required and standard fields and getters and setters, the compareNames() function was written to check if the name Strings passed as parameters are equal.

**3. Errors and Warnings:**

Table 1: List of Errors and Warnings Encountered in the Program

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Errors / Warnings** | **Details** | **How I solved them** |
| 1 | none |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 |  |  |  |

**4. Sample Input and Output:**

Case 1: Create two AddressBook objects with the same firstName, middleName and lastName fields, and call compareName method to verify they match

Case 2: Create two AddressBook objects with different firstName, middleName and lastName fields, and call compareName method to verify they do not match.

**5. Discussion:**

The question did not make sense to me in the beginning, as it’s not how I would structure an “Address Book” class myself (returning String for compare names, having no data structure fields to hold data sets instead of single values, etc…). I searched the forum and found other people with similar confusion and gained some understanding from their responses. I also found discrepancies between the API and the question, as it looks like they are passing in a name for getMiddleName which isn’t specified in the API.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **LEARNING PROFILE FOR ASSIGNMENT#\_1\_\_\_\_\_ AND QUESTION#\_2\_\_\_\_\_\_\_** | | | | | |
| *Name* | *:* | *Steven Morrissey* | *Due Date* | *:* |  |
| *Student ID* | *:* | *3300222* | *Submission Date* | *:* |  |

**1. Problem Statement:**

Space Inc. will give a quarterly and annual bonus to its employees only if the savings of the quarter and/or the year are greater than or equal to quarterly minimum (monthly commitment x 3) and/or the annual minimum (monthly commitment x 12) amount, respectively. The quarterly bonus is 3% of eligible quarterly savings, and the annual bonus is 5% of annual savings if eligible. If the annual savings exceeds the committed amount by at least 25%, Space Inc. matches the additional savings (25% or above) as part of the annual bonus. I. An employee has committed to save $2000 per month. Her quarterly savings are as follows: Q1 – $5000, Q2 – $7000, Q3 – $4000, and Q4 – $8000. II. Another employee has committed to save $3000 per month. His quarterly savings are as follows: Q1 – $6000, Q2 – $9000, Q3 – $10000, and Q4 – $17000. Write a program to compute the total bonus amount earned by these two employees in the year.

**2. Description of the Code:**

Strategy was to create a list of all quarterlies, then iterate through to have less repeating code. Finally calculate annual bonus and add it to the running total to be returned at the end of the method.

**3. Errors and Warnings:**

Table 1: List of Errors and Warnings Encountered in the Program

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Errors / Warnings** | **Details** | **How I solved them** |
| 1 | E1 | Forgot the ‘>’ before the ‘=’ in one of the if statements | Found at compile time and place the ‘>’ in the logic |
| 2 | E2 | Used number formatter in the return statement of the method – NumberFormatExcetion. | Moved the formatter to the main method where it can return a String and I don’t have to parse a double |

**4. Sample Input and Output:**

**Case 1: create new object and call computeBonus(2000, 5000, 7000, 4000, 8000), expecting 1,650 as the printed out value of total bonus**

**Case 2: create new object and call computeBonus(2000, 6000, 9000, 10000, 17000), expecting 3,180 as the printed out value of total bonus**

**5. Discussion:**

None, it was straight forward.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **LEARNING PROFILE FOR ASSIGNMENT#\_1\_\_\_\_\_ AND QUESTION#\_3\_\_\_\_\_\_\_** | | | | | |
| *Name* | *:* | *Steven Morrissey* | *Due Date* | *:* |  |
| *Student ID* | *:* | *3300222* | *Submission Date* | *:* |  |

**1. Problem Statement:**

Write a program that prompts the user to enter two points (x1, y1) and (x2, y2). Calculate and display the distance between the two points using the formula below. Round the answer up to 2 decimal points. You can use Math.pow(a,0.5) to compute the square root of an expression. Math.pow() returns a double.

**2. Description of the Code:**

The code was straight forward. I implemented it exactly as the API specified: using the built in Java Math library I calculated the squares and square root to find the distance between two given points.

**3. Errors and Warnings:**

Table 1: List of Errors and Warnings Encountered in the Program

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Errors / Warnings** | **Details** | **How I solved them** |
| 1 | none |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 |  |  |  |

**4. Sample Input and Output:**

**Case 1: user enters X1 as-2, Y1 as -3, X2 as -4, Y2 as 4 and the result is 7.28 as specified in the API**

**5. Discussion:**

None, it was straight forward.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **LEARNING PROFILE FOR ASSIGNMENT#\_1\_\_\_\_\_ AND QUESTION#\_4\_\_\_\_\_\_\_** | | | | | |
| *Name* | *:* | *Steven Morrissey* | *Due Date* | *:* |  |
| *Student ID* | *:* | *3300222* | *Submission Date* | *:* |  |

**1. Problem Statement:**

Extend the AddressBook class from Problem 1 to store the additional data. Now, write a method to find the fastest runner. Print the name, address, and his/her time (in minutes) on three separate lines. Find the second fastest runner. Print the name, address, his/her time (in minutes), and the difference in time with the fastest runner. Compute the average time of completion taken by these runners. Finally, print the name and number of years participated for each runner if the runner’s time of completion is equal to or better than the average time of completion.

**2. Description of the Code:**

Used a static initialization for the list of runners, as for the purposes of this exercise the information will not be changing. For the getFastest and getSecondFastest methods, I used the efficiency of Java 8 method references combined with Arrays utility to sort the arrays and take the [0] and [1] indexes respectively for each method.

**3. Errors and Warnings:**

Table 1: List of Errors and Warnings Encountered in the Program

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Errors / Warnings** | **Details** | **How I solved them** |
| 1 | none |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 |  |  |  |

**4. Sample Input and Output:**

**Case 1: Given an Array of BanffMarathonRunner, the name, address and time are printed out on separate lines for fastest runner; name, address, time and difference with fastest is printed out for second fastest runner; average time of all runners is printed out; name and # of years is printed out for all runners faster than the average time.**

**5. Discussion:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **LEARNING PROFILE FOR ASSIGNMENT#\_1\_\_\_\_\_ AND QUESTION#\_5\_\_\_\_\_\_\_** | | | | | |
| *Name* | *:* | *Steven Morrissey* | *Due Date* | *:* |  |
| *Student ID* | *:* | *3300222* | *Submission Date* | *:* |  |

**1. Problem Statement:**

Solve the following problem using a program: Suppose you save $100 each month into a savings account with an annual interest rate of 5%. Thus, the monthly interest rate is 0.05/12 = 0.00417. After the first month, the value in the account becomes 100 \* (1 + 0.00417) = 100.417 After the second month, the value in the account becomes (100 + 100.417) \* (1 + 0.00417) = 201.252 And after the third month, the value in the account becomes (100 + 201.252) \* (1 + 0.00417) = 302.507 … and so on. Write a program that randomly generates monthly savings amounts for the 15 runners in Problem 4. Each monthly saving should be in the range of $100 to $800. Extend the AddressBook class to store the monthly savings generated by the random number generator. Then, display the final account value for each of the 15 runners.

**2. Description of the Code:**

Calculates interest and generated savings separately and saves them into their respective fields. getReport gets the account value and creates a string for each EmployeeSavings object to be printed out

**3. Errors and Warnings:**

Table 1: List of Errors and Warnings Encountered in the Program

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Errors / Warnings** | **Details** | **How I solved them** |
| 1 | none |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 |  |  |  |

**4. Sample Input and Output:**

**Case 1: Given an Array EmployeeSavings with generated savings and interests, when a report is printed out for that Array, then all the account values are displayed to the user.**

**5. Discussion:**

**None, it was straight forward**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **LEARNING PROFILE FOR ASSIGNMENT#\_1\_\_\_\_\_ AND QUESTION#\_6\_\_\_\_\_\_\_** | | | | | |
| *Name* | *:* | *Steven Morrissey* | *Due Date* | *:* |  |
| *Student ID* | *:* | *3300222* | *Submission Date* | *:* |  |

**1. Problem Statement:**

I. Given the right triangles described below, write a program to compute the lengths of the remaining sides using a program. a. a = 48 and c = 80

b. a = 84 and c = 91

II. Determine if the following triangles are right-angled triangles: a. a = 45, b = 55, and c = 75

b. a = 28, b = 45, and c = 53

**2. Description of the Code:**

Used Java Math library functions to do calculations according to Pythagorean theorem. Calculated the B sides for part 1 in the main method, and used the isRightTriangle() method to do the calculation for right triangle.

**3. Errors and Warnings:**

Table 1: List of Errors and Warnings Encountered in the Program

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Errors / Warnings** | **Details** | **How I solved them** |
| 1 | none |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 |  |  |  |

**4. Sample Input and Output:**

**Question 1.a) b = 64 Question 1.b) b = 35 Question 2.a) false Question 2.b) true**

**5. Discussion:**

**I had a bit of trouble deciding where to put the calculation logic, but as the getters should return the values for the respective fields, I concluded that it should be done in the main method. There were no set methods in the API as well, which also led me to believe that I should do the right triangle calculation in the isRightTriangle method.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **LEARNING PROFILE FOR ASSIGNMENT#\_1\_\_\_\_\_ AND QUESTION#\_7\_\_\_\_\_\_\_** | | | | | |
| *Name* | *:* | *Steven Morrissey* | *Due Date* | *:* |  |
| *Student ID* | *:* | *3300222* | *Submission Date* | *:* |  |

**1. Problem Statement:**

Write a program that reads in a number from the user and then displays the Hailstone sequence for that number, followed by a line showing the number of steps taken to reach 1.

**2. Description of the Code:**

Implemented hailstone sequence using arraylist to store the numbers and keeping track in a variable called total.

**3. Errors and Warnings:**

Table 1: List of Errors and Warnings Encountered in the Program

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Errors / Warnings** | **Details** | **How I solved them** |
| 1 | E1 | ArrayIndedOutOfBoundsException | I didn’t notice that I hadn’t assigned an initial value then tried to access size()-1. Solved itself as I changed my implementation. |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 |  |  |  |

**4. Sample Input and Output:**

**Case 1: Given an initial integer of X, the hailstone sequence and number of steps taken is printed out.**

**5. Discussion:**

**This was fairly straight forward, nothing noteworthy to discuss.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **LEARNING PROFILE FOR ASSIGNMENT#\_1\_\_\_\_\_ AND QUESTION#\_8\_\_\_\_\_\_\_** | | | | | |
| *Name* | *:* | *Steven Morrissey* | *Due Date* | *:* |  |
| *Student ID* | *:* | *3300222* | *Submission Date* | *:* |  |

**1. Problem Statement:**

Google Inc. is looking to recruit three of the Boston runners. The criteria for selection are as follows:

…

Store these values for the fifteen candidates in an extended AddressBook class. In general, Google will not consider a candidate with average marks of less than 85%. Google will consider a candidate with average marks of less than 85% only if the candidate at least has 0.5 regulatory abilities and at least ‘average’ ability to communicate. Google will only consider a candidate with poor communication ability if the candidate has a ‘brilliant’ innovation capability. Write a program that will help Google to programmatically determine eligibility of the fifteen candidates for these positions, and print the output on the console.

**2. Description of the Code:**

Nothing special to note, followed the API and the guidelines for Google candidate selection in my getEligibleCandidates method.

**3. Errors and Warnings:**

Table 1: List of Errors and Warnings Encountered in the Program

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Errors / Warnings** | **Details** | **How I solved them** |
| 1 | none |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 |  |  |  |

**4. Sample Input and Output:**

**Case 1: Given an ArrayList of Candidates, when the list is processed through the getEligibleCandidates method, then only the candidates with the required criteria are printed out to the user.**

**5. Discussion:**

**None, straight forward.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **LEARNING PROFILE FOR ASSIGNMENT#\_1\_\_\_\_\_ AND QUESTION#\_9\_\_\_\_\_\_\_** | | | | | |
| *Name* | *:* | *Steven Morrissey* | *Due Date* | *:* |  |
| *Student ID* | *:* | *3300222* | *Submission Date* | *:* |  |

**1. Problem Statement:**

Write a program that iterates through numbers from 0 to 113 using a loop. Print the numbers, one number per line. As you print each number, say x, also print the following when appropriate, separated by commas: I. If the number is odd, print “x is odd” II. If the number is divisible by 5, print “hi five” III. If the total of a number (x) and its subsequent number (x+1) is a value divisible by 7, print “wow” IV. If the number is prime, print “prime”.

**2. Description of the Code:**

Implemented all methods in the API, with the iterate() method calling every other method to verify the conditions and add them to a list to be printed out later on.

**3. Errors and Warnings:**

Table 1: List of Errors and Warnings Encountered in the Program

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Errors / Warnings** | **Details** | **How I solved them** |
| 1 | none |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 |  |  |  |

**4. Sample Input and Output:**

**Case 1: Given a sequence of numbers 1-113, Each number is checked one by one and the correct statements are printed out for each number whether they are odd, divisible by 5 or 7, or they are prime.**

**5. Discussion:**

**No discussion required, API was comprehensive enough to write the program from start to finish.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **LEARNING PROFILE FOR ASSIGNMENT#\_1\_\_\_\_\_ AND QUESTION#\_10\_\_\_\_\_\_\_** | | | | | |
| *Name* | *:* | *Steven Morrissey* | *Due Date* | *:* |  |
| *Student ID* | *:* | *3300222* | *Submission Date* | *:* |  |

**1. Problem Statement:**

Modify the following program to the specifications given below: I. Add a new status – SingleParent – where the tax is computed as a SINGLE but with a further reduction of $5000 per child. II. Add a new tax condition – if the income is greater than $249,999 for SINGLE, then add a tax of 25% on income amount above $150,000; if the income is greater than $349,999 for MARRIED, then add a tax of 35% on income amount above $200,000. III. Unknown status – if the status doesn’t belong to SINGLE or MARRIED or SINGLE\_PARENT, then compute a 33% tax on the income.

**2. Description of the Code:**

Added a few new constants: SINGLE\_PARENT, RATE 4 5 6, SINGLE\_BRACKET 3 4, MARRIED\_BRACKET 3 4. Added new conditions for the new brackets, new condition for calculation of SIGNLE\_PARENT status, and new condition for UNKNOWN status.

**3. Errors and Warnings:**

Table 1: List of Errors and Warnings Encountered in the Program

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Errors / Warnings** | **Details** | **How I solved them** |
| 1 | none | Was easy enough to follow the existing code and avoid errors |  |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 |  |  |  |

**4. Sample Input and Output:**

**Case 1: Given a user with status <status>, children <children> and income <income>, When the user inputs the data in the program, Then the expected tax <taxAmount> is displayed.**

|  |  |  |  |
| --- | --- | --- | --- |
| **status** | **children** | **income** | **taxAmount** |
| **SINGLE** | **0** | **20000** | **3000** |
| **SINGLE** | **0** | **30000** | **5611.50** |
| **SINGLE** | **0** | **200000** | **57654.50** |
| **SINGLE\_PARENT** | **1** | **20000** | **-2000** |
| **SINGLE\_PARENT** | **1** | **30000** | **611.50** |
| **SINGLE\_PARENT** | **3** | **30000** | **-9388.50** |
| **MARRIED** | **0** | **30000** | **4500** |
| **MARRIED** | **0** | **80000** | **17746** |
| **MARRIED** | **0** | **300000** | **86751** |

**5. Discussion:**

None, it was straight forward. No special libraries imported other than Scanner.