ACS-2947-001

Lab 2

Due by Friday September 22, 11:59 PM

Instructions

- Submit your `.java` files (together in a `Lab2.zip` file) via Nexus.
- Include your name and student number as a comment in every file.

Task

1. Code the generic `Pair` class from our class notes (L01_) and include a `toString` method.

   Create a simple class named `Product` that consists of a string `pName` and double `price`. Include getters and override the `toString` method to display the details of a product as shown in the sample output below.

   In a `Lab2_Driver` class, create two `Pair` objects that hold instances of `Product` and `Integer` (representing the quantity of the product bought).

   Display the product information and total price of the two products in the following manner (for*, 5 milk(4L) at 5.05 each and 2 cheese sticks at 3.99 each)*

   ```
   5 Milk(4L) : 5.05
   2 Cheese sticks : 3.99
   Total cost: $ 33.23
   ```

2. Consider the following:

   ```
   Duo<String> friends = new Duo<>("Harry", "Ron");
   friends.setThing2("Dobby");
   System.out.println(friends);
   ```

   Output:

   ```
   [Harry & Dobby]
   ```

   Write the code for the **generic** class `Duo` that stores two elements (`thing1` and `thing2`) of **the same** object type. Include all getters, setters, and any other applicable methods.

   Add the 3 lines of code above to your `Lab2_Driver` to test your `Duo` class.

3. Code the `SinglyLinkedList` class from our class notes (L03_). Override the `toString` method by using `StringBuilder` to list every element, in order from head to tail.

   In the same `Lab2_Driver` file, create a `SinglyLinkedList` instance called `productList` that holds a list of `Product`.

   Add elements into the `SinglyLinkedList` instance so that the following statement (in your driver file):

   ```
   System.out.println(productList);
   ```

   would display the following output:

   ```
   [Milk(4L) : 5.05, Cheese sticks : 3.99, Yoghurt : 4.99,
   Ice cream(1L) : 10.99, Butter : 5.65]
   ```

   Note that your list must be built to contain and display these products in this order.

Submission

Submit your **`Lab2.zip`** file that include all the following files (`Product.java, Pair.java, Duo.java, SinglyLinkedList.java, Lab2_Driver.java`) via **Nexus**.

EXTRA WORK: Do not submit

1. In a driver class named `GenericMethodDemo`:
   Create a static generic method named `contains` that determines if a given element is found in the array. The method has 2 parameters: an array of generic elements and the generic element to find.

   Illustrate your method by displaying the results of the following:

   a) Check to see if the number 4 is in the list:
      5, 9, 8, 7, 0, 1, 3, 4
      Note that you will need to declare your array of type `Integer`

   b) Check to see if "Billy" is in the following list of names:
      Alice, Bob, Charlie, Dan Emma, Freddy, George, Hassan

   c) Create a custom class of your choice with sample data to fill an array and check if a given object is found in the list.

2. Download the `.java` files for the `BookReport` and `ReportMarks` classes (Extra work folder). Inspect the code and sample data in the driver.

   Complete the `AddMark()` method which adds and stores each submitted `BookReport` object by order of score. When adding *each* submission, check to see if it is on time, and if so, add it to the array accordingly (highest to lowest mark). If not, change its score to 0 and simply append it to the end of the list.

   Note:
   - Only add to this method, no other additions/changes should be made to the class
   - Refer to the Scoreboard example in your notes/text

3. Provide the algorithms for the following methods to be added in the SinglyLinkedList class. Then include the implementation in the `SinglyLinkedList` class.
   a. Clear – Removes all the items in the linked list
   b. Remove<*item*> – removes all similar *item* from the linked list
   c. Reverse – reverse the items in the linked list
   d. RemoveLast – removes the last element in the linked list
   e. Copy – make a deep copy of the linked list (see L04_)
   f. Contains<*item*> – determines if *item* exists in the linked list (see L04_)