|  |  |
| --- | --- |
| sait_icon_wordmark_horiz_text_black | **CPRG 251**  **Assignment 3 (Modules 9, 10, 11)**  **Winter 2021** |

**Name:**

**Marks:**  / 39

## **D2L Submission Instructions**

1. One ZIP file needs to submit to D2L with the following naming convention **CPRG251\_A3\_Firstname\_Lastname.zip** using your first and last name.

If working in a group of two (2) or more, only one team member needs to submit to D2L (both can if you so wish). Both members will receive the same feedback. The file should have the following naming convention: **CPRG251\_A3\_Lastname of member 1\_Lastname of member 2.zip**

1. The ZIP file must contain the following:   
   1. The following directory structure:

* bin/ – Compiled Java files.
* src/ – Java source code files:
  + sait/sll/utility/
  + sait/sll/problemdomain/
* doc/ – Generated Javadoc files.
  + Ensure the *private* option is checked and everything is included in the generated documentation.
* lib/ – Any third-party libraries. This folder can be empty.
* res/ – Any resource or data files.
* test/ – Unit test cases.

1. A text file named **Readme.txt** in the root folder of the ZIP archive and contain:

* A project title.
* What the program does.
* The date.
* The author
* How to run the program.

1. It is to be built using only Eclipse IDE and JDK 1.8x.

## **Assignment Instructions**

1. You will use only Eclipse IDE.
2. The due date for this assignment is posted in D2L in the assignment submission area and in the provided calendar located in the *Course Information* area**.** Any assignment submitted after the due date will receive a mark of zero, but feedback may be given.
3. Submissions must be student’s original work. Refer to the Academic Misconduct (AC.3.4) policies and procedures.

## **Problem**

You are to implement the abstract data type (ADT) for a linked list. The supplied interface LinkedListADT contains the signatures for the methods.

SLL class must implement the LinkedListADT interface. Each abstract method defined in the LinkedListADT should be completely implemented, and any exceptions are to be appropriately propagated in the SLL class. The Node class represents a node in the linked list.

The linked list interface will contain the following functionality **(Don’t use already implemented classes/methods/libraries for the following functionalities)**:

* Prepend an item to the beginning of the linked list.
* Append an item to the end of the linked list.
* Insert an item at a specific index in the linked list.
* Replace an item in the linked list.
* Get an item at an index in the linked list.
* Get the index of an item in the linked list.
* Check if linked list has an item.
* Clear all items in the linked list.
* Get the number of items in the linked list.
* Remove an item at an index in the linked list.

Place the supplied interface (LinkedListADT), the implementing class (SLL) and the Node class in the **sait.sll.utility** package.

## **JUnit Testing**

The supplied LinkedListTest class is a test suite that perform unit testing on an implemented linked list. Some of the test cases have been implemented. You are not to modify the implemented test cases. You are to come up with and implement any missing test cases.

The following JUnit test cases have been provided:

* The linked list is empty.
* An item is prepended.
* An item is appended.
* An item is inserted at index.
* An item is replaced.
* An item is deleted.
* An existing item is found and retrieved.

You will use your implemented JUnit test cases to achieve the needed code coverage.

## **Serialization**

The implemented linked list SLL must be serialized. The SLL object is to be stored in binary format. The object will be able to be reconstructed from its binary form.

The supplied SerializationTests class Junit test cases test for object serialization of the SLL. No changes are to be made to the JUnit test suite. All the JUnit test cases will need to pass with no errors given. The objects are serialized to memory in the JUnit test suite and therefore, there are no changes made on the hard drive.

The supplied **User** class in the **sait.sll.problemdomain** package will be used as the data type for the items in the linked list.

## **Notes:**

## To follow the object-oriented principles, your project should contain ONLY the following classes and methods in their respective package.

|  |  |  |
| --- | --- | --- |
| **Package** | **Class** | **Methods** |
| sait.sll.utility | LinkedListADT  SLL  Node | append, insert, prepend, set, size, get, indexOf, contains, isEmpty, clear, and remove  Implementation of the above interface methods  Getters and setters |
| sait.sll.problemdomain | User | Getters, setters, and equals |

## **Marking Guide**

|  |  |  |  |
| --- | --- | --- | --- |
| **Follows submission guidelines** | | | |
| 1. | Correct files |  |  |
| 2. | File naming conventions followed |  |  |
| 3. | NO extra files |  |  |
| **Subtotal** | |  | **/3** |
| **Linked list ADT** | | | |
|  | **Linked List ADT** |  |  |
| 4. | Uses proper package structure. |  |  |
| 5. | Adds and replaces items in linked list. |  |  |
| 6. | Removes items from linked list. |  |  |
| 7. | Retrieves items from linked list. |  |  |
| 8. | Finds existing items in linked list. |  |  |
| 9. | Clears and checks if linked list is empty. |  |  |
| **Subtotal** | |  | **/19** |
|  | **Linked List Unit testing** |  |  |
| 10. | Uses implemented linked list ADT as test fixture. |  |  |
| 11. | Created and implemented new JUnit test cases. |  |  |
| **Subtotal** | |  | **/7** |
|  | **Serialization Unit Testing** |  |  |
| 12. | Uses implemented linked list ADT as test fixture. |  |  |
| 13. | Writes linked list using object stream. |  |  |
| 14. | Reads linked list using object stream. |  |  |
| 15. | Applicable data is ignored in serialization process. |  |  |
| **Subtotal** | |  | **/10** |
| **Total** | |  | **/39** |