# Assignment 01 LinkedList, Stack, Queue and Deque 44642

**Important:** Copying and pasting code is *not allowed* (except as noted in the instructions). All code should have comments throughout, unique for each student. Code that is too similar in formatting, spacing, and variable names may be considered plagiarism/academic dishonesty. Bottom line, do your own work and you should be fine.

# Exercise 01: LinkedList

All the classes related to Exercise 01 must be placed in a package named “**linkedlist**”.

1. Create a class name **Student** with instance variables name and id of type String and int respectively.
2. Write a constructor with arguments of instances that initializes the class instances.
3. Override **equals()** and **hashCode()** methods.
4. Create a class name **StudentOperations** with instance variable studentList of type LinkedList<Student>.
5. Write a no-arg constructor that initializes studentList to a new empty LinkedList of type Student.
6. Write a method **addStudents()** that adds students into studentList by reading from a file (refer to the input file provided for sample input).
7. Write a method **removeDuplicates()** that removes duplicates from studentList.
8. Write a method **displayDuplicatesNamesByRecursion()** that takes an argument of Iterator type and returns the duplicate student names as String types, comma separated (refer to sample output for formatting).
9. Create a class **StudentDriver** and place the following code in it.

|  |
| --- |
| public class StudentDriver {  /\*\*  \* @param args the command line arguments  \* @throws java.io.FileNotFoundException  \*/  public static void main(String[] args) throws FileNotFoundException {  // TODO code application logic here  StudentOperations stuOperationsOne = new StudentOperations();  StudentOperations stuOperationsTwo = new StudentOperations();  stuOperationsOne.addStudents();  stuOperationsTwo.addStudents();  System.out.println("List one: ");  displayData(stuOperationsOne.getStudentList());  stuOperationsOne.removeDuplicates();  System.out.println("List one without duplicates: ");  displayData(stuOperationsOne.getStudentList());  System.out.println("List two: ");  displayData(stuOperationsTwo.getStudentList());  System.out.println("List two duplicate names: ");  System.out.println(stuOperationsTwo.  displayDuplicatesNamesByRecursion(  stuOperationsTwo.getStudentList().iterator()));  }    public static void displayData(LinkedList<Student> studentList){  System.out.print("{");  for(Student eachStudent: studentList){  System.out.println(eachStudent.getId()+", "+eachStudent.getName());  }  System.out.println("}");  }    } |

**Sample Output:**

|  |
| --- |
| List one:  {414339, Maria Rodriguez  414356, Robert Smith  414376, Robert Smith  414339, Maria Rodriguez  414776, Mary Smith  414776, Mary Smith  414867, Maria Martinez  414339, Maria Rodriguez  414745, Martinez Maria  }  List one without duplicates:  {414339, Maria Rodriguez  414356, Robert Smith  414376, Robert Smith  414776, Mary Smith  414867, Maria Martinez  414745, Martinez Maria  }  List two:  {414339, Maria Rodriguez  414356, Robert Smith  414376, Robert Smith  414339, Maria Rodriguez  414776, Mary Smith  414776, Mary Smith  414867, Maria Martinez  414339, Maria Rodriguez  414745, Martinez Maria  }  List two duplicate names:  Maria Rodriguez, Mary Smith, |

# Exercise 02: StacksDeques

All the classes related to Exercise 02 must be placed in a package named “**stacksanddeques**”.

1. Classes: This package has one generic class named **AStack<E>**

Private instance variable

private ArrayDeque<E> myStack;

**Constructors and methods:**

* One **no-arg constructor**: initializes myStack to a new empty ArrayDeque of type E
* **push(E element)** – adds an element to the top of the stack; no return value
* **pop()** – removes and returns the top element
* **peek()** – returns, but does not remove, the top element
* **size()** – returns the current size of the stack
* **isEmpty()** – returns true if the stack is empty; false otherwise

1. Classes: This package has one concrete class named **Book**

Private instance variables

private String title;

private String author;

**Constructors and methods:**

* One **two-argument constructor**: initializes instance variables with the arguments passed.
* **Getter** methods for the two instance variables.
* **toString()** – returns the formatted string with title of the book and author name in the below format.

**Sample Output**

|  |
| --- |
| Quidditch Through the Ages  Joanne Rowling |

1. Classes: This package has two main classes, **BooksStack** and **GreetingsStack**

BooksStack details:

* Create a new **AStack** containing objects of type Book, named books.
* Read input from **books.txt**, which contains book name and its author – one per line. **books.txt** is provided to you.
* Push each book read onto the stack.
* Print all the books in the stack.
* If there is a book with name “Fantastic Beasts: The Crimes of Grindelwald” remove it from the stack.
* Print the updated stack after removing the book.

Please refer sample output for formatting and other details.

**Sample Output for BookStack:**

|  |
| --- |
| All books in the stack from top to bottom  -------------------------------------  Charlie and the Chocolate Factory  Roald Dahl  -------------------------------------  Percy Jackson & the Olympians  Rick Riordan  -------------------------------------  A Game of Thrones  George Raymond Richard Martin  -------------------------------------  The Lord of the Rings  John Ronald Reuel Tolkien  -------------------------------------  Fantastic Beasts: The Crimes of Grindelwald  Joanne Rowling  -------------------------------------  Quidditch Through the Ages  Joanne Rowling  -------------------------------------  Fantastic Beasts and Where to Find Them  Joanne Rowling  -------------------------------------  Harry Potter and the Sorcerer's Stone  Joanne Rowling  -------------------------------------  Stack after removing book with name Fantastic Beasts: The Crimes of Grindelwald  -------------------------------------  Charlie and the Chocolate Factory  Roald Dahl  -------------------------------------  Percy Jackson & the Olympians  Rick Riordan  -------------------------------------  A Game of Thrones  George Raymond Richard Martin  -------------------------------------  The Lord of the Rings  John Ronald Reuel Tolkien  -------------------------------------  Quidditch Through the Ages  Joanne Rowling  -------------------------------------  Fantastic Beasts and Where to Find Them  Joanne Rowling  -------------------------------------  Harry Potter and the Sorcerer's Stone  Joanne Rowling  ------------------------------------- |

GreetingsStack details**:**

1. Create a new **AStack** containing objects of type Character, named **greeting**.
2. Push Happy New Year onto **greeting** stack.
3. Now, display the Happy New Year from the stack.

**Exercise 03: LinkedBlockingQueue**

All the classes related to Exercise 03 must be placed in a package named “**queue**”.

1. Create a **LinkedBlockingQueue** of capacity 5 with your preferred data type.
2. Insert 5 Elements. (Print the leftover capacity at the time of each insertion.)
3. Try to insert the 6th element using the offer method, waiting for time interval of 5 seconds.
4. Try 3rd step again using add method.
5. Discuss what happened in 3rd and 4th steps. Print the answer.
6. Display if any duplicate elements exist. (Make sure it works for any data)
7. Invoke **poll**() and **remove**() methods.
8. Now make the queue object empty.
9. Invoke **peek**() and **element**() methods.
10. Comment on four methods from 7th and 9th steps. Print answer.

(Hint: Use try catch to overcome unexpected termination of program wherever necessary.)

**Submit your solution by following the steps below:**

* Save your files in NetBeans.
* Zip your entire Project. (It should be called **LastnameAssignment01** where Last name is your last name.)
* Submit the zip file to the **Assignment01** drop box.
* Download the zip file which you have submitted.
* Look into the zip file and verify the class files, and Javadoc in the zip folder are updated. If not, resave your project in NetBeans and resubmit.