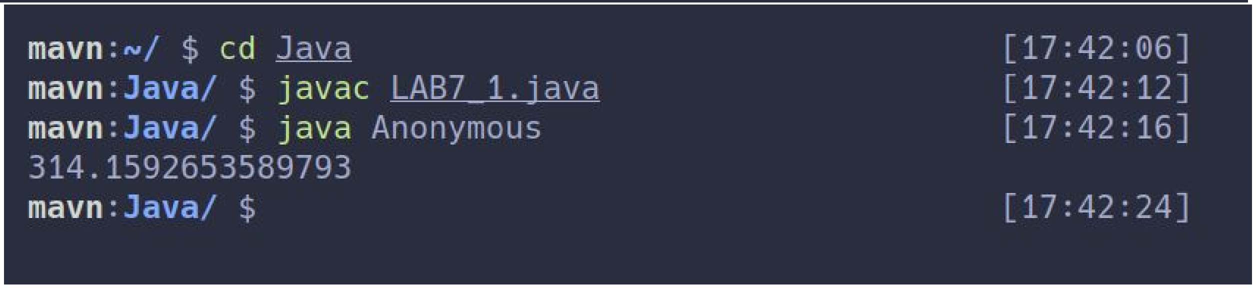
Manavendra Sen | 199302058 | IT 3A

OOP LAB 7&8

Exercise 1.2 :

```java
class Circle {
 private int rad;

 public Circle(int radius) {
   rad = radius;
 }

 public double area() {
   return Math.PI * rad * rad;
 }
}

class Anonymous {
 public Circle getCircle(int radius) {
   return new Circle(radius);
 }

 public static void main(String[] args) {
   Anonymous p = new Anonymous();
   Circle w = p.getCircle(10);
   // The output here should give correct value of area
   // of the circle.
   System.out.println(w.area());
 }
}
```



```
mavn:~/ $ cd Java                                              [17:42:06]
mavn:Java/ $ javac LAB7_1.java                                 [17:42:12]
mavn:Java/ $ java Anonymous                                    [17:42:16]
314.1592653589793
mavn:Java/ $                                                   [17:42:24]
```

EXERCISE 7.1
The L&L Bank can handle up to 30 customers who have savings accounts. Design and implement a program that manages the accounts. Keep track of key information and allow each customer to make deposits and withdrawals. Produce appropriate error messages for invalid transactions.Do this practice problem using ArrayListand Iterator.

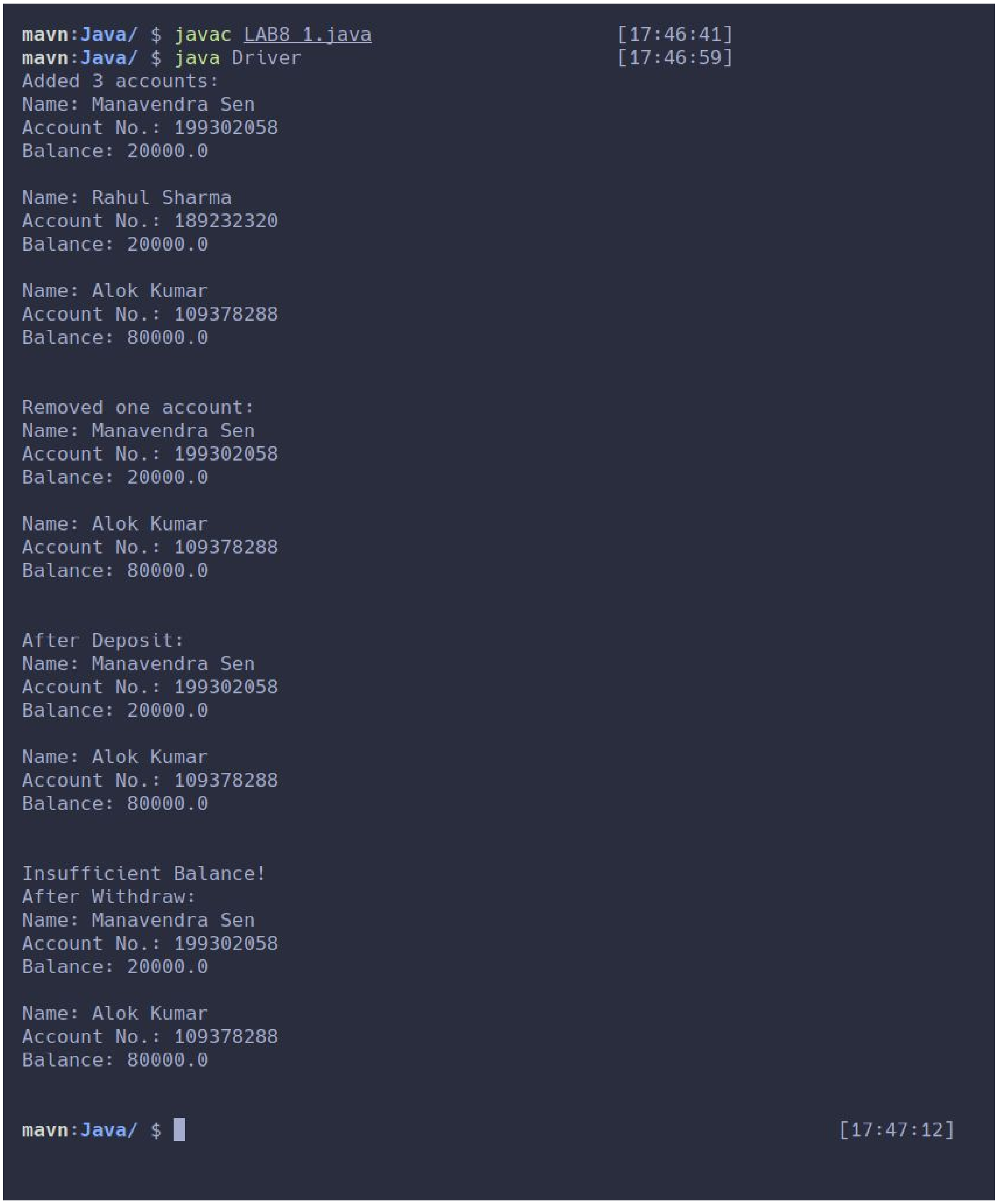
(A) Create an Account class that tracks individual customer information.
(B) Complete the code of Bank class as per the commented instructions

```
import java.util.ArrayList;
import java.util.List;
import java.util.Iterator;

class Account {
 private long acctNumber;
 private double balance;
 private String name;

 /*
  * Complete the Account class by adding proper constructor,accessor method and
  * mutator method as required.Override toSring() method to display account
  * details.
  */
 Account(long acc, double balance, String name) {
   this.acctNumber = acc;
   this.balance = balance;
   this.name = name;
 }

 public long getAcctNumber() {
   return acctNumber;
 }

 public double getBalance() {
   return balance;
 }
 public String getName() {
   return name;
 }

 public void setAcctNumber(long acctNumber) {
   this.acctNumber = acctNumber;
 }

 public void setBalance(double balance) {
   this.balance = balance;
 }

 public void setName(String name) {
   this.name = name;
 }

 @Override
 public String toString() {
   return ("Name: " + name + "\n" + "Account No.: " + acctNumber + "\n" +
"Balance: " + balance);
 }
}

class Bank {
 private ArrayList<Account> accts;
 static int maxActive = 30;
 static int currentNumber = 0;

 Bank() {
   accts = new ArrayList<Account>();
 }

 public boolean addAccount(Account newone) {
   /*
    * Write the code for adding new account, return false if account can't be
    * created
```

```java
     */
    if (currentNumber > maxActive) {
      System.out.println("At full capacity");
      return false;
    }

    if (accts.indexOf(newone) != -1)
      return false;

    accts.add(newone);
    currentNumber++;
    return true;
  }

  public boolean removeAccount(long acctnum) {
    /*
     * Write the code for removing the account, return false if account does not
     * exist
     */
    Iterator itr = accts.iterator();
    int flag = 0;
    while (itr.hasNext()) {
      Account current = (Account) itr.next();
      if (current.getAcctNumber() == acctnum) {
        flag++;
        itr.remove();
        currentNumber--;
        break;
      }
    }
    if (flag == 0)
      return false;

    return true;
  }
```

```
public double deposit(long acctnum, double amount) {
   /*
    * Write the code for depositing specified amount to the account, return -1
if account does not exist
    */
   double res = -1;
   Iterator itr = accts.iterator();
   while (itr.hasNext()) {
     Account current = (Account) itr.next();
     if (current.getAcctNumber() == acctnum) {
       res = current.getBalance() + amount;
       current.setBalance(res);
       break;
     }
   }
   return res;
 }

public double withdraw(long acctnum, double amount) {
   /*
    * Write the code for withdrawing specified amount from the account, return
-1 if insufficient balance or account does not exist
    */
   double res = -1;
   Iterator itr = accts.iterator();
   while (itr.hasNext()) {
     Account current = (Account) itr.next();
     if (current.getAcctNumber() == acctnum) {
       if (current.getBalance() < amount) {
         System.out.println("Insufficient Balance!");
         break;
       }
       res = current.getBalance() - amount;
       current.setBalance(res);
       break;
     }
   }
   return res;
 }

 // override toString() method to display details of all the accounts in bank
 @Override
 public String toString() {
   String res = "";
   Iterator itr = accts.iterator();
   while (itr.hasNext()) {
     Account current = (Account) itr.next();
     res += (current.toString() + "\n\n");
   }
   return res;
 }
}

class Driver {
 public static void main(String[] args) {
   Bank LnL = new Bank();

   Account a1 = new Account(199302058, 20000.0, "Manavendra Sen");
   Account a2 = new Account(189232320, 20000.0, "Rahul Sharma");
   Account a3 = new Account(109378288, 80000.0, "Alok Kumar");

   LnL.addAccount(a1);
   LnL.addAccount(a2);
   LnL.addAccount(a3);
   System.out.println("Added 3 accounts: \n" + LnL);
   LnL.removeAccount(189232320);
   System.out.println("Removed one account: \n" + LnL);

   LnL.deposit(199303058, 100000.0);
   System.out.println("After Deposit: \n" + LnL);
```

```java
   LnL.withdraw(199302058, 120000.0);
   System.out.println("After Withdraw: \n" + LnL);
 }
}
```



```
mavn:Java/ $ javac LAB8_1.java                    [17:46:41]
mavn:Java/ $ java Driver                          [17:46:59]
Added 3 accounts:
Name: Manavendra Sen
Account No.: 199302058
Balance: 20000.0

Name: Rahul Sharma
Account No.: 189232320
Balance: 20000.0

Name: Alok Kumar
Account No.: 109378288
Balance: 80000.0


Removed one account:
Name: Manavendra Sen
Account No.: 199302058
Balance: 20000.0

Name: Alok Kumar
Account No.: 109378288
Balance: 80000.0


After Deposit:
Name: Manavendra Sen
Account No.: 199302058
Balance: 20000.0

Name: Alok Kumar
Account No.: 109378288
Balance: 80000.0


Insufficient Balance!
After Withdraw:
Name: Manavendra Sen
Account No.: 199302058
Balance: 20000.0

Name: Alok Kumar
Account No.: 109378288
Balance: 80000.0


mavn:Java/ $                                                      [17:47:12]
```

Exercises:

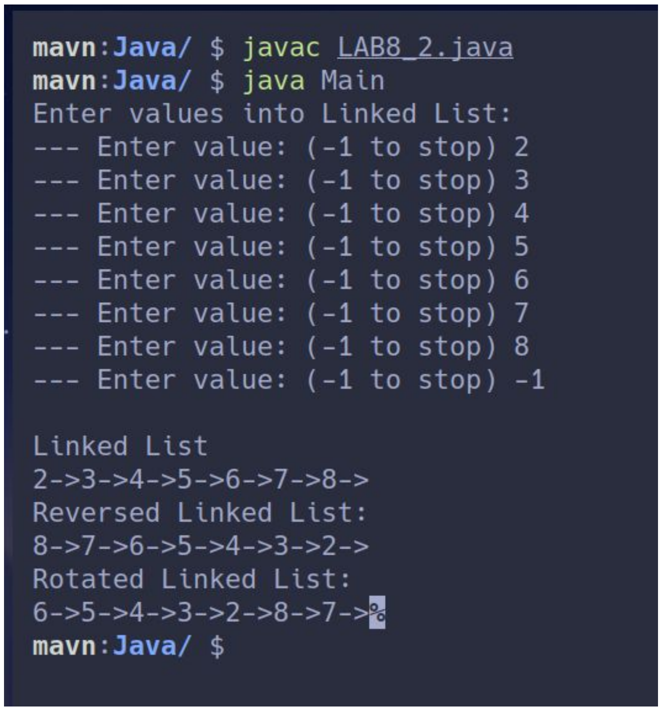
1. Add a function in the MyLinkedList class to reverse the given linked list. The function should return the new head of the linked list.
2. Add a function in the MyLinkedList class to rotate the given linked list counter-clockwise by k nodes. The function should return the new head of the linked list. For example, if the given linked list is 10->20->30->40->50->60 and k is 4, the list should be modified to 50->60->10->20->30->40. Assume that k is smaller than the count of nodes in the linked list and positive.

```java
import java.io.*;
import java.util.Scanner;

class LinkedList {
 Node head; // head of list

 // Linked list Node.
 // This inner class is made static
 // so that main() can access it
 static class Node {

   int data;
   Node next;

   // Constructor
   Node(int d) {
     data = d;
     next = null;
   }
 }

 // Method to insert a new node
 public static LinkedList insert(LinkedList list, int data) {
   // Create a new node with given data
   Node new_node = new Node(data);
   new_node.next = null;

   // If the Linked List is empty,
   // then make the new node as head
```

```
  if (list.head == null) {
    list.head = new_node;
  } else {
    // Else traverse till the last node
    // and insert the new_node there
    Node last = list.head;
    while (last.next != null) {
      last = last.next;
    }

    // Insert the new_node at last node
    last.next = new_node;
  }

  // Return the list by head
  return list;
}

// Method to print the LinkedList.
public static void printList(LinkedList list) {
  Node currNode = list.head;

  // Traverse through the LinkedList
  while (currNode != null) {
    // Print the data at current node
    System.out.print(currNode.data + "->");

    // Go to next node
    currNode = currNode.next;
  }
}

public static LinkedList reverseList(LinkedList list) {

  LinkedList reversedList = new LinkedList();
  reversedList = list;
  Node prev = null;
  Node next = null;
  Node curr = list.head;

  while (curr != null) {
    next = curr.next;
    curr.next = prev;
    prev = curr;
    curr = next;

  }
  reversedList.head = prev;
  return reversedList;
}

public static LinkedList rotateList(LinkedList list, int k) {
  int length = 0;
  LinkedList reversedList = new LinkedList();
  rotatedList = list;
  Node kth = null;
  Node kth1 = null;
  Node curr = list.head;
  Node last = null;
  while (curr != null) {
    length++;
    if (length == k) {
      kth = curr;
      kth1 = curr.next;
    }
    if (curr.next == null) {
      last = curr;
    }
    curr = curr.next;
  }

  last.next = list.head;
```

```java
   kth.next = null;
   rotateddList.head = kth1;

   return rotateddList;
 }
}

class Main {
 // Driver code
 public static void main(String[] args) {
   /* Start with the empty list. */
   LinkedList list = new LinkedList();
   LinkedList LL1 = new LinkedList();
   LinkedList LL2 = new LinkedList();

   Scanner sc = new Scanner(System.in);
   int inp = 0;

   System.out.println("Enter values into Linked List: ");
   while (true) {
     System.out.print("--- Enter value: (-1 to stop) ");
     inp = sc.nextInt();
     if (inp == -1)
       break;
     list = LinkedList.insert(list, inp);
   }

   LL1 = list;
   LL2 = list;

   System.out.println("\nLinked List");
   LinkedList.printList(list);

   LL1 = LinkedList.reverseList(LL1);
   System.out.println("\nReversed Linked List: ");
   LinkedList.printList(LL1);
```

```java
    LL2 = LinkedList.rotateList(LL2, 2);
    System.out.println("\nRotated Linked List: ");
    LinkedList.printList(LL2);

    sc.close();
  }
}
```



```
mavn:Java/ $ javac LAB8_2.java
mavn:Java/ $ java Main
Enter values into Linked List:
--- Enter value: (-1 to stop) 2
--- Enter value: (-1 to stop) 3
--- Enter value: (-1 to stop) 4
--- Enter value: (-1 to stop) 5
--- Enter value: (-1 to stop) 6
--- Enter value: (-1 to stop) 7
--- Enter value: (-1 to stop) 8
--- Enter value: (-1 to stop) -1

Linked List
2->3->4->5->6->7->8->
Reversed Linked List:
8->7->6->5->4->3->2->
Rotated Linked List:
6->5->4->3->2->8->7->%
mavn:Java/ $
```