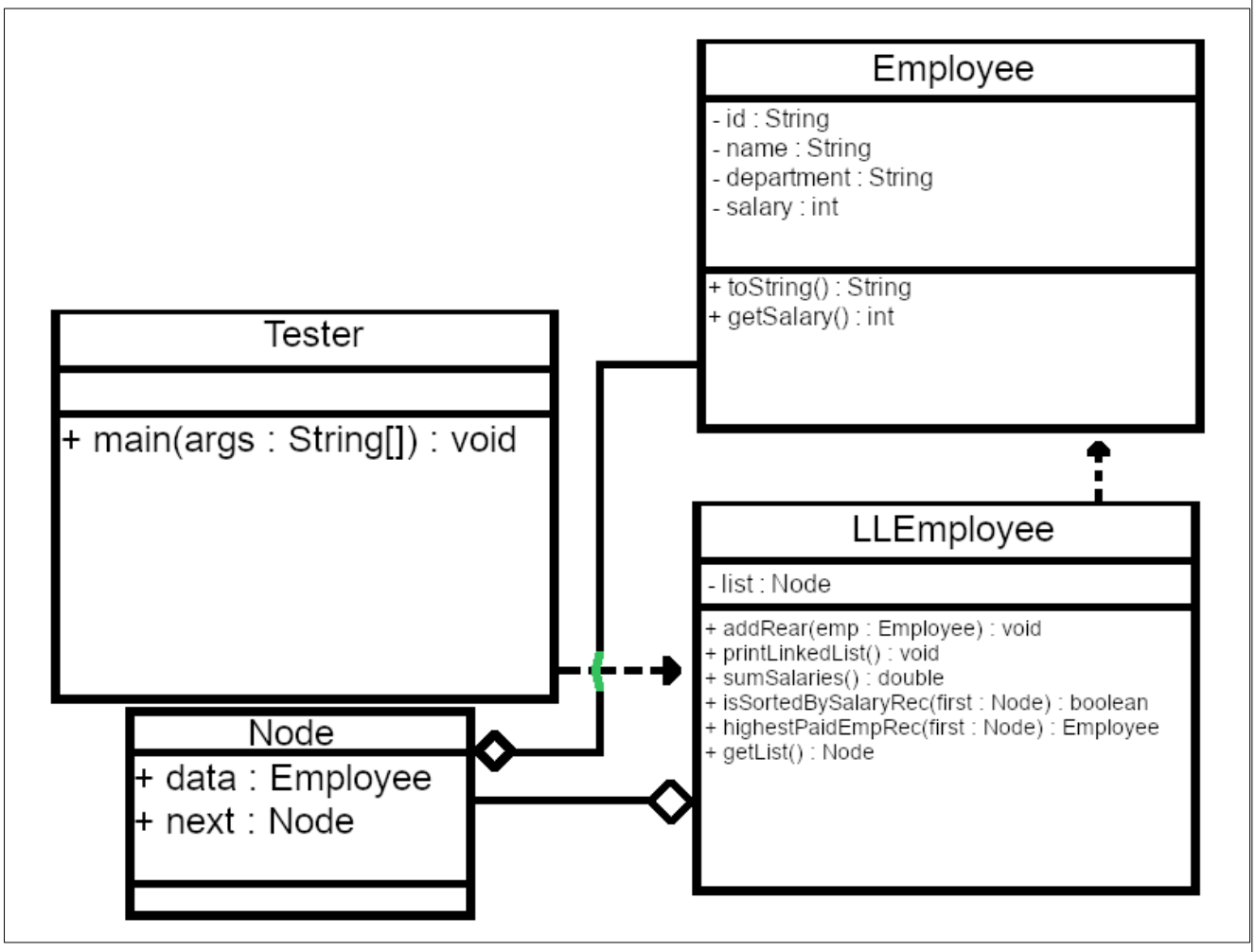
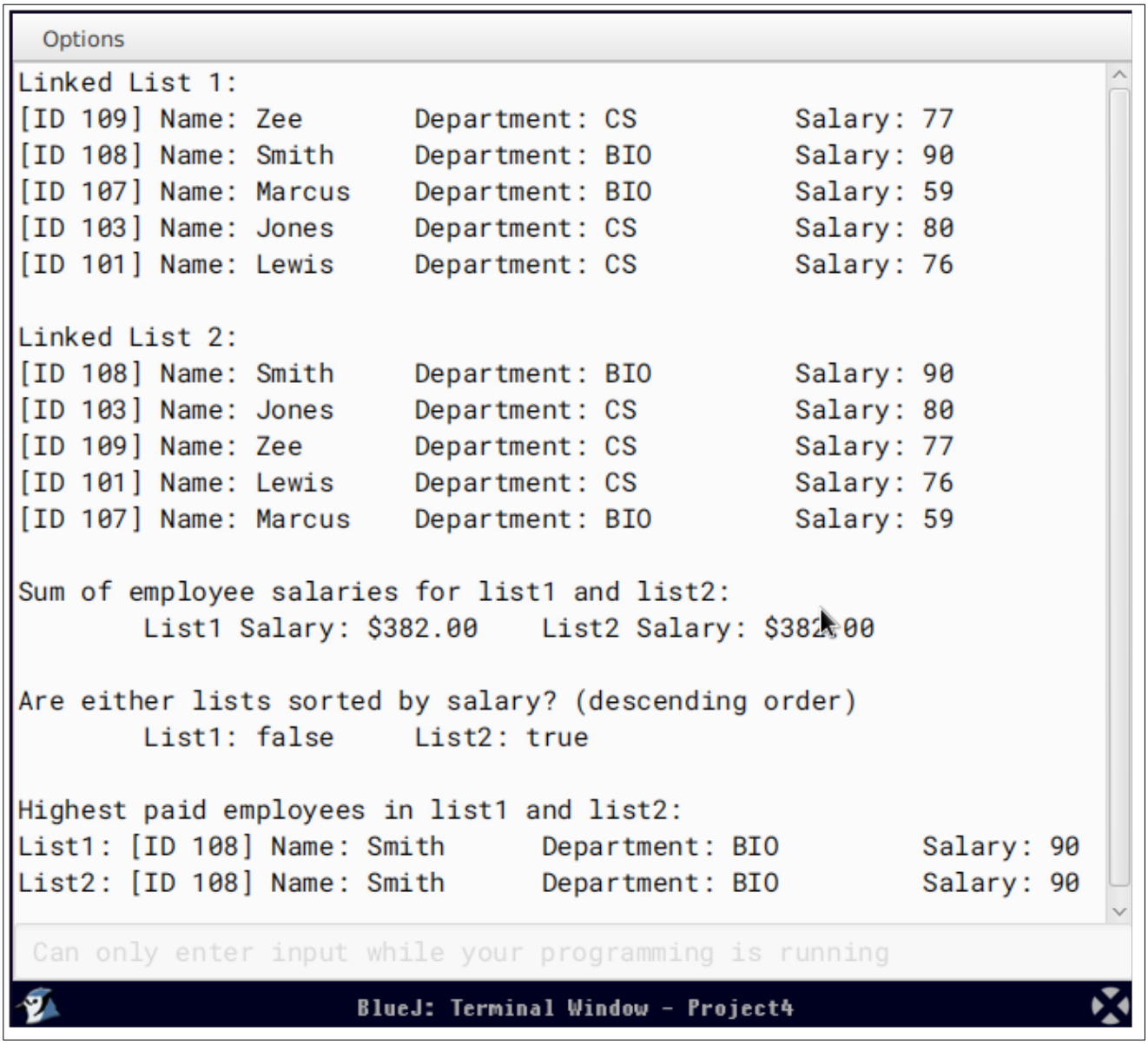
UML


Employee
- id : String
- name : String
- department : String
- salary : int
+ toString() : String
+ getSalary() : int
Tester
+ main(args : String[]) : void
LLEmployee
- list : Node
+ addRear(emp : Employee) : void
+ printLinkedList() : void
+ sumSalaries() : double
+ isSortedBySalaryRec(first : Node) : boolean
+ highestPaidEmpRec(first : Node) : Employee
+ getList() : Node
Node
+ data : Employee
+ next : Node

Output


Options
Linked List 1:
[ID 109] Name: Zee Department: CS Salary: 77
[ID 108] Name: Smith Department: BIO Salary: 90
[ID 107] Name: Marcus Department: BIO Salary: 59
[ID 103] Name: Jones Department: CS Salary: 80
[ID 101] Name: Lewis Department: CS Salary: 76
Linked List 2:
[ID 108] Name: Smith Department: BIO Salary: 90
[ID 103] Name: Jones Department: CS Salary: 80
[ID 109] Name: Zee Department: CS Salary: 77
[ID 101] Name: Lewis Department: CS Salary: 76
[ID 107] Name: Marcus Department: BIO Salary: 59
Sum of employee salaries for list1 and list2:
List1 Salary: $382.00 List2 Salary: $382.00
Are either lists sorted by salary? (descending order)
List1: false List2: true
Highest paid employees in list1 and list2:
List1: [ID 108] Name: Smith Department: BIO Salary: 90
List2: [ID 108] Name: Smith Department: BIO Salary: 90
Can only enter input while your programming is running
BlueJ: Terminal Window - Project4

TestList.java

```
/**
 * @author (Kyle Guarco)
 * @version (July 20, 2020)
 */
public class TestList
{
    public static void main(String[] args)
    {
        LLEmployee list1 = new LLEmployee();
        LLEmployee list2 = new LLEmployee();

        { // Body that holds local variables 'rawList1' and 'rawList2'.
          // Used to populate the above linked lists.
          Employee[] rawList1 = {
              new Employee("101", "Lewis", "CS ", 76),
              new Employee("103", "Jones", "CS ", 80),
              new Employee("107", "Marcus", "BIO", 59),
              new Employee("108", "Smith", "BIO", 90),
              new Employee("109", "Zee", "CS ", 77)
          };
          for (Employee emp : rawList1)
              list1.addRear(emp);

          Employee[] rawList2 = {
              new Employee("107", "Marcus", "BIO", 59),
              new Employee("101", "Lewis", "CS ", 76),
              new Employee("109", "Zee", "CS ", 77),
              new Employee("103", "Jones", "CS ", 80),
              new Employee("108", "Smith", "BIO", 90)
          };
          for (Employee emp : rawList2)
              list2.addRear(emp);
        }

        System.out.println("Linked List 1:");
        list1.printLinkedList();

        System.out.println("\nLinked List 2:");
        list2.printLinkedList();

        System.out.println("\nSum of employee salaries for list1 and list2: ");
        System.out.printf("\tList1 Salary: $%.2f\t List2 Salary: $%.2f\n",
                list1.sumSalaries(), list2.sumSalaries());
```

```
        System.out.println("\nAre either lists sorted by salary? (descending order) ");
        System.out.printf("\tList1: %b\t List2: %b\n",
                list1.isSortedBySalaryRec(list1.getList()),
                list2.isSortedBySalaryRec(list2.getList()));

        System.out.println("\nHighest paid employees in list1 and list2: ");
        System.out.println("List1: " + list1.highestPaidEmpRec(list1.getList()).toString());
        System.out.println("List2: " + list2.highestPaidEmpRec(list2.getList()).toString());
    }
}
```

LLEmployee.java

```
/**
 * @author (Kyle Guarco)
 * @version (July 20, 2020)
 */
public class LLEmployee
{
   private Node list;

   public LLEmployee()
   {
      this.list = null;
   }

   public void addRear(Employee emp)
   {
      Node newNode = new Node(emp);

      if (list == null)
      {
         list = newNode;
         return;
      }

      newNode.next = list;
      list = newNode;
   }

   public void printLinkedList()
   {
      Node n = list;
      while (n != null)
      {
         System.out.println(n.data);

         n = n.next;
      }
   }

   public double sumSalaries()
   {
      double sum = 0d;

      Node n = list;
      while (n != null)
```

```
    {
        sum += n.data.getSalary();

        n = n.next;
    }

    return sum;
}

/*
 * Though it isn't specifed in the assignment documentation, this function should be
 * able to figure out whether the list is sorted in descending order (largest -> smallest).
 *
 * This is implied when looking at the 'list2' layout, and also assuming the test should
 * return at least one true value.
 */
public boolean isSortedBySalaryRec(Node first)
{
    if (first.next == null)
        return true;

    Employee current = first.data,
    next = first.next.data;
    boolean sorted = current.getSalary() > next.getSalary();

    return sorted && isSortedBySalaryRec(first.next);
}

public Employee highestPaidEmpRec(Node first)
{
    if (first.next == null)
        return first.data;

    Employee emp = first.data;
    Employee empNext = highestPaidEmpRec(first.next);

    return (emp.getSalary() < empNext.getSalary()) ? empNext : emp;
}

public Node getList()
{
    return list;
}

class Node
```

```
   {
      public Employee data;
      public Node next;

      public Node(Employee data)
      {
         this.data = data;
         this.next = null;
      }
   }
}
```

Employee.java

```
/**
 * @author (Kyle Guarco)
 * @version (July 20, 2020)
 */
public class Employee
{
    private String id, name, department;
    private int salary;

    public Employee(String id, String name, String department, int salary)
    {
        this.id = id;
        this.name = name;
        this.department = department;
        this.salary = salary;
    }

    @Override
    public String toString()
    {
        return String.format("[ID %s] Name: %s\t Department: %s\t Salary: %d",
                             id, name, department, salary);
    }

    public int getSalary()
    {
        return salary;
    }
}
```