Marco Martinez
CISP 430
3/6/19
Assignment 2

CLASS DIAGRAM

Classes

StringDataItem
GenericItemType
ListEntry
JList
Stack
Queue
PriorityQueue

Associations

StringDataItem(1) --- inherits --- (1)GenericItemType
ListEntry(1) --- includes --- (1)GenericItemType
JList(1) --- contains --- (m)ListEntry
Stack(1) --- inherits --- (1)JList
Queue(1) --- inherits --- (1)JList
PriorityQueue(1) --- inherits --- (1)Queue

StringDataItem Class Attributes

INSTANCE VARIABLES

(-) String myString

CLASS CONSTRUCTORS

(+) StringDataItem()
(+) StringDataItem(String s)
(+) StringDataItem(StringDataItem sdi)

CHANGE STATE SERVICES

(+) void set(String)

READ STATE SERVICES

(+) boolean isLess(GenericItemType)
(+) boolean isEqual(GenericItemType)
(+) boolean isGreater(GenericItemType)
(+) String get()
(+) String toString()

GenericItemType Class Attributes

READ STATE SERVICES

(+) abstract boolean isLess(GenericItemType)
(+) abstract boolean isEqual(GenericItemType)
(+) abstract boolean isGreater(GenericItemType)

ListEntry Class Attributes

INSTANCE VARIABLES

(-) GenericItemType data
(-) ListEntry next
(-) ListEntry prev

CLASS CONSTRUCTORS

(+) ListEntry()
(+) ListEntry(GenericDataItem)
(+) ListEntry(ListEntry)

CHANGE STATE SERVICES

(+) void setData(GenericItemType)
(+) void setNext(ListEntry)
(+) void setPrev(ListEntry)

READ STATE SERVICES

(+) GenericItemType getData()
(+) ListEntry getNext()
(+) ListEntry getPrev()

JList Class Attributes

INSTANCE VARIABLES

(-) ListEntry head
(-) ListEntry tail
(-) ListEntry currentIteration
(-) int totalCount
(-) int currentCount

CLASS CONSTRUCTORS

(+) JList()
(+) JList(GenericItemType)
(+) JList(ListEntry)
(+) JList(JList)
(+) JList(Stack)
(+) JList(Queue)
(+) JList(PriorityQueue)

CHANGE STATE SERVICES

(+) void init()
(+) void add_fromHead(GenericItemType)
(+) void add_fromHead(ListEntry)
(+) void add_fromMid(GenericItemType)
(+) void add_fromMid(ListEntry)
(+) void add_fromTail(GenericItemType)
(+) void add_fromTail(ListEntry)
(+) void bubbleSort_ascending()
(+) void bubbleSort_descending()
(+) GenericItemType linearSearch(GenericItemType)

(+) GenericItemType linearSearch(ListEntry)
(-) ListEntry lSearch(GenericItemType)
(+) void remove(GenericItemType)
(+) void remove(ListEntry)
(-) void delete(GenericItemType)
(+) void reverseList()

READ STATE SERVICES

(+) boolean isFull()
(+) boolean isEmpty()
(+) int getCount()
(+) ListEntry getStart()
(+) ListEntry getEnd()
(+) void Iterator_initialize()
(+) boolean Iterator_hasNext()
(+) GenericItemType Iterator_iterate()

Queue Class Attributes

CLASS CONSTRUCTORS

(+) Stack()
(+) Stack(GenericItemType)
(+) Stack(ListEntry)
(+) Stack(JList)
(+) Stack(Stack)
(+) Stack(Queue)
(+) Stack(PriorityQueue)

CHANGE STATE SERVICES

(+) void push(GenericItemType)
(+) GenericItemType pop()

READ STATE SERVICES

(+) GenericItemType showTop()

Queue Class Attributes

CLASS CONSTRUCTORS

(+) Queue()
(+) Queue(GenericItemType)
(+) Queue(ListEntry)
(+) Queue(JList)
(+) Queue(Stack)
(+) Queue(Queue)
(+) Queue(PriorityQueue)

CHANGE STATE SERVICES

(+) void enQueue(GenericItemType)
(+) GenericItemType deQueue()

PriorityQueue Class Attributes

CLASS CONSTRUCTORS

(+) PriorityQueue()
(+) PriorityQueue(GenericItemType)
(+) PriorityQueue(ListEntry)
(+) PriorityQueue(JList)
(+) PriorityQueue(Stack)
(+) PriorityQueue(Queue)
(+) PriorityQueue(PriorityQueue)

CHANGE STATE SERVICES

(+) void SortAscending()
(+) void SortDescending()

JAVA SOURCE CODE

```
/**
  @author      Marco Martinez
  @fileName    StringDataItem.java
  @version     1.0
  @description This program will construct and manipulate StringDataItem objects.

  Classes
    GenericItemType
    IntegerDataType
    StringDataType
    GenericContainer
    AppDriver

  Associations
    IntegerDataType --- 1 : 1 (inherits) ---> GenericItempType
    StringDataType --- 1 : 1 (inherits) ---> GenericItemType
    GenericContainer --- 1 : m (contains) ---> GenericItemType
    AppDriver --- 1 : 1 (uses) ---> GenericContainer

  StringDataItem
    INSTANCE VARIABLE DECLARATION
      (-) String myString;

    CLASS CONSTRUCTORS
      (+) StringDataItem()
      (+) StringDataItem(String s)
      (+) StringDataItem(StringDataItem sdi)

    CHANGE STATE SERVICES
      (+) void set(String s)

    READ STATE SERVICES
      (+) boolean isLess(GenericItemType git)
      (+) boolean isEqual(GenericItemType git)
      (+) boolean isGreater(GenericItemType git)
      (+) String get()
      (+) String toString()

  @date        12/12/2018
```

```java
  Program Change Log
  ==========================
  Name    Date    Description
  Marco   12/12   Create baseline for StringDataItem.
 */

public class StringDataItem extends GenericItemType
{
  // INSTANCE VARIABLE DECLARATION
  private String myString;

  // CLASS CONSTRUCTORS
  // (+) StringDataItem()
  public StringDataItem(){}

  // (+) StringDataItem(String s)
  public StringDataItem(String s)
  {
    myString = new String(s);
  }

  // (+) StringDataItem(StringDataItem sdi)
  public StringDataItem(StringDataItem sdi) { set(sdi.get()); }

  // CHANGE STATE SERVICES
  // (+) void set(String s)
  public void set(String s)
  {
    myString = s;
  }

  // READ STATE SERVICES
  // (+) boolean isLess(GenericItemType git)
  public boolean isLess(GenericItemType git)
  {
    return ( myString.compareTo(((StringDataItem) git).get()) < 0);
  }

  // (+) boolean isEqual(GenericItemType git)
  public boolean isEqual(GenericItemType git)
  {
    return ( myString.compareTo(((StringDataItem) git).get()) == 0);
  }

  // (+) boolean isGreater(GenericItemType git)
  public boolean isGreater(GenericItemType git)
  {
    return ( myString.compareTo(((StringDataItem) git).get()) > 0);
  }

  // (+) String get()
  public String get() { return myString; }

  // (+) String toString()
  public String toString() { return " " + myString; }
}
```

```java
/**
  @author       Marco Martinez
  @fileName     GenericItemType.java
  @version      1.0
  @description  Used in Container class as the "only" data type.
  @date         12/18/2018

  Program Change Log
  ==========================
  Name     Date     Description
  Marco    12/18    Create baseline for GenericItemType.
 */

public abstract class GenericItemType {

  // (+) abstract boolean isLess(GenericItemType git)
  public abstract boolean isLess(GenericItemType git);

  // (+) abstract boolean isEqual(GenericItemType git)
  public abstract boolean isEqual(GenericItemType git);

  // (+) abstract boolean isGreater(GenericItemType git)
  public abstract boolean isGreater(GenericItemType git);
}

/**
 @author       Marco Martinez
 @fileName     ListEntry.java
 @version      1.0
 @description  Used in List Container with references to next and previous for bidirectional.
 @date         2/20/2018

 Program Change Log
 ========================
 Name     Date     Description
 Marco    2/20     Create baseline for ListEntry.
 */

public class ListEntry {
   // (+) INSTANCE VARIABLE DECLARATION
   GenericItemType data;
   ListEntry       next,
                   prev;

   // CLASS CONSTRUCTORS
   // (+) ListEntry()
   public ListEntry() {
      this.data = null;
      this.next = null;
      this.prev = null;
   }

   // (+) ListEntry(GenericItemType data)
   public ListEntry(GenericItemType data) {
      this.data = data;
      this.next = null;
      this.prev = null;
   }
```

```java
    // (+) ListEntry(ListEntry le)
    public ListEntry(ListEntry le) {
        this.data = le.getData();
        this.next = le.getNext();
        this.prev = le.getPrev();
    }

    // CHANGE STATE SERVICES
    // (+) void setData(GenericItemType data)
    public void setData(GenericItemType data) {
        this.data = data;
    }

    // (+) void setNext(ListEntry next)
    public void setNext(ListEntry next) {
        if (next != null)
            this.next = next;
        else
            this.next = null;
    }

    // (+) void setPrev(ListEntry prev)
    public void setPrev(ListEntry prev) {
        if (prev != null)
            this.prev = prev;
        else
            this.prev = null;
    }

    // READ STATE SERVICES
    // (+) GenericItemType getData()
    public GenericItemType getData() {
        return this.data;
    }

    // (+) ListEntry getNext()
    public ListEntry getNext() {
        return this.next;
    }

    // (+) ListEntry getPrev()
    public ListEntry getPrev() {
        return this.prev;
    }
}

/**
 @author        Marco Martinez
 @fileName      JList.java
 @version       1.0
 @description   Used as pointer based container with "standard" functionality.
 @date          2/20/2018

 Program Change Log
 ==========================
 Name      Date      Description
 Marco     2/20      Create baseline for JList.
 */

public class JList {
    // INSTANCE VARIABLE DECLARATIONS
```

```
    ListEntry   head,
                tail,
                currentIteration;
    int         totalCount,
                currentCount;

    // CLASS CONSTRUCTORS
    // (+) JList()
    public JList() {
        this.head = this.tail = this.currentIteration = null;
        this.currentCount = this.totalCount = 0;
    }

    // (+) JList(GenericItemType data)
    public JList(GenericItemType data) {
        if (data != null) {
            this.head = new ListEntry(data);
            this.head.setNext(null);
            this.head.setPrev(null);
            this.currentIteration = null;
            this.tail = this.head;
            this.totalCount = 1;
            this.currentCount = 0;

        } else {
            this.head = this.tail = this.currentIteration = null;
            this.currentCount = this.totalCount = 0;
        }
    }

    // (+) JList(ListEntry le)
    public JList(ListEntry le) {
        if (le.getData() != null) {
            this.totalCount = 1;
            this.currentCount = 0;
            this.head = this.tail = this.currentIteration = le;
            while (this.currentIteration.getNext() != null) {
                this.currentIteration = this.currentIteration.getNext();
                this.totalCount++;
            }
            this.tail = this.currentIteration;
        } else {
            this.head = this.tail = this.currentIteration = null;
            this.currentCount = this.totalCount = 0;
        }
    }

    // (+) JList(JList l)
    public JList(JList l) {
        this.head = l.getStart();
        this.tail = l.tail;
        this.totalCount = l.getCount();
    }

    // (+) JList(Stack s)
    public JList(Stack s) {
        this.head = s.getStart();
        this.tail = s.tail;
        this.totalCount = s.getCount();
    }

    // (+) JList(Queue q)
    public JList(Queue q) {
        this.head = q.getStart();
        this.tail = q.tail;
        this.totalCount = q.getCount();
    }
```

```java
    // (+) JList(PriorityQueue q)
    public JList(PriorityQueue q) {
        this.head = q.getStart();
        this.tail = q.tail;
        this.totalCount = q.getCount();
    }

    // CHANGE STATE SERVICES
    // (+) void init()
    public void init() {
        this.head = this.tail = this.currentIteration = null;
        this.currentCount = this.totalCount = 0;
    }

    // (+) void add_fromHead(GenericItemType git)
    public void add_fromHead(GenericItemType git) {
        if (this.isFull())
            return;

        if (git != null) {
            if (!this.isEmpty()) {
                this.head.setPrev(new ListEntry(git));
                this.head.getPrev().setNext(this.head);
                this.head = this.head.getPrev();
            } else {
                this.head = this.tail = new ListEntry(git);
                this.head.setPrev(null);
                this.head.setNext(null);
            }
            this.totalCount++;
        }
    }

    // (+) void add_fromMid(GenericItemType git)
    public void add_fromMid(GenericItemType git) {
        if (this.isFull())
            return;

        if (git != null) {
            if (!this.isEmpty()) {
                int mid = this.totalCount / 2;
                this.currentIteration = head;
                for (int i = 0; i < mid-1; i++) {
                    this.currentIteration = this.currentIteration.getNext();
                }
                ListEntry temp = this.currentIteration;
                this.currentIteration = new ListEntry(git);
                this.currentIteration.setPrev(temp);
                this.currentIteration.setNext(temp.getNext());
                temp.setNext(this.currentIteration);
                temp = this.currentIteration.getNext();
                temp.setPrev(this.currentIteration);
            } else {
                this.head = this.tail = new ListEntry(git);
                this.head.setNext(null);
                this.head.setPrev(null);
            }
            this.totalCount++;
        }

    }

    // (+) void add_fromTail(GenericItemType git)
    public void add_fromTail(GenericItemType git) {
        if (this.isFull())
            return;

        if (git != null) {
```

```
            if (!this.isEmpty()) {
                this.tail.setNext(new ListEntry(git));
                this.tail.getNext().setPrev(this.tail);
                this.tail = this.tail.getNext();
            } else {
                this.head = this.tail = new ListEntry(git);
                this.head.setPrev(null);
                this.head.setNext(null);
            }
            this.totalCount++;
        }
    }

    // (+) void add_fromHead(ListEntry le)
    public void add_fromHead(ListEntry le) {
        if (this.isFull())
            return;

        if (le.getData() != null) {
            if (!this.isEmpty()) {
                this.head.setPrev(new ListEntry(le.getData()));
                this.head.getPrev().setNext(this.head);
                this.head = this.head.getPrev();
            } else {
                this.head = this.tail =  new ListEntry(le.getData());
            }
            this.totalCount++;
        }
    }

    // (+) void add_fromMid(ListEntry le)
    public void add_fromMid(ListEntry le) {
        if (this.isFull())
            return;

        if (le.getData() != null) {
            if (!this.isEmpty()) {
                int mid = this.totalCount / 2;
                this.currentIteration = head;
                for (int i = 0; i < mid-1; i++) {
                    this.currentIteration = this.currentIteration.getNext();
                }
                ListEntry temp = this.currentIteration.getNext();
                this.currentIteration.setNext(new ListEntry(le.getData()));
                this.currentIteration.getNext().setPrev(this.currentIteration);
                temp.setPrev(this.currentIteration.getNext());
                temp.getPrev().setNext(temp);

            } else {
                this.head = this.tail =  new ListEntry(le.getData());
            }
            this.totalCount++;
        }

    }

    // (+) void add_fromTail(ListEntry le)
    public void add_fromTail(ListEntry le) {
        if (this.isFull())
            return;

        if (le.getData() != null) {
            if (!this.isEmpty()) {
                this.tail.setNext(new ListEntry(le.getData()));
                this.tail.getNext().setPrev(this.tail);
                this.tail = this.tail.getNext();
            } else {
                this.head = this.tail =  new ListEntry(le.getData());
```

```
            }
        }
        this.totalCount++;
    }

    // (+) void bubbleSort_ascending()
    public void bubbleSort_ascending() {
        this.currentIteration = this.head;

        for (int outer = 0; outer < this.totalCount; outer++) {
            for (int inner = 0; inner < this.totalCount-1; inner++) {
                if (this.currentIteration.getData().isGreater(this.currentIteration.getNext().getData())) {
                    GenericItemType temp = this.currentIteration.getData();
                    this.currentIteration.setData(this.currentIteration.getNext().getData());
                    this.currentIteration.getNext().setData(temp);
                }
                this.currentIteration = this.currentIteration.getNext();
            }
            this.currentIteration = this.head;
        }
    }

    // (+) void bubbleSort_descending()
    public void bubbleSort_descending() {
        this.currentIteration = this.head;

        for (int outer = 0; outer < this.totalCount; outer++) {
            for (int inner = 0; inner < this.totalCount-1; inner++) {
                if (this.currentIteration.getData().isLess(this.currentIteration.getNext().getData())) {
                    GenericItemType temp = this.currentIteration.getData();
                    this.currentIteration.setData(this.currentIteration.getNext().getData());
                    this.currentIteration.getNext().setData(temp);
                }
                this.currentIteration = this.currentIteration.getNext();
            }
            this.currentIteration = this.head;
        }
    }

    // (+) GenericItemType linearSearch(GenericItemType key)
    public GenericItemType linearSearch(GenericItemType key) { return new
ListEntry(this.lSearch(key)).getData(); }

    // (+) GenericItemType linearSearch(ListEntry key)
    public GenericItemType linearSearch(ListEntry key) { return new
ListEntry(this.lSearch(key.getData())).getData(); }

    //  (-) ListEntry lSearch(GenericItemType key)
    private ListEntry lSearch(GenericItemType key) {
        this.currentCount = 0;
        this.currentIteration = this.head;
        for (int i = 0; i < this.totalCount; i++) {
            if (this.currentIteration.getData().isEqual(key)) {
                return this.currentIteration;
            }
            this.currentIteration = this.currentIteration.getNext();
            this.currentCount++;
        }
        this.currentCount = 0;
        return new ListEntry();
    }

    // (+) void remove(GenericItemType key)
    public void remove(GenericItemType key) { this.delete(key); }

    // (+) void remove(ListEntry key)
    public void remove(ListEntry key) { this.delete(key.getData()); }
```

```java
//  (-) void delete(GenericItemType key)
private void delete(GenericItemType key) {
    this.currentIteration = this.lSearch(key);
    if (this.currentIteration != null) {
        this.currentIteration.setData(this.tail.getData());
        this.tail = this.tail.getPrev();
        this.tail.setNext(null);
        this.totalCount--;
    }
    bubbleSort_ascending();
}

// (+) void reverseList()
public void reverseList() {
    JList temp = new JList();
    this.currentIteration = this.tail;
    for (int i = 0; i < this.totalCount; i++) {
        temp.add_fromTail(this.currentIteration.getData());
        this.currentIteration = this.currentIteration.getPrev();
    }
    this.head = temp.getStart();
    this.tail = temp.getEnd();
    this.totalCount = temp.getCount();
}

// READ STATE SERVICES
// (+) boolean isFull()
public boolean isFull() {
    if (this.isEmpty())
        return false;

    try {
        this.tail.setNext(new ListEntry(this.tail));
        this.tail.getNext().setPrev(this.tail);
        this.tail = this.tail.getNext();
        this.tail = this.tail.getPrev();
        this.tail.setNext(null);
        return false;
    } catch (OutOfMemoryError e) {
        return true;
    }
}

// (+) boolean isEmpty()
public boolean isEmpty() { return this.head == null; }

// (+) int getCount()
public int getCount() { return this.totalCount; }

// (+) ListEntry getStart()
public ListEntry getStart() { return this.head; }

// (+) ListEntry getEnd()
public ListEntry getEnd() { return this.tail; }

// (+) void Iterator_initialize()
public void Iterator_initialize() {
    this.currentCount = 0;
    this.currentIteration = this.head;
}

// (+) boolean Iterator_hasNext()
public boolean Iterator_hasNext() {
    if (this.currentCount < this.totalCount)
        return true;
    return false;
}
```

```
    // (+) GenericItemType Iterator_iterate()
    public GenericItemType Iterator_iterate() {
        if (this.currentCount < this.totalCount) {
            if (this.currentCount != 0)
                this.currentIteration = this.currentIteration.getNext();
            else {
                this.currentIteration = this.head;
            }
            this.currentCount++;
            return this.currentIteration.getData();
        }
        return new ListEntry().getData();
    }
}

/**
 @author      Marco Martinez
 @fileName    Stack.java
 @version     1.0
 @description This is an extension of JList to allow for Stack operations (push, pop, show top).
 @date        2/27/2018

 Program Change Log
 ==========================
 Name    Date    Description
 Marco   2/27    Create baseline for Stack.
 */

public class Stack extends JList {
   // CLASS CONSTRUCTORS
   // (+) Stack()
   public Stack() { super(); }

   // (+) Stack(GenericItemType git)
   public Stack(GenericItemType git) { super(git); }

   // (+) Stack(ListEntry le)
   public Stack(ListEntry le) { super(le); }

   // (+) Stack(JList jl)
   public Stack(JList jl) { super(jl); }

   // (+) Stack(Stack s)
   public Stack(Stack s) { super(s); }

   // (+) Stack(Queue q)
   public Stack(Queue q) { super(q); }

   // (+) Stack(PriorityQueue q)
   public Stack(PriorityQueue q) { super(q); }

   // CHANGE STATE SERVICES
   // (+) void push(GenericItemType git)
   public void push(GenericItemType git) { this.add_fromHead(git); }

   // (+) GenericItemType pop()
   public GenericItemType pop() {
      GenericItemType temp = this.head.getData();
      this.head = this.head.getNext();
      this.head.setPrev(null);
      this.totalCount--;
```

```
        return temp;
    }

    // READ STATE SERVICES
    // (+) GenericItemType showTop()
    public GenericItemType showTop() {
        return this.head.getData();
    }
}

/**
 @author       Marco Martinez
 @fileName     Queue.java
 @version      1.0
 @description  This is an extension of JList to allow for Queue operations (enQueue,deQueue).
 @date         2/27/2018

 Program Change Log
 ==========================
 Name     Date     Description
 Marco    2/27     Create baseline for Queue.
 */

public class Queue extends JList {
    // CLASS CONSTRUCTORS
    // (+) Queue()
    public Queue() { super(); }

    // (+) Queue(GenericItemType git)
    public Queue(GenericItemType git) { super(git); }

    // (+) Queue(ListEntry le)
    public Queue(ListEntry le) { super(le); }

    // (+) Queue(JList jl)
    public Queue(JList jl) { super(jl); }

    // (+) Queue(Stack s)
    public Queue(Stack s) { super(s); }

    // (+) Queue(Queue q)
    public Queue(Queue q) {
        super(q);
    }

    // (+) Queue(PriorityQueue q)
    public Queue(PriorityQueue q) { super(q); }

    // CHANGE STATE SERVICES
    // (+) void enQueue(GenericItemType git)
    public void enQueue(GenericItemType git) { this.add_fromHead(git); }

    // (+) GenericItemType deQueue()
    public GenericItemType deQueue() {
        GenericItemType temp = this.tail.getData();
        this.tail = this.tail.getPrev();
        this.tail.setNext(null);
        this.totalCount--;
        return temp;
```

```
    }
}

/**
 @author      Marco Martinez
 @fileName    PriorityQueue.java
 @version     1.0
 @description This is an extension of JList to allow for PriorityQueue for sorting6.
 @date        2/27/2018

 Program Change Log
 ==========================
 Name    Date    Description
 Marco   2/27    Create baseline for PriorityQueue.
 */

public class PriorityQueue extends Queue {
    // CLASS CONSTRUCTORS
    // (+) PriorityQueue()
    public PriorityQueue() { super(); }

    // (+) PriorityQueue(GenericItemType git)
    public PriorityQueue(GenericItemType git) { super(git); }

    // (+) PriorityQueue(ListEntry le)
    public PriorityQueue(ListEntry le) { super(le); }

    // (+) PriorityQueue(JList jl)
    public PriorityQueue(JList jl) { super(jl); }

    // (+) PriorityQueue(Stack s)
    public PriorityQueue(Stack s) { super(s); }

    // (+) PriorityQueue(Queue q)
    public PriorityQueue(Queue q) {
        super(q);
    }

    // (+) PriorityQueue(PriorityQueue q)
    public PriorityQueue(PriorityQueue q) {
        super(q);
    }

    // CHANGE STATE SERVICES
    // (+) void SortAscending()
    public void SortAscending() { this.bubbleSort_ascending(); }

    // (+) void SortDescending()
    public void SortDescending() { this.bubbleSort_descending(); }
}

/**
 @author       Marco Martinez
 @fileName     Main.java
 @version      1.0
 @description  Tests JList, ListEntry, Stack, Queue, PriorityQueue.
 @date         2/27/2018

 Program Change Log
 ==========================
```

```
 Name     Date     Description
 Marco    2/27     Create baseline for Main.
 */

import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

public class Main {

    public static void main(String[] args) {
        try {


            // TEST VARIABLES
            OutputStream file = new FileOutputStream("REPORT.txt");
            String contents = new String();
            byte buffer[] = new byte[4096];
            JList myList = new JList();
            StringDataItem firstS = new StringDataItem("first");
            StringDataItem secondS = new StringDataItem("second");
            StringDataItem thirdS = new StringDataItem("third");
            StringDataItem fourthS = new StringDataItem("fourth");
            ListEntry fifthLE = new ListEntry(new StringDataItem("fifth"));
            StringDataItem FIRSTBREAK = new StringDataItem("FIRSTBREAK");
            StringDataItem MIDBREAK = new StringDataItem("MIDBREAK");
            myList.add_fromTail(firstS);
            myList.add_fromTail(secondS);
            myList.add_fromTail(thirdS);
            myList.add_fromTail(fourthS);
            myList.add_fromTail(fifthLE);
            myList.add_fromHead(FIRSTBREAK);
            myList.add_fromMid(MIDBREAK);

            // TESTING CALLS
            contents += "\n";
            contents += "Original order for myList: \n";
            myList.Iterator_initialize();
            while (myList.Iterator_hasNext()) {
                contents += (myList.Iterator_iterate()).toString() + "\n";
            }
            contents += "Current number of elements within myList: " + myList.getCount() + "\n";

            JList newList1 = new JList(myList);
            contents += "\n";
            contents += "Copy Constructor: \n";
            newList1.Iterator_initialize();
            while (newList1.Iterator_hasNext()) {
                contents += (newList1.Iterator_iterate()).toString() + "\n";
            }
            contents += "Current number of elements within newList1: " + newList1.getCount() + "\n";

            JList newList3 = new JList(myList.getStart().getData());
            contents += "\n";
            contents += "GenericItemType Constructor with data from head of myList: \n";
            newList3.Iterator_initialize();
            while (newList3.Iterator_hasNext()) {
                contents += (newList3.Iterator_iterate()).toString() + "\n";
            }
            contents += "Current number of elements within newList: " + newList3.getCount() + "\n";

            JList newList2 = new JList(myList.getStart());
            contents += "\n";
            contents += "List Entry Constructor with Head of myList: \n";
            newList2.Iterator_initialize();
            while (newList2.Iterator_hasNext()) {
                contents += (newList2.Iterator_iterate()).toString() + "\n";
            }
```

```
            contents += "Current number of elements within newList: " + newList2.getCount() + "\n";

            contents += "\n";
            contents += "Adding 'hello' as GIT to the head of myList: \n";
            myList.add_fromHead(new StringDataItem("hello"));
            myList.Iterator_initialize();
            while (myList.Iterator_hasNext()) {
                contents += (myList.Iterator_iterate()).toString() + "\n";
            }
            contents += "Current number of elements within myList: " + myList.getCount() + "\n";

            contents += "\n";
            contents += "Adding 'begin' as ListEntry to the head of myList: \n";
            myList.add_fromHead(new ListEntry(new StringDataItem("begin")));
            myList.Iterator_initialize();
            while (myList.Iterator_hasNext()) {
                contents += (myList.Iterator_iterate()).toString() + "\n";
            }
            contents += "Current number of elements within myList: " + myList.getCount() + "\n";

            contents += "\n";
            contents += "Adding 'midpoint' as GIT to the middle of myList: \n";
            myList.add_fromMid(new StringDataItem("midpoint"));
            myList.Iterator_initialize();
            while (myList.Iterator_hasNext()) {
                contents += (myList.Iterator_iterate()).toString() + "\n";
            }
            contents += "Current number of elements within myList: " + myList.getCount() + "\n";

            contents += "\n";
            contents += "Adding 'anothermid' as ListEntry to the middle of myList: \n";
            myList.add_fromMid(new ListEntry(new StringDataItem("anothermid")));
            myList.Iterator_initialize();
            while (myList.Iterator_hasNext()) {
                contents += (myList.Iterator_iterate()).toString() + "\n";
            }
            contents += "Current number of elements within myList: " + myList.getCount() + "\n";

            contents += "\n";
            contents += "Adding 'end' as GIT to the tail of myList: \n";
            myList.add_fromTail(new StringDataItem("end"));
            myList.Iterator_initialize();
            while (myList.Iterator_hasNext()) {
                contents += (myList.Iterator_iterate()).toString() + "\n";
            }
            contents += "Current number of elements within myList: " + myList.getCount() + "\n";

            contents += "\n";
            contents += "Adding 'anotherend' as GIT to the tail of myList: \n";
            myList.add_fromHead(new ListEntry(new StringDataItem("anotherend")));
            myList.Iterator_initialize();
            while (myList.Iterator_hasNext()) {
                contents += (myList.Iterator_iterate()).toString() + "\n";
            }
            contents += "Current number of elements within myList: " + myList.getCount() + "\n";

            contents += "\n";
            contents += "Delete FIRSTBREAK as GIT and MIDBREAK as ListEntry from myList: \n";
            myList.remove(new StringDataItem("FIRSTBREAK"));
            myList.remove(new ListEntry(new StringDataItem("MIDBREAK")));
            myList.Iterator_initialize();
            while (myList.Iterator_hasNext()) {
                contents += (myList.Iterator_iterate()).toString() + "\n";
            }
            contents += "Current number of elements within myList: " + myList.getCount() + "\n";

            contents += "\n";
            contents += "Ascending sorted order of myList: \n";
```

```
            myList.bubbleSort_ascending();
            myList.Iterator_initialize();
            while (myList.Iterator_hasNext()) {
                contents += (myList.Iterator_iterate()).toString() + "\n";
            }
            contents += "Current number of elements within myList: " + myList.getCount() + "\n";

            contents += "\n";
            contents += "Descending sorted order of myList: \n";
            myList.bubbleSort_descending();
            myList.Iterator_initialize();
            while (myList.Iterator_hasNext()) {
                contents += (myList.Iterator_iterate()).toString() + "\n";
            }
            contents += "Current number of elements within myList: " + myList.getCount() + "\n";

            contents += "\n";
            contents += "Reversing myList: \n";
            myList.reverseList();
            myList.Iterator_initialize();
            while (myList.Iterator_hasNext()) {
                contents += (myList.Iterator_iterate()).toString() + "\n";
            }
            contents += "Current number of elements within myList: " + myList.getCount() + "\n";

            Stack myStack = new Stack(myList);
            contents += "\n";
            contents += "Converting myList to Stack \n";
            myStack.Iterator_initialize();
            while (myStack.Iterator_hasNext()) {
                contents += (myStack.Iterator_iterate()).toString() + "\n";
            }
            contents += "Current number of elements within myList: " + myStack.getCount() + "\n";

            contents += "\n";
            contents += "Pop stack \n";
            myStack.pop();
            myStack.Iterator_initialize();
            while (myStack.Iterator_hasNext()) {
                contents += (myStack.Iterator_iterate()).toString() + "\n";
            }
            contents += "Current number of elements within myList: " + myStack.getCount() + "\n";

            contents += "\n";
            contents += "Push 'push' as git into Stack \n";
            myStack.push(new StringDataItem("push"));
            myStack.Iterator_initialize();
            while (myStack.Iterator_hasNext()) {
                contents += (myStack.Iterator_iterate()).toString() + "\n";
            }
            contents += "Current number of elements within myList: " + myStack.getCount() + "\n";

            Queue myQueue = new Queue(myStack);
            contents += "\n";
            contents += "Converting myStack to myQueue: \n";
            myQueue.Iterator_initialize();
            while (myQueue.Iterator_hasNext()) {
                contents += (myQueue.Iterator_iterate()).toString() + "\n";
            }
            contents += "Current number of elements within myList: " + myQueue.getCount() + "\n";

            contents += "\n";
            contents += "Perform enQueue to add 'enqueue' as GIT to myQueue \n";
            myQueue.enQueue(new StringDataItem("enqueue"));
            myQueue.Iterator_initialize();
            while (myQueue.Iterator_hasNext()) {
                contents += (myQueue.Iterator_iterate()).toString() + "\n";
            }
```

```
            contents += "Current number of elements within myList: " + myQueue.getCount() + "\n";

            contents += "\n";
            contents += "Perform deQueue to remove from bottom of myQueue \n";
            myQueue.deQueue();
            myQueue.Iterator_initialize();
            while (myQueue.Iterator_hasNext()) {
                contents += (myQueue.Iterator_iterate()).toString() + "\n";
            }
            contents += "Current number of elements within myList: " + myQueue.getCount() + "\n";

            PriorityQueue myPrioQueue = new PriorityQueue(myQueue);
            contents += "\n";
            contents += "Converting myQueue to myPrioQueue: \n";
            myPrioQueue.Iterator_initialize();
            while (myPrioQueue.Iterator_hasNext()) {
                contents += (myPrioQueue.Iterator_iterate()).toString() + "\n";
            }
            contents += "Current number of elements within myList: " + myPrioQueue.getCount() + "\n";

            contents += "\n";
            contents += "Sort myPrioQueue in ascending order: \n";
            myPrioQueue.SortAscending();
            myPrioQueue.Iterator_initialize();
            while (myPrioQueue.Iterator_hasNext()) {
                contents += (myPrioQueue.Iterator_iterate()).toString() + "\n";
            }
            contents += "Current number of elements within myList: " + myPrioQueue.getCount() + "\n";

            contents += "\n";
            contents += "Sort myPrioQueue in descending order: \n";
            myPrioQueue.SortDescending();
            myPrioQueue.Iterator_initialize();
            while (myPrioQueue.Iterator_hasNext()) {
                contents += (myPrioQueue.Iterator_iterate()).toString() + "\n";
            }
            contents += "Current number of elements within myList: " + myPrioQueue.getCount() + "\n";

            contents += "\nChecking for OutOfMemory error...\n";
            while (!myList.isFull()) {
                myList.add_fromTail(new StringDataItem("test"));
            }
            myList = null;
            contents += "OutOfMemory error successfully caught. Closing file...\n";

            buffer = contents.getBytes();
            file.write(buffer);
            file.close();
        } catch (IOException e ) {
            System.err.println("Error: " + e.getMessage());
        }
    }
}
```

REPORT

Original order for myList:
FIRSTBREAK
first
second
MIDBREAK
third

fourth
fifth
Current number of elements within myList: 7

Copy Constructor:
FIRSTBREAK
first
second
MIDBREAK
third
fourth
fifth
Current number of elements within newList1: 7

GenericItemType Constructor with data from head of myList:
FIRSTBREAK
Current number of elements within newList: 1

List Entry Constructor with Head of myList:
FIRSTBREAK
first
second
MIDBREAK
third
fourth
fifth
Current number of elements within newList: 7

Adding 'hello' as GIT to the head of myList:
hello
FIRSTBREAK
first
second
MIDBREAK
third
fourth
fifth
Current number of elements within myList: 8

Adding 'begin' as ListEntry to the head of myList:
begin
hello
FIRSTBREAK
first
second
MIDBREAK
third
fourth
fifth

Current number of elements within myList: 9

Adding 'midpoint' as GIT to the middle of myList:
begin
hello
FIRSTBREAK
first
midpoint
second
MIDBREAK
third
fourth
fifth
Current number of elements within myList: 10

Adding 'anothermid' as ListEntry to the middle of myList:
begin
hello
FIRSTBREAK
first
midpoint
anothermid
second
MIDBREAK
third
fourth
fifth
Current number of elements within myList: 11

Adding 'end' as GIT to the tail of myList:
begin
hello
FIRSTBREAK
first
midpoint
anothermid
second
MIDBREAK
third
fourth
fifth
end
Current number of elements within myList: 12

Adding 'anotherend' as GIT to the tail of myList:
anotherend
begin
hello
FIRSTBREAK

first
midpoint
anothermid
second
MIDBREAK
third
fourth
fifth
end
Current number of elements within myList: 13

Delete FIRSTBREAK as GIT and MIDBREAK as ListEntry from myList:
anotherend
anothermid
begin
end
fifth
first
fourth
hello
midpoint
second
third
Current number of elements within myList: 11

Ascending sorted order of myList:
anotherend
anothermid
begin
end
fifth
first
fourth
hello
midpoint
second
third
Current number of elements within myList: 11

Descending sorted order of myList:
third
second
midpoint
hello
fourth
first
fifth
end
begin

anothermid
anotherend
Current number of elements within myList: 11

Reversing myList:
anotherend
anothermid
begin
end
fifth
first
fourth
hello
midpoint
second
third
Current number of elements within myList: 11

Converting myList to Stack
anotherend
anothermid
begin
end
fifth
first
fourth
hello
midpoint
second
third
Current number of elements within myList: 11

Pop stack
anothermid
begin
end
fifth
first
fourth
hello
midpoint
second
third
Current number of elements within myList: 10

Push 'push' as git into Stack
push
anothermid
begin

end
fifth
first
fourth
hello
midpoint
second
third
Current number of elements within myList: 11

Converting myStack to myQueue:
push
anothermid
begin
end
fifth
first
fourth
hello
midpoint
second
third
Current number of elements within myList: 11

Perform enQueue to add 'enqueue' as GIT to myQueue
enqueue
push
anothermid
begin
end
fifth
first
fourth
hello
midpoint
second
third
Current number of elements within myList: 12

Perform deQueue to remove from bottom of myQueue
enqueue
push
anothermid
begin
end
fifth
first
fourth
hello

midpoint
second
Current number of elements within myList: 11

Converting myQueue to myPrioQueue:
enqueue
push
anothermid
begin
end
fifth
first
fourth
hello
midpoint
second
Current number of elements within myList: 11

Sort myPrioQueue in ascending order:
anothermid
begin
end
enqueue
fifth
first
fourth
hello
midpoint
push
second
Current number of elements within myList: 11

Sort myPrioQueue in descending order:
second
push
midpoint
hello
fourth
first
fifth
enqueue
end
begin
anothermid
Current number of elements within myList: 11

Checking for OutOfMemory error...
OutOfMemory error successfully caught. Closing file...