

INFO-C307 Laboratory Session #5

LAB GOALS

- Generic Classes (also known as Template classes in C++)

Step 1: Start with a simple project called Lab5.

Step 2: Add the following Generic class to your program. Note the syntax for generic classes highlighted as YELLOW. Generic in Java is similar to templates in C++. Generic allows functions and classes to work with different data types such as (Integer, Character, String, or even user defined types such as people or cars, etc....).

```
class aNode <T> {  
    T data;  
    aNode next;  
  
    aNode(T mydata) { // Constructor  
        data = mydata;  
        next = null;  
    }  
};
```

Note that the “data” in the class has a generic type “T”. In order to create objects of the aNode, we must instantiate the class with a particular “type” of data.

For example you can put the following code in your main() function and compile the code. Note that the compiler will be able to compile and generate aNode that can hold an Integer and another node that can hold a Character without any trouble.

```
aNode<Integer> myNode1;           // Instantiate a generic class with the Integer data type  
myNode1 = new aNode<Integer>(5);  
  
//Similarly we can instantiate the above class with “Characters” or “Strings”, etc.  
aNode<Character> myNode2;        // Instantiate a generic class with the Character data type  
myNode2 = new aNode<Character>('A');
```

Step 3: Now that we have a Generic “aNode” class we can also take the next step and create a Generic “LinkedList” class. Place the following class below your aNode class. Once again note how the Linked List class is instantiated with a Generic data type <T>

```
class linkedList<T> {  
  
    aNode head;           // Head of the linked list  
    int size;  
  
    linkedList() { // Constructor  
        head = null;  
        size = 0;  
    }  
  
    //-----  
    public void insert_at_beginning(T value) {  
        aNode newNode = new aNode(value); // create aNew node
```

```

        newNode.next = head;
        head = newNode;
        size++;
    }
    //-----

    public void insert_at_end(T value) {
        aNode newNode = new aNode(value); // create aNew node
        if (isEmpty()) {
            newNode.next = head;
            head = newNode;
            size++;
        } else {
            //find the last node
            aNode ptr;
            ptr = head;
            while (ptr.next != null) {
                ptr = ptr.next;
            }
            ptr.next = newNode; //add the node to the end
            size++;
        }
    }

    //-----
    public void insert_after(T value, T searchValue) {

        if (isEmpty()) {
            System.out.println("Linked List is empty, no way to insert " + value + " after " + searchValue);
        } else {
            //find the node with searchValue
            aNode ptr;
            boolean found = false;
            ptr = head;
            while (ptr != null && found == false) {
                if (ptr.data == searchValue) {
                    found = true;
                } else {
                    ptr = ptr.next;
                }
            }
            if (ptr == null) {
                System.out.println("Did not find " + searchValue + "Nothing Inserted");
            } else {
                aNode newNode = new aNode(value); // create aNew node
                newNode.next = ptr.next;
                ptr.next = newNode; //add the node after the searchValue
                size++;
            }
        }
    }

    //-----
    // Delete the first node with the value
    public void delete(T deleteValue) {
        if (isEmpty()) {
            System.out.println("Linked List is empty, nothing to delete");
        } else {
            aNode deletePtr = head; // create a referene to head

```

```

        if (head.data == deleteValue) {
            head = head.next;    // remove the head
            deletePtr = null;    // make the node available for garbage collection.
            size--;
        } else {
            aNode prevPtr;
            deletePtr = prevPtr = head;
            boolean found = false;

            //find the value to be deleted
            while (deletePtr != null && found == false) {
                if (deletePtr.data == deleteValue) {
                    found = true;
                    prevPtr.next = deletePtr.next;
                    deletePtr = null;    // make deletePtr available to garbage collection
                    size--;
                } else {
                    prevPtr = deletePtr;
                    deletePtr = deletePtr.next;
                }
            }
            if (found == false) {
                System.out.println("Not able to find/delete " + deleteValue + " in the Linked List");
            }
        }
    }
}

//-----
public boolean isEmpty() {
    if (head == null) {
        return (true);
    } else {
        return (false);
    }
}

//-----
public void print() {
    aNode ptr;
    ptr = head;

    System.out.print("Head--> ");
    while (ptr != null) {
        System.out.print(ptr.data + " --> ");
        ptr = ptr.next;
    }
    System.out.println("NULL");
}

//-----
public int getSize() {
    return (size);
}

//-----
public void freeAll() {
    aNode freePtr = head;

```

```

while (head != null) {
    head = head.next;
    // the next two lines are unnecessary, but are included for
    // illustration of how memory is freed up
    //
    freePtr = null;    // make the node available for garbage collector
    freePtr = head;    // now let the freePtr to the new head
}
head = null;
size = 0;

// https://www.geeksforgeeks.org/garbage-collection-java/
// An object is said to be eligible for GC(garbage collection) iff it
// is unreachable.
// There are generally four different ways to make an object eligible for garbage collection.
//1) Nullifying the reference variable
//2) Re-assigning the reference variable
//3) Object created inside method
//4) Island of Isolation
// Once we made object eligible for garbage collection, it may not
// destroy immediately by the garbage collector.
// Whenever JVM runs the Garbage Collector program, then only the
// object will be destroyed. But when JVM runs Garbage Collector,
// we can not expect.
//
// We can also request JVM to run Garbage Collector. There are two
// ways to do it :
// 1) Using System.gc() method : System class contain static
//    method gc() for requesting JVM to run Garbage Collector.
// 2) Using Runtime.getRuntime().gc() method : Runtime class
//    allows the application to interface with the JVM in which
//    the application is running. Hence by using its gc() method,
//    we can request JVM to run Garbage Collector.
//Note :
// There is no guarantee that any one of above two methods will
// definitely run Garbage Collector.
}

};

```

Step 4: Now, place the following code in your main() function in the driver class and instantiate the generic linkedList class, creating two linkedList objects (one linkedList that holds Characters and the other that holds Integers)

```
linkedList<Character> myCharLL = new linkedList<Character>();    // Holds Characters
myCharLL.insert_at_beginning('A');
myCharLL.insert_at_beginning('B');
myCharLL.print();

linkedList<Integer> myIntLL = new linkedList<Integer>();        // Holds Integers
myIntLL.insert_at_beginning(5);
myIntLL.insert_at_beginning(-124);
myIntLL.print();
```

Note: A Generic class is really not a class until it is instantiated with a data type. It is only a definition until is instantiated by the user and given a data type. Once the user instantiates the class with a data type, the proper TYPE is passed to the definition and a COPY of the Generic class is made and added to the code and compiled into the program.

Step 5: On your own. Instantiate the linkedList Class with “Double” data type and test it.

Questions to think about?

- 1) Can we use the same template class to instantiate any arbitrary data type? For example can we place various objects such as images, people or buildings in the LL class? If so, what does the linkedList's print() function should look like for these object?

What to hand in:

Produce a word document include the code as well as screen shot of the output after steps 2, 4, and 5.