Tuesday, March 20

Warmup

Write a listCopy method that takes in a NodeDouble named existing that is the head of a list and returns a NodeDouble that is the head of a separate copy of the list with the same contents in the same order as the existing list. This is *not* a NodeDouble class method.

```java
public static NodeDouble listCopy(NodeDouble existing) {
    NodeDouble copyHead;
    NodeDouble copyTail;

    // handle special case of empty list
    if (existing == null) {
        return null;
    }

    // make the first node
    copyHead = new NodeDouble(existing.getData(), null);
    copyTail = copyHead;

    // copy the rest of the list
    while (existing.getLink() != null) {
        existing = existing.getLink();
        copyTail.setLink(
            new NodeDouble(existing.getData(), null));
        copyTail = copyTail.getLink();
    }

    return copyHead;
}
```

Java Collections

standard containers you don’t need to write
short list of some of the containers

Sequences:   List   LinkedList(list implemented in linked list)
Stack   Queue
Vector
ArrayList
Associative Containers:   SortedSet   HashSet   Map
(key:data pairs)

Generics

All of these containers are independent of the type of element they contain
Use <E> to indicate that there is some element type there
   can refer to E as the type of object
   but E has to be an object type (NOT PRIMITIVE)
Wrapper Classes
Java automatically converts between them (boxing and auto-boxing)
- int becomes Integer
- double becomes Double
- char becomes Character
- boolean becomes Boolean

Primitive types can use standard math operations and are more efficient

The \(<E>\) is the \textit{generic type parameter}
- do NOT use == with objects
- use equals to compare contents
- do use == to check whether a reference is null

Wrapper Classes Using Generic Nodes
- Node\(<E>\)
- NodeDouble becomes Node\(<\text{Double}>\)
- NodeInt becomes Node\(<\text{Integer}>\)
- LinkedSeq\(<\text{Character}>\) \hspace{1em} charSeq = new LinkedSeq\(<\text{Character}>\);
- NodeDouble \hspace{0.5em} list = new NodeDouble(42, null);
- LinkedSeq\(<\text{Double}>\) \hspace{0.5em} list2 = new LinkedSeq\(<\text{Double}>\);

Warmup code for listCopy modified for generics

```java
public static Node\(<E>\) listCopy(Node\(<E>\) existing) {
    Node\(<E>\) copyHead;
    Node\(<E>\) copyTail;

    // handle special case of empty list
    if (existing == null) {
        return null;
    }

    // make the first node
    copyHead = new Node\(<E>\)(existing.getData(), null);
    copyTail = copyHead;

    // copy the rest of the list
    while (existing.getInfo() != null) {
        existing = existing.getInfo();
        copyTail.setInfo(new Node\(<E>\)(existing.getInfo(), null));
        copyTail = copyTail.getInfo();
    }

    return copyHead;
}
```