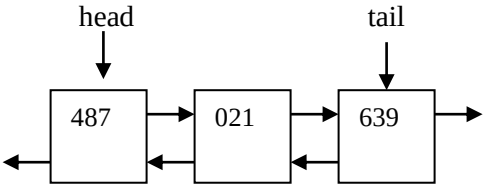
CSC205AB
Assignment 6: InfiniteInt

Background:

Please use the doubly linked list we developed in class (DLList.java). As you conceptualize the program, you will see why a doubly linked list is used.

The Program:

Write a new class called InfiniteInt. We can (theoretically) store an "infinite" integer by linking together nodes that hold actual complete integers. For this program, just store 3 digits in each node (concept is the same, but we won't have to generate hundreds of digits to test it). For example, the integer 487021639 will be stored like this:



As you can see, it uses a doubly linked list; therefore, make your InfiniteInt class a subclass of DLList and define using Generics it so it holds Integers. All of the data (head and tail) are inherited from the superclass. The other methods are also inherited, but you will need to implement the following methods:

- A constructor that receives a String as an argument and builds the linked list. If an InfiniteInt is created as follows

  InfiniteInt myList = new InfiniteInt("487021639");

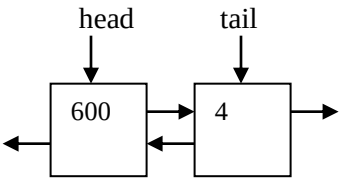
  then the list should be built as shown above.

  This constructor should also check for illegal Strings (containing a non-digit). If encountered, throw an IllegalArgumentException(<message of your choice>).

  Note that you can build the list by using the methods already available from the superclass.

- A constructor that takes no arguments and builds a linked list with that contains a value of 0

- A toString() method that will override the one in the superclass. Your toString() method should return the String representation of the integer with no spaces between the digits. It should also have commas inserted for readability (our example integer should be represented as "487,021,639"). Note that the following InfiniteInt should print as 600,004.



- A static add method which will receive 2 InfiniteInts as arguments, add them up, and return a new InfiniteInt with the total in it. If you think about this and try a few examples on paper, you will see how to do it – you must traverse each number backwards (using the prev link in the doubly linked list) and add each digit, carrying to the next place when necessary. Make sure that you carry correctly and handle the case when one list is longer than the other. This code should work in a driver program:

```
InfiniteInt int1 = new InfiniteInt("646746734");
InfiniteInt int2 = new InfiniteInt("543534");
InfiniteInt int3;
int3 = InfiniteInt.add(int1, int2);
System.out.println(int3);                                    //should print 647,290,268
```

Notice that the add method will create and return an entirely new InfiniteInt. So in the code, it will call InfiniteInt's default constructor; however, that will put a 0 on the new instance. You will have to take off the 0 manually – otherwise you will be stuck with an extra 0 at the end.

- A compareTo(Object o) that will implement the Comparable interface (please actually put "implements Comparable in the class definition). So compareTo will return 1 if the InfiniteInt is greater than what is passed in, -1 if the InfiniteInt is less than what is passed in, and 0 if the InfiniteInt is the same as what is passed in. Refer to the java website for specifications on Comparable and notice that it throws a new ClassCastException if what is passed in is not an InfiniteInt (we also checked the class type in the .equals method). This code should work in a driver program:

```
InfiniteInt int1 = new InfiniteInt("24");
InfiniteInt int2 = new InfiniteInt("6");
InfiniteInt int3 = new InfiniteInt("24");
Integer int4 = new Integer(24);
System.out.println(int1.compareTo(int2));    //should print 1
System.out.println(int2.compareTo(int1));    //should print -1
System.out.println(int1.compareTo(int1));    //should print 0
System.out.println(int1.compareTo(int3));    //should print 0
System.out.println(int1.compareTo(int4));    //throw a new instance of ClassCastException
```

- A reverse toString() method that will also remove all commas from the String. It should take a number such as 123,456 and return 654321. Name the method revAndRemoveToString()
- A replaceCommasWithHyphens() method that replaces the comments generated by the toString() with hyphens and returns the String.

Comments and formatting: Please use the Java conventions for variable names, indenting, and formatting. Each class should have an opening comment which briefly describes the class and includes your name and class on a separate line. Each method should have a short opening comment which describes it. "Sections" of code or parts that are tricky should have comments. See programs from the book for examples (although I prefer that opening and closing "squigglies" be indented the same).

Please submit: your InfiniteInt.java file.