Sample Exam Question: **Huge** Integers

Week 15 – Presentation 1

Huge Integers

In Java a long can have a value that is at most $2^{63}-1$:

9,223,372,036,854,775,807

Yes its big, but might not be big enough for everyone…

Multiples of 103 (polynomial approach of the problem)...

123,456,789,012,345,678,901 can be written as:

123 x 10006

+ 456 x 10005
+ 789 x 10004
+ 12 x 10003
+ 345 x 10002
+ 678 x 10001
+ 901 x 10000

Question

Of the interfaces available in Java collections (List, Queue/Dequeue, Set), to which we can add the Map interface, which one seems to you the most appropriate to store HugeInteger values?

Answer: Which interface??

One value → ~~Map~~

Same number can appear several times → ~~Set~~

A List or Queue is better

using pseudocode…

Question Write (pseudo) code for the two following constructors.

HugeInteger(long intValue)

HugeInteger(String strValue)

The string can contain commas to separate thousands or not. Define exceptions and create specific exceptions if needed, or you may throw any of the following existing exceptions if appropriate:

ArithmeticException ArrayIndexOutOfBoundsException IllegalArgumentException IndexOutOfBoundsException NegativeArraySizeException NullPointerException NumberFormatException StringIndexOutOfBounds UnsupportedOperationException

HugeInteger(long intValue)

```
do:
        append (intvalue % 1000) to the list
        intvalue = intvalue / 1000

while ( intvalue > 1)
```

HugeInteger(String strValue)

```
public HugeInteger(String strValue) throws NumberFormatException {
        String str = strValue.replace(",", "");
        while length of strValue > 3 {
                Take the 3 last characters
                Convert to integer using Integer.ParseInt
                add value to list
                set strValue to substring that excludes the last 3 characters
        }

        if length(strValue) > 0 {
                Convert to integer
                add value to list
        }
```

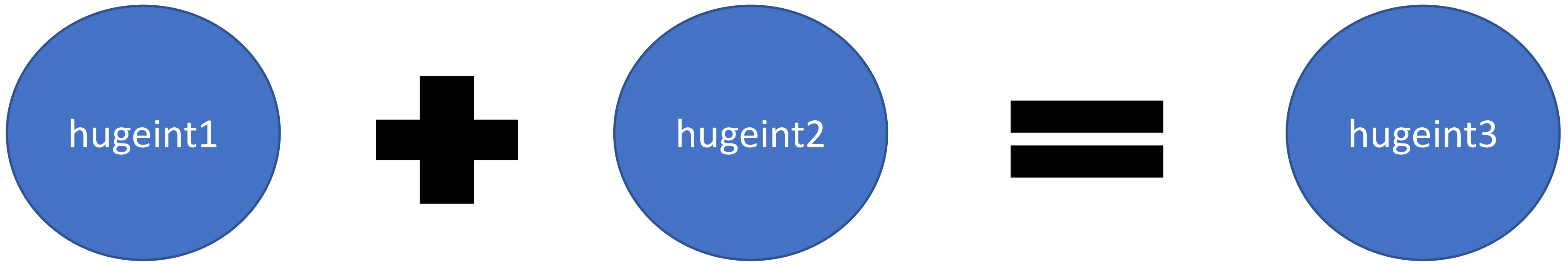
Could also use regular expressions to split/ tokenize the string on ","…

Question

Define an add method to add two HugeInteger objects. Please note that with regular addition the variables that are added are left unmodified (if you have two integer values a and b, the result a + b leaves both a and b unchanged).

What would you prefer – A separate class or a method in the HugeInteger objects? **Justify you preference..**.

Adding 2 HugeInteger objects



No reason to make one of the two objects we are adding more important than the other (would be different with something that would implement a kind of += operation).

A class method makes sense here.

- Give the pseudo code for the method

Pseudo-code

```
HugeInteger result = new HugeInteger();
                                        // Default constructor needed for this

set **min** to the minimum size of the lists in h1 and h2
set **max** to the maximum size of the lists in h1 and h2
int sum;
int val;
int carry = 0;
for (int i = 0; i < **min**; i++) {
        sum = h1.list.get(i) + h2.list.get(i) + carry;
        carry = sum / 1000;
        result.list.add(sum % 1000);
}
```

```
for (int i = min; i < max; i++) {
          if   i >= h1.list.size()   {
                    val = h2.list.get(i);
          } else {
                    val = h1.list.get(i);
          }
          sum = carry + val;
          carry = sum / 1000;
          result.list.add(sum % 1000);
}
if (carry > 0) {
          result.list.add(carry);
}
return result;
```

Question

Use the previously defined method to write the pseudo code for a method that adds a long to a HugeInteger. You can assume it exists even if you didn't manage to write it.

Adding a long to a HugeInteger

```
static HugeInteger add(HugeInteger h, long longVal) {
        return add(h, new HugeInteger(longVal));
}
```

Should also be overloaded with

add(long longVal, HugeInteger h)

to ensure commutativity (ie a*b = b*a).