CS209

MidTerm Review

Huge Integers

In Java, the value of a long can be at most 2^{63}-1, which is
9,223,372,036,854,775,807

Large enough, but not enough for everybody (as a good motivator, in 2009, a project was awarded a US$100,000 prize for first discovering a prime with at least 10 million digits).

Multiples of 10^3 (polynomial approach of the problem)...

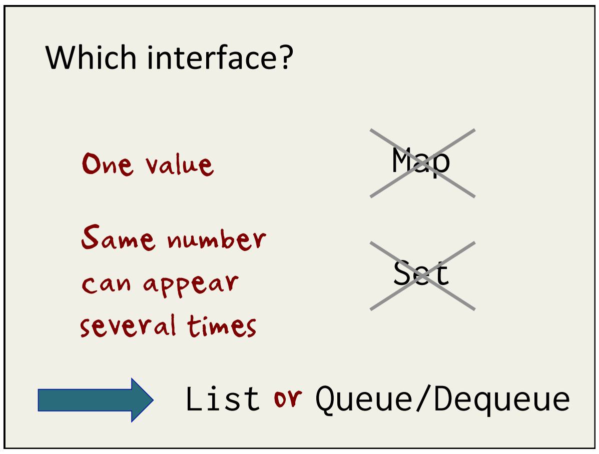
```
123,456,789,012,345,678,901
```

Can be seen as

```
  123 x 1000⁶
+ 456 x 1000⁵
+ 789 x 1000⁴
+  12 x 1000³
+ 345 x 1000²
+ 678 x 1000¹
+ 901 x 1000⁰
```

We'll assume in what follows that the numbers we are dealing with are non-negative (no need to worry about the sign).

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?

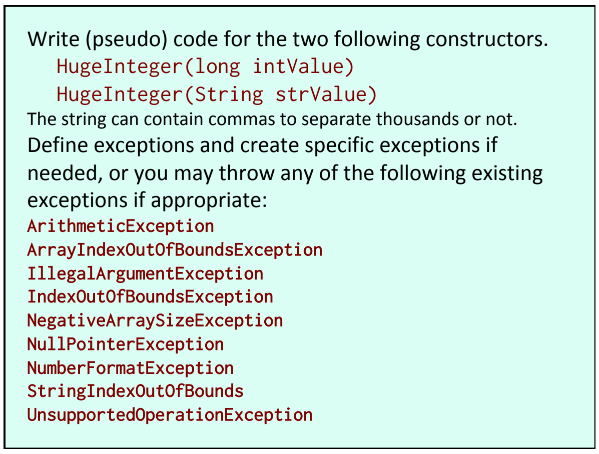


Which interface?

One value ~~Map~~

Same number can appear several times ~~Set~~

List or Queue/Dequeue



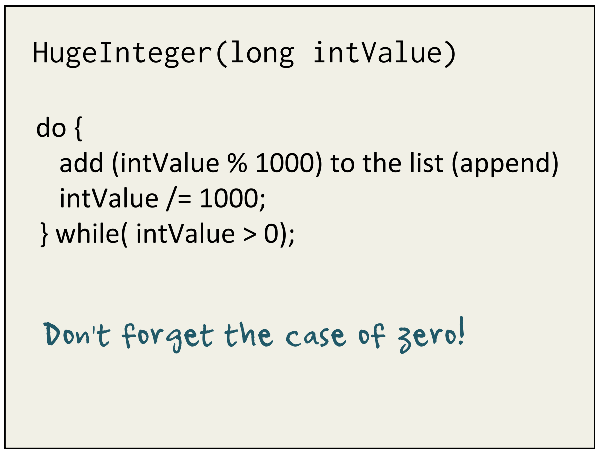
Write (pseudo) code for the two following constructors.

`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:

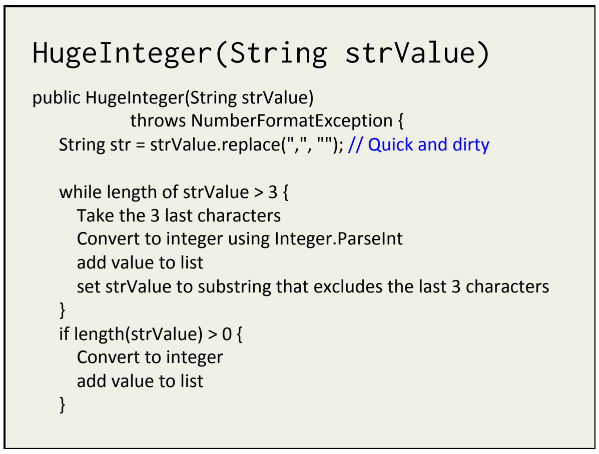
```
ArithmeticException
ArrayIndexOutOfBoundsException
IllegalArgumentException
IndexOutOfBoundsException
NegativeArraySizeException
NullPointerException
NumberFormatException
StringIndexOutOfBounds
UnsupportedOperationException
```



HugeInteger(long intValue)

```
do {
  add (intValue % 1000) to the list (append)
  intValue /= 1000;
} while( intValue > 0);
```

Don't forget the case of zero!



HugeInteger(String strValue)

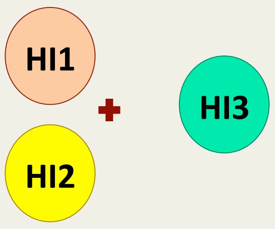
```
public HugeInteger(String strValue)
            throws NumberFormatException {
  String str = strValue.replace(",", ""); // Quick and dirty

  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
  }
```

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 class or object method? Justify.

Add two HugeInteger objects

class or object method?



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;
```

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.

Add 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.

Quiz

1. What is the output of the following program:

```
public class ZeroDiv{
  public static void main(String args[]){
    int x = 0, y = 10;
    try {
       y /= x;
    }
    System.out.print("/ by 0");
    catch(Exception e) {
       System.out.print("error");
    }
  }
}
```

a. 0
b. error
c. [✓ Compilation fails]
d. An uncaught exception is thrown at runtime.
e. No output

2. (2 points) Which one(s) of the following statements about Java exceptions are correct? (Several answers possible)

a. "throw" can be used to declare an exception in a method, and the exception will be thrown in this method

b. "throws" is used to throw the exception objects

c. [✓ "try" is used to detect if there is an exception in its block. If so, it intercepts the exception and executeis the code in the "catch" block.]

d. [✓ No matter if there is an exception or not, the code in "finally" will be executed.]

e. You cannot throw an exception in a "try" block.

3. What is the output of the following code?

```
public class Test {
   private static void test(int[] arr) {
       for (int i = 0; i < arr.length; i++) {
           try {
               if (arr[i] % 2 == 0) {
                   throw new NullPointerException();
               } else {
                   System.out.print(i);
               }
           } finally {
               System.out.print("e");
           }
       }
   }
   public static void main(String[] args) {
       try {
           test(new int[] {0, 1, 2, 3, 4, 5});
       } catch (Exception e) {
           System.out.print("E");
       }
   }
}
```

a. Compilation error
b. [✓ eE]
c. Ee
d. eE1eE3eE5
e. Ee1Ee3Ee5

4. (2 points) What is the output of the following code?

```
public class Test {

    public static void main(String[] args) {
        System.out.println("return value of getValue(): "
                           + getValue());
    }

    public static int getValue() {
         try {
             return 0;
         } finally {
             return 1;
         }
     }
 }
```

a. [✓ return value of getValue(): 1]
b. return value of getValue(): 0
c. return value of getValue(): 0return value of getValue(): 1
d. return value of getValue(): 1return value of getValue(): 0

5. Which is the correct statement about Java exceptions?

a. If you defined a possible exception in a method with "throw" you definitely have this exception when you use this method.

b. If there is no exception throwing out in the "try" block, the code in the "finally" block will never be executed.

c. If your program thows an exception,there must be an error in your program. You must debug and fix the error.

d. An unchecked exception in Java derives from RuntimeException or its children.

6. You don't need a "finally" block to release resources if you wrote your "try" block as follows:

```
try (// Acquire resources here) {
  ...
} catch ... {
}
```

a. True
b. False

7. Which statement is incorrect?

a. a "try" block can't be omitted.

b. multiple "catch" blocks can be used.

c. a "finally" block can be omitted.

d. a "finally" block can be used without any "try" or "catch" block.

8. The program reads a value entered by the user. How to create a custom exception that is thrown if the input value is greater than 10?

a. `if (i > 10) throw new Exception("something's wrong!");`

b. `if (i > 10) throw Exceptione("something's wrong!");`

c. `if (i > 10) throw new Exceptione("something's wrong!");`

d. `if (i > 10) throw Exception("something's wrong!");`

9. In the program below, variables a, b, and c will be stored respectively where?

```
class A {
    private String a = "aa";

    public boolean methodB() {
        String b = "bb";
        final String c = "cc";
  }
}
```

a. heap, heap, heap

b. heap, stack, heap

c. heap, stack, stack

d. heap, heap, stack

e. static, stack, heap

"aa", "bb" and "cc" are all in the heap but we are talking here about references to these values.

10. What characterizes the Set Interface?

a. A Set is a collection of element which contains elements along with their key.

b. A Set is a collection of element which contains hashcode of elements.

c. A Set is a collection of element which cannot contain duplicate elements.

d. A Set is a collection of element which is always ordered.

11. What is the parent class of Error and Exception classes?

a. Throwable

b. Catchable

c. MainError

d. MainException

12. Which arithmetic operations can result in the throwing of an ArithmeticException?

a. / , %

b. * , +

c. ! , -

d. >>, <<

13. The following are descriptions about List interface, Set interface and Map interface, which is false?

a. They all inherit from the Collection interface

b. List is an ordered interface, so we can control precisely where each element is inserted when using the interface

c. Set is a collection that does not contain duplicate elements

d. Map provides a mapping from key to value. A Map cannot contain the same key several times, each key can only map a value.

14. The following description is about the Collection class, which is false?

a. Both ArrayList and LinkedList implement the List interface

b. Access to elements of an ArrayList is faster than elements of a LinkedList

c. When adding and removing elements, ArrayList performs better than LinkedList

d. HashMap implements the Map interface, which allows any type of key and value object and allows null to be used as a key or value

15. Which of the following interfaces are directly inherited from the Collection interface? (Multiple answers)

a. List

b. Map

c. Set

d. Iterator

16. The following statement is about Collection and Collections, which is correct? (Multiple answers)

a. Collection is a class under java.util, which contains various static methods for collection operations;

b. Collection is a class under java.util, which is the parent interface of various collection structures;

c. Collections is a class under java.util, which is the parent interface of the various collection structures;

d. Collections is a class under java.util, which contains various static methods for collection operations

17. What will the running program print out ?

```
 public class Test{
    public static void main(String [] args){
        List list=new ArrayList();
        list.add("a");
        list.add("b");
        list.add("a");
        Set set=new HashSet();
        set.add("a");
        set.add("b");
        set.add("a");
        System.out.println(list.size()+","+set.size());
    }
}
```

a. 2,2
b. 2,3
c. 3,2
d. 3,3

18. (2 points) After the following code is executed, what are the elements in NumberList?

```
List<Integer> NumberList =new ArrayList<Integer>();
NumberList.add(2);
NumberList.add(4);
NumberList.add(1);
NumberList.add(3);
NumberList.add(5);
for(int i = 0; i < NumberList.size(); ++i) {
    int v = NumberList.get(i);
    if(v%2 == 0) {
        NumberList.remove(v);
    }
}
System.out.println(NumberList);
```

a. 2,4,1,3,5
b. 2,1,3,5
c. 4,1,3,5
d. There will be an out of bounds exception

19. A local variable in a method can be public

a. True

b. False

20. (2 points) Consider the following code; what will happen?

```
class MyError extends Error{ }

public class TestError {
       public static void main(String args[]) {
         try {
           test();
         } catch(Error ie) {
           System.out.println("Error caught");
         }
       }

       static void test() throws Error {
          throw new MyError();
       }
}
```

a. Run time error test() method does not throw an error type instance
b. Compile time error Cannot catch Error type objects
c. Compile time error Error class cannot be extended
d. Prints "Error caught"

21. Annotations about annotations are called:

a. Deprecations

b. Metadata

c. Meta-annotations

d. There is no such thing

22. If a method is tagged with the @Deprecated annotation:

a. Nothing special happens, it's just an information for javac

b. javac issues a warning but generates the .class file

c. javac ends in error

It says "throw a warning if someone uses this method" but nothing happens when compiling the class where the annotation is added.

23. If a Java interface defines two methods or more, you cannot use lambda expressions:

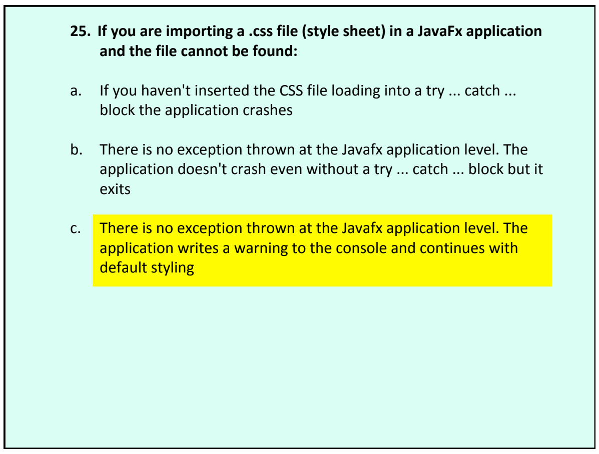
a. True

b. False

24. In a Graphical User Interface you can only store a container in a different kind of container (for instance you can add a vertical box to a grid or to a horizontal box, but not to another vertical box)

a. True

b. False

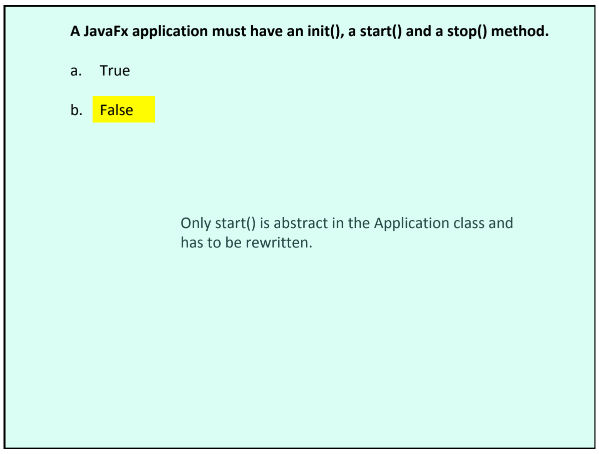


25. If you are importing a .css file (style sheet) in a JavaFx application and the file cannot be found:

a. If you haven't inserted the CSS file loading into a try ... catch ... block the application crashes

b. There is no exception thrown at the Javafx application level. The application doesn't crash even without a try ... catch ... block but it exits

c. [✓ There is no exception thrown at the Javafx application level. The application writes a warning to the console and continues with default styling]



A JavaFx application must have an init(), a start() and a stop() method.

a. True

b. [✓ False]

Only start() is abstract in the Application class and has to be rewritten.