CSC 151 Assignment 6

Objectives:

- Implementing solutions that uses java.util package(Deque and ArrayDeque)
- Using Generics
- Using Interfaces
- Simulating basic arithmetic operations on very huge numbers using Deque

Definition

Please refer to Project 10 of Chapter 7 in the textbook.

The largest positive integer of type int is 2,147,483,647. Another integer type, long, represents integers up to 9,223,372,036,854,775,807. Imagine that you want to represent even larger integers. For example, cryptography uses integers having more than 100 digits. Design and implement a class Huge of very large nonnegative integers. The largest integer should contain at least 30 digits. Use a deque to represent the value of an integer.

Provide operations for the class that

- Set the value of a nonnegative integer (provide both set methods and constructors)
- Return the value of a nonnegative integer as a string
- Read a large nonnegative integer (skip leading zeros, but remember that zero is a valid number)
- Display a large nonnegative integer (do not display leading zeros, but if the integer is zero, display a single zero)
- Add two nonnegative integers to produce the sum as a third integer
- Multiply two nonnegative integers to produce the product as a third integer

You should handle overflow when reading, adding, or multiplying integers. An integer is too large if it exceeds MAX_SIZE digits, where MAX_SIZE is a named constant that you define. Write a test program that demonstrates each method.

Class Huge

- Use the partial code provided at the end of the document.
- It will be in **package assignment** (it is required otherwise **Gradescope** cannot run the tests)
- See the details below written as comments and complete the partial code as instructed.
- Test your class in the main method.
- Test your class by JUnit

General Submission Instructions

- You may have groups for the assignments **if it is allowed.** If you have a group state it **explicitly in the honor code** by writing all group members' names. Please name the pdf files accordingly. **However, all group members should submit individually.**
- All submissions require a package called **assignment** and your source codes (.java files) will be under it.
- You should submit
  - Your java files to **Gradescope** as required and
  - Your java files and sample outputs for various scenarios as a **single pdf** file named as **YourFullName(s)_AssignmentN.pdf** with the **honor code** at the top as a comment to Nexus under the appropriate link. Replace N in AssignmentN with the appropriate value. (e.g., if I submit Assignment1 I will submit a pdf file whose name is **ZeynepOrhan_Assignment1.pdf.** If I have a group whose members are me and Jane Doe, then the file name will be **JaneDoeZeynepOrhanAssignment1.pdf** and the honor code will be modified). Use the template .docx file attached and modify it accordingly.
- **If you use any resource to get help, please state it in the honor code. Otherwise, it will be an honor code violation!!!**
- **Gradescope provides a feature to check code similarity by comparing all submissions. If the percentage of your code's similarity is above a threshold, then this will be an honor code violation!!!**
- Make sure that you have no compiler errors.
- When you have compiler error free codes and submit appropriately to Gradescope, your code will be tested automatically and graded. You do not need to know the test cases, but you will see the test results. Informative messages will be provided for success and failure cases.
- You can resubmit your files any number of times if you have failing tests until the due date.
- Your final submission or the one you selected will be graded after the due date of the assignment.
- Please start as early as possible so that you may have time to come and ask if you have any problems.
- Read the late submission policy in the syllabus.
- Please be aware that the correctness, obeying the OOP design and submission rules are equally important
- Gradescope testing will generally be 100% and sometimes manual grading will be done. Please be aware that **your grade can be reduced if you do not follow the instructions and apply good programming practices**.
- Use the starter code if provided.
- Do not use any predefined Collection classes of Java unless stated otherwise.

Assignment specific instructions

- General submission instructions apply.
- **Group work:** **NOT ALLOWED!!!**
- **Due date:** February 13, 2021, until 11:50 PM
- **Nexus:** Submit the code file (**Huge.java**) as a single pdf file named as **YourFullName_Assignment6.pdf** with the honor code at the top as a comment under Assignment 6 link.
- **Gradescope:** Submit the java files (**Huge.java**) under the Assignment 6
- **Grading:**
    - Gradescope: Yes
    - Manual: No
- **Starter code:** Use the code given below

```java
/*
 * Honor code
 */

package assignment;

import java.util.ArrayDeque;
import java.util.Deque;

/**
 * This class represents large nonnegative integers having up to 50 digits.
 *
 * @author Jesse Grabowski
 * @author Charles Hoot
 * @author Frank M. Carrano
 * @author Joseph Erickson
 * @author Zeynep Orhan modified
 * @version 5.0
 */
public class Huge {
	private Deque<Integer> hugeNumber;
	private final static int MAX_SIZE = 50;

	/**
	 * Constructor: No parameters. Creates an ArrayDeque for hugeNumber and add 0
	 */




	/**
	 * Constructor with an Integer array parameter: Creates an ArrayDeque for
	 * hugeNumber and sets the value of the Huge number based on a given array of
	 * Integers
	 *
	 * @param digits Integer[] digits
	 */




	/**
	 * Constructor with a String parameter: Creates an ArrayDeque for hugeNumber and
	 * sets the value of the Huge number based on a given String
	 *
	 * @param hugeString a String of digits
	 */




	/**
	 * Constructor with another Huge number parameter: Creates an ArrayDeque for
	 * hugeNumber and sets the value of the Huge number based on a given Huge number
	 *
	 * @param huge
	 */




	/**
	 * setHuge: Sets the value of the Huge number based on a given array of Integers. May
	 * throw an Error if the number is too large. throw new Error("Overflow") when
	 * the number of digits is greater than the MAX_SIZE Ignore zeros at the
	 * beginning and add the digits starting from the 0th index to the end of the
	 * deque
	 *
	 * @param digits An array of Integers that represents the Huge's digits.
	 *
	 */
```

```java
        /**
         * setHuge: Sets the value of the Huge number based on a string of numbers. May throw an
         * Error if the number is too large, or a NumberFormatException if the string is
         * formatted incorrectly.
         *
         * throw new Error("Overflow") when the number of digits is greater than the
         * MAX_SIZE
         *
         * throw new NumberFormatException("Non-hugeNumber in huge int"); when a digit
         * is not a character in [0-9]
         *
         * Ignore zeros at the beginning and add the digits starting from the char at
         * 0th index to the end of the deque
         *
         * @param hugeString The string to convert into a Huge.
         * @throws NumberFormatException if the string is formatted incorrectly.
         */





        /**
         * Override toString: Print the digits next to each other without a space
         */




        /**
         * toArray: Converts the Huge into an array of Integers.
         *
         * @return An array representation of the Huge.
         */




        /**
         * addHuge: Adds another Huge to this Huge without changing either one.
         *
         * @param h The Huge to add to this Huge.
         * @return A Huge which is the sum of both Huges.
         */





        /**
         * multiplyHuge: Multiplies another Huge to this Huge without changing either one.
         *
         * @param h The Huge to multiply to this Huge.
         * @return A Huge which is the product of both Huges.
         */




        /**
         * getHuge: Returns a duplicate of the given Huge representation of a String.
         *
         * @param s The String to convert into a Huge.
         * @return A duplicate of the Huge version of the String.
         */



        /**
         * isZero: Determines if the Huge is = 0.
         *
         * @return true if the Huge is 0, otherwise false
         */



} // end Huge
```

```
package assignment;

/**
 * This class to demonstrate the class Huge.
 *
 * @author Charles Hoot
 * @author Frank M. Carrano
 * @author Zeynep Orhan modified
 * @version 5.0
 */
public class Driver {
        public static void main(String[] args) {
                Integer h1[] = { 0, 0, 0, 0, 0, 0, 0, 0, 0 };
                Integer h2[] = { 0, 0, 0, 0, 0, 0, 0, 0, 1 };
                Integer h3[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
                Integer h4[] = { 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2 };
                Huge huge1 = new Huge(h1);
                Huge huge2 = new Huge(h2);
                Huge huge3 = new Huge(h3);
                Huge huge4 = new Huge(h4);
                Huge addHuge, multHuge, rHuge;

                if (huge1.isZero()) System.out.println(huge1 + " is zero.");
                else                System.out.println(huge1 + " is not zero.");

                if (huge2.isZero()) System.out.println(huge2 + " is zero.");
                else                System.out.println(huge2 + " is not zero.");

                if (huge3.isZero()) System.out.println(huge3 + " is zero.");
                else                System.out.println(huge3 + " is not zero.");

                if (huge4.isZero()) System.out.println(huge4 + " is zero.");
                else                System.out.println(huge4 + " is not zero.");

                rHuge = new Huge();
                addHuge = huge3.addHuge(huge4);
                multHuge = huge3.multiplyHuge(huge4);
                System.out.println(huge3 + " + " + huge4 + " = " + addHuge);
                System.out.println(huge3 + " * " + huge4 + " = " + multHuge);

                System.out.println("huge1 is " + huge1);
                System.out.println("huge2 is " + huge2);
                System.out.println("huge3 is " + huge3);
                System.out.println("huge4 is " + huge4);

                String goodString = "1234567890123456789012345678901234567890";
                String badString = "12345678901234567890x12345678901234567890";

                try {
                        System.out.println("Converting a string to a huge integer:");
                        rHuge = Huge.getHuge(goodString);
                        System.out.println(rHuge.toString());

                        System.out.println("Converting a string to a huge integer that is too large:");
                        rHuge = Huge.getHuge(badString);
                        System.out.println(rHuge.toString());
                } // end try
                catch (NumberFormatException e) {
                        System.out.println("Error converting a string to a huge integer.");
                } // end catch

                System.out.println("Zeros at the beginning should be removed");
                Integer h7[] = { 0, 0, 2, 2, 2 };
                Huge huge7 = new Huge(h7);
                System.out.println(huge7);

                System.out.println("Set to zero");
                huge7.setHuge("0");
                System.out.println(huge7);

                System.out.println("Trying a multiplication that will result in overflow.");
                Huge huge5 = Huge.getHuge(goodString);
                Huge huge6 = huge5.multiplyHuge(huge5);

        } // end main
} // end Driver
```

Output

```
0 is zero.
1 is not zero.
123456789123456789 is not zero.
222222222222222222 is not zero.
123456789123456789 + 222222222222222222 = 345679011345679011
123456789123456789 * 222222222222222222 = 27434842027434841972565157972565158
huge1 is 0
huge2 is 1
huge3 is 123456789123456789
huge4 is 222222222222222222
Converting a string to a huge integer:
1234567890123456789012345678901234567890
Converting a string to a huge integer that is too large:
Error converting a string to a huge integer.
Zeros at the beginning should be removed
222
Set to zero
0
Trying a multiplication that will result in overflow.
Exception in thread "main" java.lang.Error: Overflow
        at assignment.Huge.grow(Huge.java:295)
        at assignment.Huge.multiplyHuge(Huge.java:276)
        at assignment.Driver.main(Driver.java:81)
```