CS 251
Program 08

Due: Tuesday, November 21st

Main topics: Inheritance
Polymorphism
Abstract Classes

Program Specification:

Being able to program with numbers that do not (in theory) have a maximum value is a necessity in many applications of computer science. You are going to write a series of classes to get started on this task. Your final class will allow you to represent and at least add Binary numbers of arbitrary length.

`public abstract AbstractBit`: You are to write (implement) this class exactly as dictated by the following list of class members.

```
private boolean bit;

public abstract AbstractBit clone();
public abstract AbstractBit addBits(AbstractBit guest);
public abstract AbstractBit addBits(AbstractBit guest1, AbstractBit guest2);
public abstract AbstractBit carryBit(AbstractBit guest);
public abstract AbstractBit carryBit(AbstractBit guest1, AbstractBit guest2);

protected void setBit(boolean value)

public boolean getBit()

public AbstractBit()

public AbstractBit(boolean value)

public AbstractBit(AbstractBit guest)

public boolean equals(AbstractBit guest)

public String toString()
```

`public BinaryBit extends AbstractBit`: You are to write (implement) this class exactly as dictated by the following list of class members.

```
public static final BinaryBit zero = new BinaryBit(false);
public static final BinaryBit one = new BinaryBit(true);

public BinaryBit()

// allows construction with a boolean - false->0, true->1
public BinaryBit(boolean bit)
```

```
// allows construction with an int - should be 0 or 1
public BinaryBit(int bit)

public BinaryBit(BinaryBit guest)

public BinaryBit clone()

public boolean equals(BinaryBit guest)

public String toString()

// returns the low order bit of adding the host bit to the guest bit
public AbstractBit addBits(AbstractBit guest)

// returns the low order bit of adding the host bit, the guest1 bit, and the guest2 bit
public AbstractBit addBits(AbstractBit guest1, AbstractBit guest2)

// returns the high order bit of adding the host bit to the guest bit
public AbstractBit carryBit(AbstractBit guest)

// returns the high order bit of adding the host bit to the guest bit
public AbstractBit carryBit(AbstractBit guest1, AbstractBit guest2)
```

public BitString: You are to write (implement) this class exactly as dictated by the following list of class members.

```
// an ordered sequence of bits
private ArrayList<AbstractBit> bitString;

private void setAbstractBitList(ArrayList<AbstractBit> bitList)

protected ArrayList<AbstractBit> getAbstractBitList()

// adds a bit to the end of the arrayList of bits
public void addBit(AbstractBit bit)

// returns a reference to the bit at location loc in the ArrayList of bits
public AbstractBit bitAt(int loc)

public BitString()

protected BitString(ArrayList<AbstractBit> bitList)

public BitString(BitString guest)

public boolean isEmpty()

public int length()

public BitString clone()
```

```
  public boolean equals(BitString guest)

  public String toString()
```

public Binary extends BitString: You are to write (implement) this class exactly as dictated by the following list of class members.

```
  public Binary()

  // encode a non-negative (base 10) number val such into
  // the host's bitstring - used in the Binary(long val) constructor
  private void encode(long val)

  public Binary(long val)

  public Binary(BitString guest)

  public Binary(Binary guest)

  public Binary clone()

  // return a new Binary number that is the result of adding the
  // host bitstring to the guest bitstring under then assumption
  // that both are representing (base 2) numbers
  public Binary addition(Binary guest)
```

Your Class must also work with the following Driver Class public Driver:

```
public class Driver
{
  public static void main(String[] args)
  {
    Binary n1 = new Binary();
    Binary n2 = new Binary(10);
    Binary n3 = n2.clone();

    System.out.println("n1 = " + n1);
    System.out.println("n1.length() = " + n1.length());
    System.out.println("n2 = " + n2);
    System.out.println("n2.length() = " + n2.length());
    System.out.println("n3 = " + n3);
    System.out.println("n3.length() = " + n3.length());
    System.out.println("n3.bitAt(1) = " + n3.bitAt(1));
    System.out.println();
    System.out.println("n1 equals n1 ? " + n1.equals(n1));
    System.out.println("n1 equals n2 ? " + n1.equals(n2));
    System.out.println("n2 equals n3 ? " + n2.equals(n3));
    System.out.println();
    Binary n4 = n2.addition(n2);
```

```
    System.out.println("n4 = " + n4);
     for (int i = 0; i <= 10; ++i)
    {
      n4 = n4.addition(n4);
      n4 = n4.addition(n2);
      System.out.println("n4 = " + n4);
    }
  }
}
```

And produce the following output exactly:

```
n1 = 0
n1.length() = 1
n2 = 1010
n2.length() = 4
n3 = 1010
n3.length() = 4
n3.bitAt(1) = 1

n1 equals n1 ? true
n1 equals n2 ? false
n2 equals n3 ? true

n4 = 10100
n4 = 110010
n4 = 1101110
n4 = 11100110
n4 = 111010110
n4 = 1110110110
n4 = 11101110110
n4 = 111011110110
n4 = 1110111110110
n4 = 11101111110110
n4 = 111011111110110
n4 = 1110111111110110
```

Submission:

1. Use your web browser to open:

   `https://uwm.courses.wisconsin.edu/`

2. Login to D2L
3. Under *2179 - Fall 2017* you should see *CEAS-Computer Science* and under that *Intro Computer Programming*
4. Click on Intro Computer Programming
5. Click on **Dropbox** in the lower top menu bar

6. Click on **Program 08** in the *Programming Assignments* folder of the the current window
7. Click the **Add a File** button in the left center of the current window
8. Click the **Upload** button in the right top of the *Submit a File* pop-up window
9. Use the *File Upload* pop-up window to find the Java source code file you wish to submit: e.g. *Program08.java*
10. Click on this file name in the right panel of the *File Upload* pop-up window
11. Click the **Open** button in the *File Upload* pop-up window
12. Click the **Add** button in the bottom right top of the *Submit a File* pop-up window
13. Click the **Submit** button in the top / bottom right right of the current window