```java
import java.util.ArrayList;
import java.util.Scanner;


/**
 * @author 1828799
 */
public class BigInt
{
   // Variable Declaration
   private ArrayList<Integer> _bigInt = new ArrayList<Integer>();


   /**
    * Takes a string representation of a number to be used as a BigInt
    * @param value the value to be used as a bigInt
    */
   public BigInt(String value)
   {
      // Construct the ArrayList representation of the value
      for (int i = value.length() - 1; i > -1; i--)
      {
         _bigInt.add((int) (value.charAt(i)));
      }


   }

   /**
    * Takes a long representation of a number to be used as a BigInt
    * @param value the value to be used as a bigInt
    */
   public BigInt(long value)
   {
      // Create the variable to be manipulated to create a BigInt
      long theNum = value;

      // Construct the ArrayList representation of the value
      while (theNum > 0)
      {
         _bigInt.add((int) (theNum % 10));
         theNum = theNum / 10;
      }
   }

   /**
    * Overrides the toString method and gives a string representation
    * of the number
    * @override toString
    */
   public String toString()
   {
      // String representation of BigInt
      String bigInt = "";

      for (int value: _bigInt)
      {
         bigInt = String.valueOf(value) + bigInt;
      }
      return bigInt;
   }

   /**
    * Performs addition
    * @param number the number to be added
```

```java
     * @return the sum of the BigInt and the number passed in
     */
    public BigInt add(BigInt number)
    {
       int size;
       int remainder;

       if(_bigInt.size() < (number.toString()).length())
       {
          size = _bigInt.size();
       }
       else
       {
          size = number.toString().length();
       }

       for (int i = 0; i < size; i++)
       {

       }

       return null;
    }


    /**
     * Performs multiplication
     * @param number the number to be multiplied
     * @return the product of the two
     */
    public BigInt multiply(BigInt number)
    {
       return null;
    }


    public static void main(String[] args)
    {
       Scanner in = new Scanner(System.in);

       String firstArg;
       String operator;
       String secondArg;
       boolean repeat = true;

       while(repeat)
       {
          System.out.println("Compute: ");
          firstArg = in.next();
          operator = in.next();
          secondArg = in.next();
          in.hasNext();



       }
    }
}
```

```java

import java.util.Vector;
/**
 *
 * @author 1828799
 *
 */
public class Matrix
{
   private int _height, _width; // size of matrix
   private Vector _rows; // vector of row vectors


   public Matrix(int h, int w)
   // pre: h >= 0, w >= 0
   // post: constructs an h row by w column matrix
   {
      _height = h; // initialize height and width
      _width = w;

      // allocate a vector of rows
      _rows = new Vector(_height);

      for (int r = 0; r < _height; r++)
      {  // each row is allocated and filled with nulls
         Vector theRow = new Vector(_width);
         _rows.add(theRow);

         for (int c = 0; c < _width; c++)
         {
            theRow.add(null);
         }
      }
   }

   public Object get(int row, int col)
   // pre: 0 <= row < height(), 0 <= col < width()
   // post: returns object at (row, col)
   {
      Vector theRow = null;

      if(0 <= row && row < _height)
      {
         if(0 <= col && col < _width)
         {
            theRow = (Vector)_rows.get(row);
         }
      }
      return theRow.get(col);
   }

   public void set(int row, int col, Object value)
   // pre: 0 <= row < height(), 0 <= col < width()
   // post: changes location (row, col) to value
   {
      if(0 <= row && row < _height)
      {
         if(0 <= col && col < _width)
         {
            Vector theRow = (Vector)_rows.get(row);
            theRow.set(col, value);
         }
      }
   }

   public void addRow(int r)
   // pre: 0 <= row < height()
   // post: inserts row of null values to be row r
   {
      if(0 <= r && r < height())
      {
         _height++;
         Vector theRow = new Vector(_width);

         for (int c = 0; c < _width; c++)
         {
            theRow.add(null);
         }

         _rows.add(r, theRow);
      }
   }

   public void addCol(int c)
   // pre: 0 <= col < width()
   // post: inserts column of null values to be column c
   {
     if(0 <= c && c < width())
     {
        _width++;


        // Iterate forward through the rows
        for (int i = 0; i < _height; i++)
        {
           // Adds a column to each row that is null and shifts the elements
           // to the right over by one
           ((Vector) (_rows.get(i))).add(c, null);
        }
     }
   }

   public Vector removeRow(int r)
   // pre: 0 <= row < height()
   // post: removes row r and returns it as a Vector
   {
      Vector theRow = null;

      if(0 <= r && r < _height)
      {
         _height--;
         theRow = (Vector) _rows.remove(r);
      }

      return theRow;
   }

   public Vector removeCol(int c)
   // pre: 0 <= col < width()
   // post: removes column c and returns it as a Vector
   {
      Vector theRow = null;

      if(0 <= c && c < _width)
      {
         _width--;

         // Iterate forward through the rows
         for (int i = 0; i < _height; i++)
         {
```

```java
                theRow = (Vector) ((Vector) (_rows.get(i))).remove(c);
            }
        }

        return theRow;
    }

    public int width()
    // post: returns number of columns in matrix
    {
        return _width;
    }

    public int height()
    // post: returns number of rows in matrix
    {
        return _height;
    }

}
```

```
Note: Matrix.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
```

```
Author: 1828799

1.) 2.2: What are the pre- and postconditions for String’s charAt method?

    Preconditions: 0 < index < length of string, not null

    Postconditions: it returns a char at the given index

2.) 3.4: The implementation of java.util.Vector provides a method setSize.
This method explicitly sets the size of the Vector. Why is this useful? Is it
possible to set the size of the Vector without using this method?

    Yes. You can use a constructor to set the size of a vector because it
    accepts an initial capacity.
```