```
package a3;

import java.util.Scanner;

/**
 *  @author Connor Cousineau a3 cs1410
 *
 */
public class LoopPatterns {


      /**
       * calls all the functions to prove that they do what they are
supposed to do.
       * @param args still don't know.
       */
      public static void main(String[] args)
                 {

                 System.out.println("The Smallest Positive number is : " +
findSmallestPositiveNumber("2 -4 5"));
                 System.out.println("The Smallest Positive number is : " +
findSmallestPositiveNumber("1 -2 -5"));
                 System.out.println("Is a palindrome: " +
isPalindrome("abbba"));
                 System.out.println("Is a palindrome: " +
isPalindrome("abbbA"));
                 System.out.println("Has more even than odd: " +
hasMoreEvenThanOdd( " 1 3 4 6 -8"));
                 System.out.println("Has more even than odd: " +
hasMoreEvenThanOdd( " 1 3 0 0 0 0 0 0 0 0 0"));
                 System.out.println("Here is the camelCase: " +
camelCase("make this a camel case"));
                 System.out.println( "The lowest alphabetical word is: " +
lowestAlphabetically("cat dog apple fish "));
                 System.out.println("The lowest alphabetical word is: " +
lowestAlphabetically("zebra dog duck fish cat  "));
                 System.out.println( timesTable(3, 3));
                 System.out.println( timesTable(4, 4));
                 System.out.println( timesTable(1, 1));

                 }

                 /**
                  * Scans the String and compares the numbers, sets the
smallest number if it is smaller.
                  * @param numbers takes in a string of numbers.
                  * @return the smallest number
                  */
                 public static int findSmallestPositiveNumber(String
numbers)
                 {
                             Scanner scanner = new Scanner(numbers);
```

```
                              int smallestNumber = scanner.nextInt();

                              while (scanner.hasNext())
                              {
                                    if(smallestNumber < 0)
                              {
                                    smallestNumber = scanner.nextInt();
                              }
                                    int newNumber = scanner.nextInt();
                                    if (newNumber < 0)
                                    {
                                     newNumber = scanner.nextInt();
                                    }
                                    else if (newNumber < smallestNumber)
                                    {
                                                smallestNumber = newNumber;
                                    }
                                    else if (newNumber > smallestNumber)
                                    {
                                                newNumber = 
scanner.nextInt();

                                    }
                              }


                               return smallestNumber;
                  }


                  /**
                   * reverses the number and compares it to the original.
                   * @param palindrome takes in a word to test if it is a 
Palindrome.
                   * @return true if it is, false if it is not.
                   */
                  public static boolean isPalindrome(String palindrome)
                  {

                              String reversedPalindrome = new 
StringBuilder(palindrome).reverse().toString();

                              if (palindrome.equals(reversedPalindrome))
                                    return true;
                              else
                                    return false;
                  }



                  /**
                   * Scans the String and changes count values to determine 
even or not.
```

```
                  * @param numbers takes in a string of numbers.
                  * @return true if it does, false if it does not.
                  */
                 public static boolean hasMoreEvenThanOdd(String numbers)
                 {
                             int evenCount = 0;
                             int oddCount = 0;

                            Scanner scanner = new Scanner(numbers);
                             while(scanner.hasNext())
                             {
                                  int nextNumber = scanner.nextInt();
                                  if (nextNumber == 0)
                                  {

                                  }
                                  else if(nextNumber%2 > 0)
                                             oddCount++;
                                  else if (nextNumber%2 == 0)
                                  {
                                             evenCount++;
                                  }

                             }

                             if (evenCount > oddCount)
                             {
                                   return true;
                             }
                             else
                             {
                                   return false;
                             }


                 }


                 /**
                  * Takes in the string and upper-cases the first letter
of all the next words, gets rid of the spaces.
                  * @param sentence takes in a string of words to be
processed.
                  * @return returns the words as a camel case.
                  */
                 public static String camelCase(String sentence)
                 {
                            String strangeSentence = "";
                            String strangeWord = "";
                            String word;
                            Scanner scanner = new Scanner(sentence);
                          String firstWord = scanner.next();
                 while (scanner.hasNext())
                 {
                            if (scanner.hasNext())
                            {
                            word = scanner.next();
                            char bigLetter = word.charAt(0);
                            word = word.substring(1, word.length());
                            bigLetter = Character.toUpperCase(bigLetter);
                            strangeWord = bigLetter + word;
                            }
                            strangeSentence = strangeSentence +
strangeWord;
                 }
                            // Change or remove this statement as needed
                            return firstWord + strangeSentence;







                 }




                 /**
                  * Takes the string and compares gets a word, it then
compares the word to determine its location and value.
                  * @param words takes in words to be processed.
                  * @return the lowest alphabetical word.
                  */
                 public static String lowestAlphabetically(String words)
                 {
                            Scanner scanner = new Scanner(words);
                            String firstWord = scanner.next();
                            while(scanner.hasNext())
                            {
                            String     wordToCompare = scanner.next();
                            int result =
firstWord.compareTo(wordToCompare);
                            if (result >= 0)
                            {
                                  firstWord = wordToCompare;
                            }




                            }
                            return firstWord;
                 }
```

```
                /**
                 * Creates the spacing for the words, by adding a space
per the spacing parameter.
                 * @param spacing creates the spacing for the numbers on
the table.
                 * @return return the spaces to be input into the table.
                 */
                public static String spacingForAll(int spacing)
                {
                            String space = "";

                             for(int b = 1; b<=spacing-1; b++)
                             {
                                  space += ' ';
                             }
                            return space;
                }


                /**
                 * creates the header and side of the times table.
                 * @param maxNumber The biggest number.
                 * @param spacing The number of spaces in between the
numbers
                 * @return returns the entire times table.
                 */
                public static String timesTable(int maxNumber, int
spacing)
                {

                            int x = 1;
                            int y = 1;

                            String line = "";
                            String header = "";
                            String column = "";

                             while(x<= maxNumber)
                             {
                                  header += spacingForAll(spacing) + x;
                                  x++;
                                  while(y <= maxNumber)
                                   {

                                              column += y + "|" +
math(maxNumber, y, spacing) + "\n" ;
                                               // add the numbers code to
this line
                                               y++;
                                   }
                             }
```

```
                                   for(int z = 1; z <= spacing; z++)
                                    {
                                             for(int a = 1; a <= spacing;
a++ )
                                              line += '-' ;
                                    }





                                             return "  " +  header + "\n"
+"  "+ line +  "\n" + column ;


                  }

                  /**
                   * @param maxNumber The largest number.
                   * @param x the number for the purpose of creating the
math for the table.
                   * @param spacing The number that determines the spacing
                   * @return returns the values for the times table.
                   */
                  public static String math(int maxNumber, int x, int
spacing)
                  {
                              int y = 1;
                              int numberToPrint = 0;
                              String numbers = "";
                              while (y<= maxNumber)
                              {
                                    numberToPrint = y * x;
                                    y++;

                                    numbers += spacingForAll(spacing +
biggerThan9(numberToPrint)) + Integer.toString(numberToPrint) ;
                              }

                              return numbers;
                  }

                  /**
                   * Returns a minus one if the number is bigger than 9.
                   * @param x takes in a number from the times table math.
                   * @return remove a space or do nothing.
                   */
                  public static int biggerThan9(int x)

                  {

                              if(x > 9)
                              {
```

```
                        return -1;
                        }
                        else
                        {
                             return 0;
                        }
              }


}
```