P6.1

```
public class ArrayPrinter
{
   public static void main(String[] args)
   {
      int[] data = new int[10];
      for (int i = 0; i < 10; i++)
      {
         data[i] = (int) (Math.random() * 100 + 1);
      }

      // Print out even indices
      for (int i = 0; i < data.length; i = i + 2)
      {
         System.out.print(data[i] + " ");
      }
      System.out.println();

      // Print out even elements
      for (int i = 0; i < data.length; i++)
      {
         if (data[i] % 2 == 0)
         {
            System.out.print(data[i] + " ");
         }
      }
      System.out.println();

      // Print out elements in reverse order
      for (int i = data.length - 1; i >= 0; i--)
      {
         System.out.print(data[i] + " ");
      }
      System.out.println();

      // Print out only first and last element
      System.out.printf("%d %d\n", data[0], data[data.length - 1]);
   }
}
```

P6.2

Array methods that do the following:
f) Move all even elements to the front, otherwise preserving order

```
public static void moveEvenToFront(int[] arr)
{
   int endOfEvens = 0;
   int temp;

   for (int i = 0; i < arr.length; i++)
   {
      if (arr[i] % 2 == 0) // Even
      {
```

```
            temp = arr[i];  // Save the even number

            // Move array element from end of
            // evens to current location
            for (int j = i; j > endOfEvens; j--)
            {
               arr[j] = arr[j - 1];
            }
            arr[endOfEvens] = temp;
            endOfEvens++;
         }
      }
   }

   public static void main(String[] args)
   {
      int[] randoms = new int[10];

      // Create a test array containing random numbers.
      for (int i = 0; i < randoms.length; i++)
      {
         randoms[i] = (int) (Math.random() * 100) + 1;
        // Print the values as they are assigned.
         System.out.print(randoms[i] + " " );
      }
      System.out.println();

      // Move the evens to the front.
      moveEvenToFront(randoms);

      // Print again to see new order.
      for (int i = 0; i < randoms.length; i++)
      {
         System.out.print(randoms[i] + " " );
      }
      System.out.println();
   }
```

P6.20

```
final static int ROWS = 4;
final static int COLS = 4;

public static double neighborAverage(int[][] values, int row, int
column)
{
   int i = row;
   int j = column;
   double total = 0;
   int count = 0;

   if (i == 0)
   {
      if (j == 0)
      {
         total = values[i][j + 1]
```

```
            + values[i + 1][j]
            + values[i + 1][j + 1];
         count = 3;
      }


      if ((j > 0) &&  (j != COLS - 1))
      {
         total = values[i][j - 1]
            + values[i][j + 1]
            + values[i + 1][j]
            +  values[i + 1][j + 1]
            + values[i + 1][j - 1];
         count = 5;
      }
      if ((j > 0) && (j == COLS - 1))
      {
         total = values[i][j - 1]
            + values[i + 1][j]
            + values[i + 1][j - 1];
         count = 3;
      }
   }

   else if (i == ROWS -1)
   {
      if (j == 0)
      {
         total = values[i - 1][j]
            + values[i - 1] [j + 1]
            + values[i][j + 1];
         count = 3;
      }

      if ((j > 0) &&  (j != COLS - 1))
      {
         total = values[i][j - 1]
            + values[i][j + 1]
            + values[i - 1][j]
            + values[i - 1][j + 1]
            + values[i - 1][j - 1];
         count = 5;
      }

      if ((j > 0) && (j == COLS - 1))
      {
         total = values[i][j - 1]
            + values[i - 1][j]
            + values[i - 1][j - 1];
         count = 3;
      }
   }

   else if (j == 0)
   {
      if (i == 0)
      {
```

```
      total = values[i][j + 1]
         + values[i + 1][j]
         + values[i + 1][j + 1];
      count = 3;
   }

   if ((i > 0) &&  (i != ROWS - 1))
   {
      total = values[i - 1][j]
         + values[i][j + 1]
         + values[i - 1][j + 1]
         + values[i + 1][j + 1]
         + values[i + 1][j];
      count = 5;
   }

   if ((i > 0) && (i == ROWS - 1))
   {
      total = values[i - 1][j + 1]
         + values[i - 1][j]
         + values[i][j + 1];
      count = 3;
   }
}

else if (j == COLS -1)
{
   if (i == 0)
   {
      total = values[i][j-1]
            + values[i + 1][j-1]
            + values[i + 1][j];
         count = 3;
      }

      if ((i > 0) &&  (i != ROWS - 1))
      {
         total = values[i - 1][j]
            + values[i][j - 1]
            + values[i - 1][j-1]
            + values[i + 1][j - 1]
            + values[i + 1][j];
         count = 5;
      }

      if ((i>0) && (i == ROWS - 1))
      {
         total = values[i - 1][j]
            + values[i - 1][j-1]
            + values[i][j - 1];
         count = 3;
      }
}

else
{
   total = total +
```

```
            + values[i + 1][j]
            + values [i + 1][j + 1]
            + values [i + 1][j - 1]
            + values [i][j - 1]
            + values [i][j + 1]
            + values [i - 1][j]
            + values [i - 1][j + 1]
            + values [i - 1][j - 1];
         count = 8;
      }

   return total / count;

}
```

P6.28

```
   public static ArrayList<Integer> mergeSorted(ArrayList<Integer> a,
      ArrayList<Integer> b)
   {
      int i = 0;
      int j = 0;
      int k = 0;
      ArrayList<Integer> c = new ArrayList(a.size() + b.size());
      while (i < a.size() && j < b.size())
      {
         if (a.get(i) < b.get(j))
         {
            c.set(k, a.get(i));
            i++;
         }
         else
         {
            c.set(k, b.get(j));
            j++;
         }
         k++;
      }

      // More a elements to add?
      if (i < a.size())
      {
         while (i < a.size())
         {
            c.set(k, a.get(i));
            i++;
            k++;
         }
      }
      else if (j < b.size())  // More b elements to add?
      {
         while (j < b.size())
         {
            c.set(k, b.get(j));
            j++;
            k++;
```

```
         }
      }
      return c;
   }

public static void main(String[] args)
{
   ArrayList<Integer> a = new ArrayList(4);
   ArrayList<Integer> b = new ArrayList(5);

   int i;

   // Initialize array list a to some values
   for (i = 0; i < a.size(); i++)
   {
      a.set(i, (i + 1) * (i + 1));
   }

   // Initialize array list b to some values
   b.set(0, 9);
   b.set(1, 7);
   b.set(2, 4);
   b.set(3, 9);
   b.set(4, 11);

   ArrayList<Integer> c = mergeSorted(a, b);

   System.out.println("Result of merge sort of a and b is " );

   for (i = 0; i < c.size(); i++)
   {
      System.out.println(c.get(i) + " ");
   }
}
```

P7.9

```
import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
   This program reads each line in a file, reverses its lines and
writes them
   back to another file.
*/
public class ReverseLines
{
   /**
      This method reads a file line by line and returns the result in
an array
      list.
      @param filename the file to read
      @return an array list with the lines in the file
```

```
   */
   public static ArrayList<String> readLinesToArray(String filename)
   {
      ArrayList<String> lineList = new ArrayList<String>();
      try
      {
         Scanner inFile = new Scanner((new File(filename)));
         while (inFile.hasNextLine())
         {
            lineList.add(inFile.nextLine());
         }
         inFile.close();
      } catch (FileNotFoundException e)
      {
         System.out.println(filename + " not found.");
      }
      return lineList;
   }

   /**
      Write all lines in array list to given file in reverse order.
      @param filename the name of the file to write to
      @param lines the lines to write to the file
   */
   public static void writeInReverse(String filename, ArrayList<String>
lines)
   {
      try
      {
         PrintWriter out = new PrintWriter(filename);
         for (int i = lines.size() - 1; i >= 0; i--)
         {
            out.println(lines.get(i));
         }
         out.close();
      }
      catch (FileNotFoundException e)
      {
         System.out.println("Cannot open " + filename + " for
writing.");
      }
   }

   public static void main(String[] args) // File names are passed as
arguments
   {
      ArrayList<String> linesOfFile = readLinesToArray(args[0]);
      writeInReverse(args[1], linesOfFile);
   }
}
```

P7.13

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

```
public class P7_13
{
   /**
      Reads a set of floating-point values from the console and returns
      the sum. When the user enters a value that is not a number, it
      gives the user a second chance to enter the value.  After that it
      returns the sum.
    */
   public static double readGetPoint()
   {
      Scanner console = new Scanner(System.in);

      System.out.println("Enter a list of floating point values: ");
      int failures = 0;
      double sum = 0;
      while (failures < 2)
      {
         try
         {
            sum += console.nextDouble();
            failures = 0;
         }
         catch (InputMismatchException e)
         {
            System.out.println("That was not a floating point value.");
            String garbage = console.next();
            failures++;
         }
      }
      System.out.println("Two consecutive non-floating point values
were "
         + "entered.");
      return sum;
   }

   public static void main(String[] args)
   {
      double sum = readGetPoint();
      System.out.println("The sum is: " + sum);
   }
}
```