APCS Cool Set 5

1. Write a simple program that fills a 10 element array with random integers between 1 and 20 and graphs the information in the form of a bar chart, or histogram — each number is printed, and then a bar consisting of that many asterisks is printed beside the number.

OUTPUT

Element	Value	Histogram
0	19	**********
1	3	***
2	15	**********
3	7	*******
4	11	**********
5	9	*********
6	13	**********
7	5	*****
8	17	**********
9	1	*

2. **Create a method that will receive an integer parameter and then return an ArrayList that contains all of the number's factors, excluding itself. Note that `getListOfFfactors` is a static method which means you don't create an `ArrayListFunHouse` object to call it.**

```
import java.util.ArrayList;
public class ArrayListFunHouse
{
  //returns an ArrayList with the factors of number
  public static ArrayList<Integer> getListOfFactors(int number)
  {
     ArrayList<Integer> factors = new ArrayList<Integer>();
        ?

  }
}

//Driver
public class FactorList
{
   public static void main( String args[] )
   {
      System.out.println(ArrayListFunHouse.getListOfFactors(9));
      System.out.println(ArrayListFunHouse.getListOfFactors(23));
      System.out.println(ArrayListFunHouse.getListOfFactors(50));
      System.out.println(ArrayListFunHouse.getListOfFactors(100));
      System.out.println(ArrayListFunHouse.getListOfFactors(762));
   }
}
```

Sample Data :

```
9
23
50
100
762
```

Sample Output :

```
[1, 3]
[1]
[1, 2, 5, 10, 25]
[1, 2, 4, 5, 10, 20, 25, 50]
[1, 2, 3, 6, 127, 254, 381]
```

3. **Write a `Number` class that generates an <u>array</u> of 1500 distinct integers between 1 and 10000. Your `Number` class will write this <u>array</u> to an external file and will also have a method that will find all the perfect numbers in this array. These perfect numbers are to be added to an <u>ArrayList</u> as they are found. This <u>ArrayList</u> is then output to the screen and to the aforementioned external file. A perfect number is any number equal to the sum of its divisors (6 = 1+2+3).**

The output shown below only shows 300 numbers (because 1500 were a bit too many to put in this document).

```
Array "bunch" is:

221   17    117   337   31    382   46    345   180   383   126   32    27    359   35
280   227   278   96    63    56    394   106   322   144   53    156   274   388   248
85    386   159   3     342   301   204   286   151   67    326   312   271   113   233
232   282   178   285   324   49    172   70    54    273   329   302   374   341   254
311   57    212   36    69    361   102   354   316   399   51    347   384   47    234
79    43    2     230   310   164   62    292   376   87    400   219   65    21    206
194   109   323   86    99    290   295   125   334   34    40    143   391   373   313
262   330   253   199   186   145   333   306   240   119   13    105   268   58    135
169   396   308   19    351   170   272   211   116   20    214   247   275   84    352
304   197   279   155   163   203   401   146   88    389   284   181   6     235   364
355   358   357   123   107   97    129   22    132   281   288   276   153   91    251
83    71    131   381   379   264   182   370   75    277   196   25    188   73    339
372   246   298   55    80    14    296   133   293   37    208   327   367   200   111
74    110   215   244   207   16    226   243   112   30    236   269   256   224   307
150   222   343   7     241   257   238   183   48    398   15    28    332   387   167
134   185   344   223   44    42    176   305   375   395   331   368   321   317   148
377   245   303   365   72    92    121   217   287   362   356   202   336   195   158
142   130   318   141   267   64    38    193   41    320   249   60    338   184   283
77    29    108   154   300   59    294   160   50    95    82    120   348   392   173
205   76    157   33    10    250   349   161   350   137   231   218   162   175   360

The perfect squares are:  6       28
```

```
//Driver
import java.util.ArrayList;
import static java.lang.System.*;

public class NumberTester
{
   public static void main(String[] args)
   {
      Numbers burrito = new Numbers(300);
      out.println("Array \"bunch\" is:\n");
      burrito.showArray();
      burrito.findPerfect();
      ArrayList<Integer> taco = burrito.getPerfectList();
      out.print("\n\nThe perfect squares are: ");
      for( Integer i : taco )
      { out.printf("%-8d", i); }
   }
}
```

4. Write a program that creates a text file with 20 randomly generated “students”. Each student consists of a name and an ID number (both random). Your program will:
   - Create the name and ID number of each student (make sure all id #’s are different).
   - Create an external file "names&IDs.txt" with this data.
   - Read back this information and store it in an array of `Student` objects.
   - Call the `alphabetizeArray` of the `Sort` class to put the list in alpha order.
   - Display the alphabetized array on the screen and also append it to the "names&IDs.txt" file.

Unsorted group:

qfq	**wdbtteplq**	**80404**
upcf	**int**	**94048**
di	**ktclqxr**	**54402**
xtxelgh	**msvdxv**	**79620**
bfpgalk	**h**	**98937**
nhn	**qvexbd**	**98195**
rnv	**vpfvti**	**61599**
qqvndfb	**lgp**	**93058**
gcnj	**jmngbsu**	**73000**
wcnps	**xg**	**44532**
gnihrc	**ng**	**99496**
nc	**xdjmboa**	**95343**
mjbv	**mdmsn**	**20913**
clofpo	**qnkpwn**	**47582**
cfx	**pnkhi**	**86674**
qijqrus	**fssfb**	**75914**
pwc	**iqsdje**	**65568**
ir	**tpaplvdl**	**28592**
cubingn	**wbq**	**63665**
vbdwt	**taowh**	**24779**

Alphabetized list:

bfpgalk	**h**	**98937**
cfx	**pnkhi**	**86674**
clofpo	**qnkpwn**	**47582**
cubingn	**wbq**	**63665**
di	**ktclqxr**	**54402**
gcnj	**jmngbsu**	**73000**
gnihrc	**ng**	**99496**
ir	**tpaplvdl**	**28592**
mjbv	**mdmsn**	**20913**
nc	**xdjmboa**	**95343**
nhn	**qvexbd**	**98195**
pwc	**iqsdje**	**65568**

```
qfq         wdbtteplq    80404
qijqrus     fssfb        75914
qqvndfb     lgp          93058
rnv         vpfvti       61599
upcf        int          94048
vbdwt       taowh        24779
wcnps       xg           44532
xtxelgh     msvdxv       79620
```

```
public class Student
{
    private String name;
    private int iD;

    public Student()
    {}

    public Student( String n, int i )
    {
        name = n;
        iD = i;
    }

    public String getName()
    { return name; }

    public int getId()
    { return iD; }
}
```

```
import java.util.*; //needed for class Random
public class RandomInfo  //Driver
{
    public static void main( String[] args )
    {
        int num;
        int id;
        int lastID=0;
        FileOutput out = new FileOutput( "names&IDs.txt" );
        Random number = new Random();
        Student[] group = new Student[20]; //group is an array of
                                           //twenty Student objects

        //in this next section you have to generate twenty random
        //names, each with a corresponding random 5-digit number.
        //Each name (first and last) can be up to 20 characters long
        //including the space between first and last.
        //With each name (and corresponding id #) a new Student is
        //instantiated (i.e. created) and added to array group:
        // group[k] = new Student( name, id );
```

```
                .
                .
                ?
                .
                .
        System.out.print( "Unsorted group:" + "\n\n" );
        out.writeString( "Unsorted group:" );
        out.writeEndOfLine();
        out.writeEndOfLine();
        for( Student r: group )
        {
            out.writeString( r.getName() + "\t\t" + r.getId() );
            out.writeEndOfLine();
            System.out.printf( "%18s %10d", r.getName(), r.getId() );
            System.out.println();
        }
        out.writeEndOfLine();
        System.out.println();
        out.close();

        FileOutput moreOut = new FileOutput( "names&IDs.txt", true );

        Sort.alphabetizeArray( group );

        System.out.print( "\n" + "Alphabetized list:" + "\n\n" );
        moreOut.writeString( "\n" + "Alphabetized list:" );
        moreOut.writeEndOfLine();
        moreOut.writeEndOfLine();
        for( Student r: group )
        {
            moreOut.writeString( r.getName() + "\t\t" + r.getId() );
            moreOut.writeEndOfLine();
            System.out.printf( "%18s %10d", r.getName(), r.getId() );
            System.out.println();
        }
    }
}

public class Sort
{
   public static void alphabetizeArray( Student[] objects )
   {
          .
          ?
          .
   }
}
```

5. Prime Numbers — SIEVE OF ERATOSTHENES

A prime integer is any integer that is evenly divisible only by itself and 1. The Sieve of Eratosthenes is a method of finding prime numbers. It operates as follows:

- Create an array with all elements initialized to 1 (true).

- Starting with array subscript 2, every time an array element is found whose value is 1, loop through the remainder of the array and set to zero every element whose subscript is a multiple of the subscript for the element with value 1. For array subscript 2, all elements beyond 2 in the array that are multiples of 2 will be set to zero (subscripts 4, 6, 8, 10, etc.); for array subscript 3, all elements beyond 3 in the array that are multiples of 3 will be set to zero (subscripts 6, 9, 12, 15, etc.); and so on.
  When this process is complete, the array elements that are still set to 1 indicate that the subscript is a prime number.

- Lastly, create an ArrayList containing only the prime numbers found.

Write a program that finds and prints all the prime numbers smaller than a user-defined number.

OUTPUT

You want to generate a list of all prime numbers smaller than...200

The prime numbers between 0 and 200 are...

```
2   3   5   7   11  13  17  19  23  29  31  37
41  43  47  53  59  61  67  71  73  79  83  89
97  101 103 107 109 113 127 131 137 139 149 151
157 163 167 173 179 181 191 193 197 199
```

There are a total of 46 prime numbers between 0 and 200

6. Write a program that takes in an integer (typed in by the user) and returns its binary equivalent.

For example:

Enter a positive integer...28
28 ==> 11100

Enter a positive integer...137
137 ==> 10001001

Enter a positive integer...649
649 ==> 1010001001

You must use the driver program shown below. Note that the `getBinary` method of the `Binary` class returns an array of `ints`.

```
public class TestBinary
{
    public static void main( String args[] )
    {
        System.out.print("Enter a positive integer...");
        int n = SavitchIn.readInt();
        Binary b = new Binary( n );
        int[] a = b.getBinary();
        System.out.print( n + " ==> " );
        for( int k = 0; k < a.length; k++ )
        { System.out.print( a[k] ); }
    }
}
```

7. Write a program that converts from:

dec to hex/bin
hex to dec/bin
bin to hex/dec

<u>Driver</u> (or write your own)

```
import java.util.ArrayList;
public class Driver
{
    public static void main(String args [])
    {
        ArrayList<Integer> binary = new ArrayList<Integer>();
        boolean okay = true;
        do
        {
            okay = true;
            System.out.print("Input Number Type(dec/hex/bin): ");
            String inputType = SavitchIn.readLine();
            if(inputType.equals("hex"))
            {
                System.out.print("Input Number = ");
                binary = Calculator.hexToBin(SavitchIn.readLine());
                System.out.println("dec= " + Calculator.binToDec(binary));
                System.out.print("bin= " );
                int h = 0;
                while(binary.get(h) == 0)h++;
                for(int k = h; k < binary.size(); k++)System.out.print(binary.get(k));
            }
            else
            {
                System.out.print("Input Number = ");
                int input = SavitchIn.readLineInt();
                if(inputType.equals("dec"))
                {
                    binary = Calculator.decToBin(input);
                    System.out.print("bin= " );
                    int h = 0;
                    while(binary.get(h) == 0)h++;
                    for(int k = h; k < binary.size(); k++)
                        System.out.print(binary.get(k));
                    System.out.println();
                    ArrayList<String> hexNumber = Calculator.binToHex(binary);
                    System.out.print("hex= " );
                    for(int k = 0; k < hexNumber.size(); k++)
                        System.out.print(hexNumber.get(k));
                }
                else
                {
                    if(inputType.equals("bin"))
                    {
                        binary = Calculator.binToBin(input);
                        System.out.println("dec= " + Calculator.binToDec(binary));
                        ArrayList<String> hexNumber = Calculator.binToHex(binary);
                        System.out.print("hex= " );
                        for(int k = 0; k < hexNumber.size();k++)
                            System.out.print(hexNumber.get(k));
                    }
                    else okay = false;
                }
            }
        }while(!okay);
    }
}
```

SAMPLE OUTPUT

Input Number Type(dec/hex/bin): dec
Input Number = 563
bin= 1000110011
hex= 233

Input Number Type(dec/hex/bin): hex
Input Number = 34A
dec= 842
bin= 1101001010

nput Number Type(dec/hex/bin): bin
Input Number = 1110101
dec= 117
hex= 75