Lab 2. Exercises on Java Basics

Writing Good Programs

The only way to learn programming is program, program and program. Learning programming is like learning cycling, swimming or any other sports. You can't learn by watching or reading books. Start to program immediately. On the other hands, to improve your programming, you need to read many books and study how the masters program.
It is easy to write programs that work. It is much harder to write programs that not only work but also easy to maintain and understood by others – I call these good programs. In the real world, writing program is not meaningful. You have to write good programs, so that others can understand and maintain your programs.
Pay particular attention to:

1. **Coding style**:
   - Read Java code convention: "Java Style and Commenting Guide".
   - Follow the Java Naming Conventions for variables, methods, and classes STRICTLY. Use CamelCase for names. Variable and method names begin with lowercase, while class names begin with uppercase. Use nouns for variables (e.g., radius) and class names (e.g., Circle). Use verbs for methods (e.g., getArea(), isEmpty()).
   - **Use Meaningful Names**: Do not use names like a, b, c, d, x, x1, x2, and x1688 - they are meaningless. Avoid single-alphabet names like i, j, k. They are easy to type, but usually meaningless. Use single-alphabet names only when their meaning is clear, e.g., x, y, z for co-ordinates and i for array index. Use meaningful names like row and col (instead of x and y, i and j, x1 and x2), numStudents (not n), maxGrade, size (not n), and upperbound (not n again). Differentiate between singular and plural nouns (e.g., use books for an array of books, and book for each item).
   - Use consistent indentation and coding style. Many IDEs (such as Eclipse / NetBeans) can re-format your source codes with a single click.
2. **Program Documentation**: Comment! Comment! and more Comment to explain your code to other people and to yourself three days later.
3. The only way to learn programming is program, program and program on challenging problems. The problems in this tutorial are certainly NOT challenging. There are tens of thousands of challenging problems available – used in training for various programming contests (such as International Collegiate Programming Contest (ICPC), International Olympiad in Informatics (IOI)).

1 Exercises on Array

1.1 PrintArray

Write a program called **PrintArray** which prompts user for the number of items in an array (a non-negative integer), and saves it in an int variable called NUM_ITEMS. It then prompts user for the values of all the items and saves them in an int array called items. The program shall then print the contents of the array in the form of $[x_1, x_2, ..., x_n]$. For example,

Command window

```
Enter the number of items: 5
Enter the value of all items (separated by space): 3 2 5 6 9
The values are: [3, 2, 5, 6, 9]
```

Hints

```
// Declare variables
final int NUM_ITEMS;
int[] items;  // Declare array name, to be allocated after NUM_ITEMS is known
......

// Prompt for for the number of items and read the input as "int"
......
NUM_ITEMS = ......

// Allocate the array
items = new int[NUM_ITEMS];

// Prompt and read the items into the "int" array, if array length > 0
if (items.length > 0) {
   ......
   for (int i = 0; i < items.length; ++i) {  // Read all items
      ......
   }
}

// Print array contents, need to handle first item and subsequent items differently
......
for (int i = 0; i < items.length; ++i) {
   if (i == 0) {
      // Print the first item without a leading commas
      ......
   } else {
      // Print the subsequent items with a leading commas
      ......
```

```
    }
31  // or, using a one liner
    //System.out.print((i == 0) ? ..... : .....);
33 }
```

1.2 PrintArrayInStars

Write a program called **PrintArrayInStars** which prompts user for the number of items in an array (a non-negative integer), and saves it in an *int* variable called NUM_ITEMS. It then prompts user for the values of all the items (non-negative integers) and saves them in an int array called items. The program shall then print the contents of the array in a graphical form, with the array index and values represented by number of stars. For examples,

Command window

```
1  Enter the number of items: 5
   Enter the value of all items (separated by space): 7 4 3 0 7
3  0: *******(7)
   1: ****(4)
5  2: ***(3)
   3: (0)
7  4: *******(7)
```

Hints

```
1  // Declare variables
   final int NUM_ITEMS;
3  int[] items;  // Declare array name, to be allocated after NUM_ITEMS is
       ↪ known
   ......
5  ......

7  // Print array in "index: number of stars" using a nested-loop
   // Take note that rows are the array indexes and columns are the value
       ↪ in that index
9  for (int idx = 0; idx < items.length; ++idx) {  // row
     System.out.print(idx + ": ");
11   // Print value as the number of stars
     for (int starNo = 1; starNo <= items[idx]; ++starNo) {  // column
13     System.out.print("*");
     }
15   ......
   }
17 ......
```

1.3 GradesStatistics

Write a program which prompts user for the number of students in a class (a non-negative integer), and saves it in an *int* variable called *numStudents*. It then prompts user for the grade of each of the students (integer between 0 to 100) and saves them in an *int* array called *grades*. The program shall then compute and print the average (in *double* rounded to 2 decimal places) and minimum/maximum (in *int*).

Command window

```
Enter the number of students: 5
Enter the grade for student 1: 98
Enter the grade for student 2: 78
Enter the grade for student 3: 78
Enter the grade for student 4: 87
Enter the grade for student 5: 76
The average is: 83.40
The minimum is: 76
The maximum is: 98
```

1.4 Hex2Bin

Write a program called **Hex2Bin** that prompts user for a hexadecimal string and print its equivalent binary string. The output shall look like:

Command window

```
Enter a Hexadecimal string: 1abc
The equivalent binary for hexadecimal "1abc" is: 0001 1010 1011 1100
```

Hints

1. Use an array of 16 Strings containing binary strings corresponding to hexadecimal number $0-9A-F$ (or $a-f$), as follows:

```
final String[] HEX_BITS = {"0000", "0001", "0010", "0011",
  "0100", "0101", "0110", "0111",
  "1000", "1001", "1010", "1011",
  "1100", "1101", "1110", "1111"};
```

1.5 Dec2Hex

Write a program called **Dec2Hex** that prompts user for a positive decimal number, read as *int*, and print its equivalent hexadecimal string. The output shall look like:

```
Command window
1  Enter a decimal number: 1234
   The equivalent hexadecimal number is 4D2
```

2 Exercises on Method

2.1 exponent()

Write a method called *exponent(int base, int exp)* that returns an *int* value of base raises to the power of exp. The signature of the method is:

```
public static int exponent(int base, int exp);
```

Assume that exp is a non-negative integer and base is an integer. Do not use any Math library functions.
Also write the *main()* method that prompts user for the *base* and *exp*; and prints the result. For example,

```
Command window
1  Enter the base: 3
   Enter the exponent: 4
3  3 raises to the power of 4 is: 81
```

Hints

```
1  ......
   public class Exponent {
3    public static void main(String[] args) {
       // Declare variables
5      int exp;    // exponent (non-negative integer)
       int base;   // base (integer)
7      ......
       // Prompt and read exponent and base
9      ......
       // Print result
11     System.out.println(base + " raises to the power of " + exp + " is:
           ↪ " + exponent(base, exp));
     }
13
     // Returns "base" raised to the power "exp"
```

```java
    public static int exponent(int base, int exp) {
      int product = 1;   // resulting product

      // Multiply product and base for exp number of times
      for (.....) {
        product *= base;
      }

      return product;
    }
  }
```

2.2 hasEight()

Write a *boolean* method called *hasEight()*, which takes an int as input and returns true if the number contains the digit 8 (e.g., 18, 168, 1288). The signature of the method is as follows:

```java
public static boolean hasEight(int number);
```

Write a program called **MagicSum**, which prompts user for integers (or −1 to end), and produce the sum of numbers containing the digit 8. Your program should use the above methods. A sample output of the program is as follows:

```
Command window

Enter a positive integer (or -1 to end): 1
Enter a positive integer (or -1 to end): 2
Enter a positive integer (or -1 to end): 3
Enter a positive integer (or -1 to end): 8
Enter a positive integer (or -1 to end): 88
Enter a positive integer (or -1 to end): -1
The magic sum is: 96
```

Hints

1. The *coding pattern* to repeat until input is −1 (called sentinel value) is:

```java
    final int SENTINEL = -1;  // Terminating input
    int number;

    // Read first input to "seed" the while loop
    System.out.print("Enter a positive integer (or -1 to end): ");
```

```
number = in.nextInt();

while (number != SENTINEL) {  // Repeat until input is -1
   ......
   ......

   // Read next input. Repeat if the input is not the SENTINEL
   // Take note that you need to repeat these codes!
   System.out.print("Enter a positive integer (or -1 to end): ");
   number = in.nextInt();
}
```

2. You can either repeatably use modulus/divide ($n\,\%\,10$ and $n = n/10$) to extract and drop each digit in *int*; or convert the *int* to *String* and use the *String*'s *charAt()* to inspect each *char*.

2.3 print()

Write a method called *print*(), which takes an int array and print its contents in the form of $[a_1, a_2, \ldots, a_n]$. Take note that there is no comma after the last element. The method's signature is as follows:

```
public static void print(int[] array);
```

Also write a test driver to test this method (you should test on empty array, one-element array, and n-element array).
How to handle *double*[] or *float*[]? You need to write a overloaded version for *double*[] and a overloaded version for *float*[], with the following signatures:

```
public static void print(double[] array)
public static void print(float[] array)
```

The above is known as *method overloading*, where the same method name can have many versions, differentiated by its parameter list.

Hints

1. For the first element, print its value; for subsequent elements, print commas followed by the value.

2.4 arrayToString()

Write a method called *arrayToString*(), which takes an int array and return a String in the form of $[a_1, a_2, \ldots, a_n]$. Take note that this method returns a *String*, the previous exercise returns void but prints the output. The method's signature is as follows:

```
public static String arrayToString(int[] array);
```

Also write a test driver to test this method (you should test on empty array, one-element array, and n-element array).

Notes

1. This is similar to the built-in function *Arrays.toString*(). You could study its source code.

2.5 contains()

Write a boolean method called *contains*(), which takes an array of *int* and an *int*; and returns *true* if the array contains the given *int*. The method's signature is as follows:

```
1 public static boolean contains(int[] array, int key);
```

Also write a test driver to test this method.

2.6 search()

Write a method called test *search*(), which takes an array of int and an *int*; and returns the array index if the array contains the given *int*; or -1 otherwise. The method's signature is as follows:

```
1 public static int search(int[] array, int key);
```

Also write a test driver to test this method.

2.7 equals()

Write a boolean method called *equals*(), which takes two arrays of int and returns true if the two arrays are exactly the same (i.e., same length and same contents). The method's signature is as follows:

```
public static boolean equals(int[] array1, int[] array2)
```

Also write a test driver to test this method.

2.8 copyOf()

Write a boolean method called *copyOf()*, which takes an *int* Array and returns a copy of the given array. The method's signature is as follows:

```
public static int[] copyOf(int[] array)
```

Also write a test driver to test this method.
Write another version for *copyOf()* which takes a second parameter to specify the length of the new array. You should truncate or pad with zero so that the new array has the required length.

```
public static int[] copyOf(int[] array, int newLength)
```

Notes

- This is similar to the built-in function *Arrays.copyOf()*.

2.9 swap()

Write a method called *swap()*, which takes two arrays of int and swap their contents if they have the same length. It shall return true if the contents are successfully swapped. The method's signature is as follows:

```
public static boolean swap(int[] array1, int[] array2)
```

Also write a test driver to test this method.

Notes

```java
// Swap item1 and item2
int item1;
int item2;
int temp;

temp = item1;
item1 = item2;
item2 = item1;
// You CANNOT simply do: item1 = item2; item2 = item2;
```

2.10 reverse()

Write a method called *reverse()*, which takes an array of int and reverse its contents. For example, the reverse of $[1, 2, 3, 4]$ is $[4, 3, 2, 1]$. The method's signature is as follows:

```java
public static void reverse(int[] array)
```

Take note that the array passed into the method can be modified by the method (this is called **pass by sharing**). On the other hand, primitives passed into a method cannot be modified. This is because a clone is created and passed into the method instead of the original copy (this is called **pass by value**).
Also write a test driver to test this method.

Hints

1. You might use two indexes in the loop, one moving forward and one moving backward to point to the two elements to be swapped.

```java
for (int fIdx = 0, bIdx = array.length - 1; fIdx < bIdx; ++fIdx
    ↪ , --bIdx) {
  // Swap array[fIdx] and array[bIdx]
  // Only need to transverse half of the array elements
}
```

2. You need to use a temporary location to swap two storage locations.

```java
// Swap item1 and item2
int item1;
int item2;
int temp;
```

```
5
      temp = item1;
7     item1 = item2;
      item2 = item1;
9     // You CANNOT simply do: item1 = item2; item2 = item2;
```

2.11 GradesStatistics

Write a program called **GradesStatistics**, which reads in n grades (of int between 0 and 100, inclusive) and displays the average, minimum, maximum, median and standard deviation. Display the floating-point values upto 2 decimal places. Your output shall look like:

Command window

```
Enter the number of students: 4
Enter the grade for student 1: 50
Enter the grade for student 2: 51
Enter the grade for student 3: 56
Enter the grade for student 4: 53
The grades are: [50, 51, 56, 53]
The average is: 52.50
The median is: 52.00
The minimum is: 50
The maximum is: 56
The standard deviation is: 2.29
```

The formula for calculating standard deviation is: $\sigma = \sqrt{\frac{1}{n}\sum_{i=0}^{n-1} x_i^2 - \mu^2}$, where μ is the mean.

Hints

```
1   public class GradesStatistics {
      public static int[] grades;  // Declare an int[], to be allocated
         ↪ later.
3     // This array is accessible by all the methods.

5     public static void main(String[] args) {
        readGrades();  // Read and save the inputs in static int[] grades
7       System.out.println("The grades are: ");
        print(grades);
9       System.out.println("The average is " + average(grades));
        System.out.println("The median is " + median(grades));
11      System.out.println("The minimum is " + min(grades));
        System.out.println("The maximum is " + max(grades));
13      System.out.println("The standard deviation is " + stdDev(grades));
```

```
    }

    // Prompt user for the number of students and allocate the static "
       ↪ grades" array.
    // Then, prompt user for grade, check for valid grade, and store in "
       ↪ grades".
    public static void readGrades() { ..... }

    // Print the given int array in the form of [x1, x2, x3,..., xn].
    public static void print(int[] array) { ..... }

    // Return the average value of the given int[]
    public static double average(int[] array) { ..... }

    // Return the median value of the given int[]
    // Median is the center element for odd-number array,
    // or average of the two center elements for even-number array.
    // Use Arrays.sort(anArray) to sort anArray in place.
    public static double median(int[] array) { ..... }

    // Return the maximum value of the given int[]
    public static int max(int[] array) {
      int max = array[0];   // Assume that max is the first element
      // From second element, if the element is more than max, set the
         ↪ max to this element.
      ......
    }

    // Return the minimum value of the given int[]
    public static int min(int[] array) { ..... }

    // Return the standard deviation of the given int[]
    public static double stdDev(int[] array) { ..... }
}
```

Take note that besides *readGrade()* that relies on class variable grades, all the methods are self-contained general utilities that operate on any given array.

2.12 GradesHistogram

Write a program called **GradesHistogram**, which reads in n grades (as in the previous exercise), and displays the horizontal and vertical histograms. For example:

Command window

```
  0 –  9: ***
 10 – 19: ***
 20 – 29:
 30 – 39:
 40 – 49: *
 50 – 59: *****
 60 – 69:
 70 – 79:
 80 – 89: *
 90 –100: **

  *
  *
  *      *                       *
  *      *                       *                       *
  *      *                 *     *                 *     *
 0–9  10–19 20–29 30–39 40–49 50–59 60–69 70–79 80–89 90–100
```