

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 **String** and **char** Operations

Write a Java program called **StringAndCharacterExercise** that implements the following methods.

1.1 ReverseString

Write a method called `reverseString(String inStr)` (with prototype **String reverseString(String inStr)**), which reverses the string **inStr** by extracting and processing each character. Write a method called **testReverseString()** which prompts user for a string, and prints the reverse of the string using the method **reverseString()**. The output shall look like:

Command window

```

1 Enter a String: abcdef
  The reverse of the String "abcdef" is "fedcba".

```

Hints

For a string called **inStr**, you can use **inStr.length()** to get the length of the **String**; and **inStr.charAt(idx)** to retrieve the **char** at the **idx** position, where **idx** begins at 0, up to **instr.length() - 1**.



```

// Define variables
2 String inStr;           // input String
  int inStrLen;           // length of the input String
4 .....

6 // Prompt and read input as "String"
  System.out.print("Enter a String: ");
8 inStr = in.next();      // use next() to read a String
  inStrLen = inStr.length();
10 .....

12 // Use inStr.charAt(index) in a loop to extract each character
  // The String's index begins at 0 from the left.
  // Process the String from the right
14 for (int charIdx = inStrLen - 1; charIdx >= 0; --charIdx) {
    // charIdx = inStrLen-1, inStrLen-2, ... ,0
16 .....
  }

```

1.2 CountVowelsDigits

Write a method called **countVowels()** (with prototype **int countVowels(String inStr)**) that counts the number of vowels (a, e, i, o, u, A, E, I, O, U) contained in the string **inStr**. Write a method called **countDigits()** (with prototype **int countDigits(String inStr)**) that

counts the number of digits (0 – 9) contained in the String `inStr`. Write a method called `testCountVowelsDigits()`, which prompts the user for a String, counts the number of vowels and digits contained in the string, and prints the counts and the percentages (rounded to 2 decimal places). For example,

```

Command window
1 Enter a String: testing12345
   Number of vowels: 2 (16.67%)
3  Number of digits: 5 (41.67%)

```

Hints

1. To check if a `char c` is a digit, you can use boolean expression `(c ≥ '0' && c ≤ '9')`; or use built-in boolean function `Character.isDigit(c)`.
2. You could use `in.next().toLowerCase()` to convert the input string to lowercase to reduce the number of cases.
3. To print a % using `printf()`, you need to use `%%`. This is because % is a prefix for format specifier in `printf()`, e.g., `%d` and `%f`.

1.3 PhoneKeyPad

On your phone keypad, the alphabets are mapped to digits as follows: ABC(2), DEF(3), GHI(4), JKL(5), MNO(6), PQRS(7), TUV(8), WXYZ(9). Write a method called `phoneKeyPad()` (with prototype `String phoneKeyPad(String inStr)`), which converts the string `inStr` to a sequence of keypad digits. Write a method called `testPhoneKeyPad()`, which prompts user for a string (case insensitive), and converts to a sequence of keypad digits using the method `phoneKeyPad()`. Use (a) a nested-if, (b) a switch-case-default.

Hints


1. You can use `in.next().toLowerCase()` to read a `String` and convert it to lowercase to reduce your cases.
2. In switch-case, you can handle multiple cases by omitting the break statement, e.g.,



```

1 switch (inChar) {
   case 'a':
3    case 'b':
   case 'c': // No break for 'a' and 'b', fall thru 'c'
5       System.out.print(2);
       break;
7    case 'd':
   case 'e':

```



```

9   case 'f':
    .....
11  default:
    .....
13  }

```

1.4 Caesar's Code

Caesar's Code is one of the simplest encryption techniques. Each letter in the plaintext is replaced by a letter some fixed number of position (n) down the alphabet cyclically. In this exercise, we shall pick $n = 3$. That is, 'A' is replaced by 'D', 'B' by 'E', 'C' by 'F', ..., 'X' by 'A', ..., 'Z' by 'C'.

Write a method called `cipherCaesarCode()` (with prototype `String cipherCaesarCode(String inStr)`) to cipher the Caesar's code. Write a method called `testCipherCaesarCode()`, which prompts user for a plaintext string consisting of mix-case letters only; compute the ciphertext; and print the ciphertext in uppercase. For example,

```

Command window
1 Enter a plaintext string: Testing
  The ciphertext string is: WHVWLQJ

```

Hints

1. Use `in.next().toUpperCase()` to read an input string and convert it into uppercase to reduce the number of cases.
2. You can use a big nested-if with 26 cases ('A' - 'Z'). But it is much better to consider 'A' to 'W' as one case; 'X', 'Y' and 'Z' as 3 separate cases.
3. Take note that char 'A' is represented as Unicode number 65 and char 'D' as 68. However, 'A' + 3 gives 68. This is because char + `int` is implicitly casted to `int + int` which returns an `int` value. To obtain a char value, you need to perform explicit type casting using `(char)('A' + 3)`. Try printing ('A' + 3) with and without type casting.

1.5 Decipher Caesar's Code

Write a method called `decipherCaesarCode()` (with prototype `String decipherCaesarCode(String inStr)`) to decipher the Caesar's code described in the previous exercise. Write a method called `testDecipherCaesarCode()`, which prompts user for a ciphertext string consisting of mix-case letters only; compute the plaintext; and print the plaintext in uppercase. For example,

Command window

```

Enter a ciphertext string: wHVwLQJ
2 The plaintext string is: TESTING

```

1.6 CheckHexString

The hexadecimal (hex) number system uses 16 symbols, 0 – 9 and A - F (or a - f). Write a method called `isHexString()` (with prototype `boolean isHexString(String hexStr)`) to verify a hex string. Write a method called `testHexString()`, which prompts user for a hex string; and decide if the input string is a valid hex string. For examples,

Command window

```

Enter a hex string: 123aBc
2 "123aBc" is a hex string

4 Enter a hex string: 123aBcx
  "123aBcx" is NOT a hex string

```

Hints



```

1 if (!((inChar >= '0' && inChar <= '9')
      || (inChar >= 'A' && inChar <= 'F')
      || (inChar >= 'a' && inChar <= 'f')))) { // Use positive logic and
3     // then reverse
    .....
5 }

```

1.7 BinaryToDecimal

Write a method called `binaryToDecimal()` (with prototype `int binaryToDecimal(String binStr)`) to convert an input binary string into its equivalent decimal number. Write a method called `testBinaryToDecimal()`, which prompts user for a binary string, and convert the input binary string into its equivalent decimal number. Your output shall look like:

Command window

```

1 Enter a Binary string: 1011
  The equivalent decimal number for binary "1011" is: 11
3
4 Enter a Binary string: 1234
5 error: invalid binary string "1234"

```

1.8 HexadecimalToDecimal

Write a method called `hexadecimalToDecimal()` (with prototype `int hexadecimalToDecimal(String hexStr)`) to convert an input hexadecimal string into its equivalent decimal number. Write a method called `testHexadecimalToDecimal()`, which prompts user for a hexadecimal string, and convert the input hexadecimal string into its equivalent decimal number. Your output shall look like:

```
Command window
1 Enter a Hexadecimal string: 1a
  The equivalent decimal number for hexadecimal "1a" is: 26
3
3 Enter a Hexadecimal string: 1y3
5 error: invalid hexadecimal string "1y3"
```

1.9 OctalToDecimal

Write a method called `octalToDecimal()` (with prototype `int octalToDecimal(String octalStr)`) to convert an input octal string into its equivalent decimal number. Write a method called `testOctalToDecimal()`, which prompts user for a octal string, and convert the input octal string into its equivalent decimal number. Your output shall look like:

```
Command window
1 Enter an Octal string: 147
  The equivalent decimal number "147" is: 103
```

1.10 RadixNToDecimal

Write a method called `radixNToDecimal()` (with prototype `int radixNToDecimal(String radixNStr)`) to convert an input string of any radix (≤ 16) into its equivalent decimal number. Write a method called `testRadixNToDecimal()`, which prompts user for a radix n string, and convert the input radix n string into its equivalent decimal number. Your output shall look like:

```
Command window
Enter the radix: 16
2 Enter the string: 1a
  The equivalent decimal number "1a" is: 26
```

2 Exercises on Array

Write a Java program called `ArrayExercise` that implements the following methods.

2.1 PrintArray

Write a method called `createArray()` (with prototype `int[] createArray(Scanner in)`) 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`. Write a method called `printArray(int[] arr)` to print the array `arr` in the form of $[x_1, x_2, \dots, x_n]$. Using the method `printArray()` to print the created array. For example,

Command window

```

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

```

Hints



```

1 // Declare variables
  final int NUM_ITEMS;
3 int [] items; // Declare array name, to be allocated after NUM_ITEMS is
  ↪ known
  .....
5
  // Prompt for for the number of items and read the input as "int"
7 .....
  NUM_ITEMS = .....
9
  // Allocate the array
11 items = new int[NUM_ITEMS];

13 // Prompt and read the items into the "int" array, if array length > 0
  if (items.length > 0) {
15     .....
    for (int i = 0; i < items.length; ++i) { // Read all items
17         .....
    }
19 }

21 // Print array contents, need to handle first item and subsequent items
  ↪ differently
  .....
23 for (int i = 0; i < items.length; ++i) {
    if (i == 0) {
25        // Print the first item without a leading commas
        .....
27    } else {
        // Print the subsequent items with a leading commas
29        .....
    }
}

```



```

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

```

2.2 GradesStatistics

Write a method called `generateStudentGrades()` (with prototype `int[] generateStudentGrades(Scanner in)`) 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. Write a method called `simpleGradesStatistics()`, which generates an array of grade of each of the students and saves them in an `int` array called `grades`. The method shall then compute and print the average (in `double` rounded to 2 decimal places) and minimum/maximum (in `int`). For example,

Command window

```

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

```

2.3 HexadecimalToBinary

Write a method called `hexadecimalToBinary()` (with prototype `String hexadecimalToBinary(String hexStr)`) to convert an input hexadecimal string into its equivalent binary string. Write a method called `testHexadecimalToBinary()` that prompts user for a hexadecimal string and print its equivalent binary string. The output shall look like:

Command window

```

1 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",
2  "0100", "0101", "0110", "0111",
  "1000", "1001", "1010", "1011",
4  "1100", "1101", "1110", "1111"};
```

2.4 DecimalToHexadecimal

Write a method called `decimalToHexadecimal()` (with prototype `String decimalToHexadecimal(int positiveInteger)`) to convert a positive decimal number into its equivalent hexadecimal string. Write a method call `testDecimalToHexadecimal()` 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
```

3 Exercises on Method

Write a Java program called `MethodExercise` that implements the following methods.

3.1 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 `testExponent()` 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 .....
2 public class MethodExercise {
3     public static void main(String[] args) {
4         .....
5     }

6
7     public static void testExponent(Scanner in) {
8         // Declare variables
9         int exp;    // exponent (non-negative integer)
10        int base;   // base (integer)
11        .....
12        // Prompt and read exponent and base
13        .....
14        // Print result
15        System.out.println(base + " raises to the power of "
16                             + exp + " is: " + exponent(base, exp));
17    }

18
19    // Returns "base" raised to the power "exp"
20    public static int exponent(int base, int exp) {
21        int product = 1;    // resulting product

22
23        // Multiply product and base for exp number of times
24        for (.....) {
25            product *= base;
26        }
27
28        return product;
29    }
30 }

```

3.2 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:



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

Write a method called `int testMagicSum(Scanner in)` 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
1 Enter a positive integer (or -1 to end): 1
  Enter a positive integer (or -1 to end): 2
3 Enter a positive integer (or -1 to end): 3
  Enter a positive integer (or -1 to end): 8
5 Enter a positive integer (or -1 to end): 88
  Enter a positive integer (or -1 to end): -1
7 The magic sum is: 96

```

Hints

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



```

1 final int SENTINEL = -1; // Terminating input
  int number;
3
  // Read first input to "seed" the while loop
5 System.out.print("Enter a positive integer (or -1 to end): ");
  number = in.nextInt();
7
  while (number != SENTINEL) { // Repeat until input is -1
9      .....
      .....
11
      // Read next input. Repeat if the input is not the SENTINEL
13 // Take note that you need to repeat these codes!
      System.out.print("Enter a positive integer (or -1 to end): ");
15 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`.

3.3 print()

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



```

1 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:



```
1 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.

3.4 arrayToString()

Write a method called `arrayToString()`, which takes an `int` array and return a `String` in the form of $[a_1, a_2, \dots, a_n]$. 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.

3.5 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.

3.6 search()

Write a method called `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.

3.7 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:



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

Also write a test driver to test this method.

3.8 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:



```
1 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.



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

Notes

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

3.9 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:



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

Also write a test driver to test this method.

Notes



```
1 // Swap item1 and item2
  int item1;
3 int item2;
  int temp;
5
  temp = item1;
7 item1 = item2;
  item2 = item1;
9 // You CANNOT simply do: item1 = item2; item2 = item2;
```

3.10 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:



```
1 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.



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

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



```
1 // Swap item1 and item2  
  int item1;  
3  int item2;  
  int temp;  
  
5  temp = item1;  
7  item1 = item2;  
  item2 = temp;  
9  // You CANNOT simply do: item1 = item2; item2 = item2;
```