Java Programming Tutorial

Exercises on Java Basics

You need to do these exercises by yourself. Please don't ask me for solutions!

TABLE OF CONTENTS (HIDE)

1. Getting Started Exercises
   - 1.1 HelloWorld
   - 1.2 CheckPassFail (if-else)
   - 1.3 CheckOddEven (if-else)
   - 1.4 PrintNumberInWord (nested-
   - 1.5 PrintDayInWord (nested-if,
2. Exercises on Number Systems (f
3. Writing Good Programs
4. Exercises on Decision and Loop
   - 4.1 SumAverageRunningInt (Dec
   - 4.2 Product1ToN (or Factorial)
   - 4.3 HarmonicSum (Decision & Loc
   - 4.4 ComputePI (Decision & Loop)
   - 4.5 CozaLozaWoza (Decision & Lo
   - 4.6 Fibonacci (Decision & Loop)
   - 4.7 ExtractDigits (Decision & L
5. Exercises on Input, Decision and
   - 5.1 Add2Integer (Input)
   - 5.2 SumProductMinMax3 (Arithme
   - 5.3 CircleComputation (double
   - 5.4 Swap2Integers
   - 5.5 IncomeTaxCalculator (Decis
   - 5.6 IncomeTaxCalculatorWithS
   - 5.7 PensionContributionCalcu
   - 5.8 PensionContributionCalcu
   - 5.9 SalesTaxCalculator (Decisi
   - 5.10 ReverseInt (Loop with Mod
   - 5.11 SumOfDigitsInt (Loop with
   - 5.12 InputValidation (Loop wit
   - 5.13 AverageWithInputValidat
6. Exercises on Nested-Loops
   - 6.1 SquarePattern (nested-loop)
   - 6.2 CheckerPattern (nested-loo
   - 6.3 TimeTable (nested-loop)
   - 6.4 TriangularPattern (nested-
   - 6.5 BoxPattern (nested-loop)
   - 6.6 HillPattern (nested-loop)
   - 6.7 NumberPattern (nested-loop)
7. Debugging/Tracing Programs us
   - 7.1 Factorial (Using a graphic d
8. Exercises on String and char C
   - 8.1 ReverseString (String & ch
   - 8.2 CountVowelsDigits (String
   - 8.3 PhoneKeyPad (String & char
   - 8.4 Caesar's Code (String & cha

1. Getting Started Exercises

1.1 HelloWorld

1. Install JDK on your machine. Follow the instructions in "How to Install JDK".
2. Write a Hello-world program using JDK and a source-code editor, such as:
   - For All Platforms: Sublime Text, Atom
   - For Windows: TextPad, NotePad++
   - For macOS: jEdit, gedit
   - For Ubuntu: gedit
3. Read "Introduction to Java Programming for Novices & First-Time Programmers". Do ALL the exercises.

1.2 CheckPassFail (if-else)

Write a program called **CheckPassFail** which prints "PASS" if the `int` variable "mark" is more than or equal to 50; or prints "FAIL" otherwise. The program shall always print "DONE" before exiting.

Hints

Use >= for *greater than or equal to* comparison.

```
/**
 * Trying if-else statement.
 */
public class CheckPassFail {  // Save as "CheckPassFail.java"
   public static void main(String[] args) {  // Program entry point
      int mark = 49;   // Set the value of "mark" here!
      System.out.println("The mark is " + mark);

      // if-else statement
      if ( ...... ) {
         System.out.println( ...... );
      } else {
```

```
         System.out.println( ..... );
      }
      System.out.println( ..... );
   }
}
```

Try `mark = 0`, 49, 50, 51, 100 and verify your results.

Take note of the source-code **indentation**!!! Whenever you open a block with '{', indent all the statements inside the block by 3 (or 4 spaces). When the block ends, un-indent the closing '}' to align with the opening statement.

1.3 CheckOddEven (if-else)

Write a program called **CheckOddEven** which prints "Odd Number" if the int variable "number" is odd, or "Even Number" otherwise. The program shall always print "bye!" before exiting.

Hints

n is an even number if (*n* % 2) is 0; otherwise, it is an odd number. Use == for comparison, e.g., (n % 2) == 0.

```
/**
 * Trying if-else statement and modulus (%) operator.
 */
public class CheckOddEven {   // Save as "CheckOddEven.java"
   public static void main(String[] args) {  // Program entry point
      int number = 49;       // Set the value of "number" here!
      System.out.println("The number is " + number);
      if ( ..... ) {
         System.out.println( ..... );   // even number
      } else {
         System.out.println( ..... );   // odd number
      }
      System.out.println( ..... );
   }
}
```

Try number = 0, 1, 88, 99, -1, -2 and verify your results.

Again, take note of the source-code identation! Make it a good habit to ident your code properly, for ease of reading your program.

1.4 PrintNumberInWord (nested-if, switch-case)

Write a program called **PrintNumberInWord** which prints "ONE", "TWO",... , "NINE", "OTHER" if the int variable "number" is 1, 2,... , 9, or other, respectively. Use (a) a "nested-if" statement; (b) a "switch-case-default" statement.

Hints

```
/**
 * Trying nested-if and switch-case statements.
 */
public class PrintNumberInWord {   // Save as "PrintNumberInWord.java"
   public static void main(String[] args) {
      int number = 5;  // Set the value of "number" here!

      // Using nested-if
      if (number == 1) {   // Use == for comparison
         System.out.println( ..... );
```


8.5 Decipher Caesar's Code (Stri
8.6 Exchange Cipher (String & c
8.7 TestPalindromicWord and T
8.8 CheckBinStr (String & char
8.9 CheckHexStr (String & char
8.10 Bin2Dec (String & char)
8.11 Hex2Dec (String & char)
8.12 Oct2Dec (String & char)
8.13 RadixN2Dec (String & char
9. Exercises on Array
9.1 PrintArray (Array)
9.2 PrintArrayInStars (Array)
9.3 GradesStatistics (Array)
9.4 Hex2Bin (Array for Table Look
9.5 Dec2Hex (Array for Table Look
10. Exercises on Method
10.1 exponent() (method)
10.2 isOdd() (method)
10.3 hasEight() (method)
10.4 print() (Array & Method)
10.5 arrayToString() (Array & I
10.6 contains() (Array & Metho
10.7 search() (Array & Method)
10.8 equals() (Array & Method)
10.9 copyOf() (Array & Method)
10.10 swap() (Array & Method)
10.11 reverse() (Array & Metho
10.12 GradesStatistics (Array
10.13 GradesHistogram (Array &
11. Exercises on Command-line Ar
11.1 Arithmetic (Command-Line
12. More (Difficult) Exercises
12.1 JDK Source Code
12.2 Matrices (2D Arrays)
12.3 PrintAnimalPattern (Spec
12.4 Print Patterns (nested-loop)
12.5 Print Triangles (nested-loop)
12.6 Trigonometric Series
12.7 Exponential Series
12.8 Special Series
12.9 FactorialInt (Handling Ov
12.10 FibonacciInt (Handling O
12.11 Number System Conversion
12.12 NumberGuess
12.13 WordGuess
12.14 DateUtil
13. Exercises on Recursion
13.1 Factorial Recusive
13.2 Fibonacci (Recursive)
13.3 Length of a Running Numbe
13.4 GCD (Recursive)
13.5 Tower of Hanoi (Recursive)

```
      } else if ( ..... ) {
         .....
      } else if ( ..... ) {
         .....
      .....
      .....
      } else {
         .....
      }

      // Using switch-case-default
      switch(number) {
         case 1:
            System.out.println( ..... ); break;  // Don't forget the "break" after each case!
         case 2:
            System.out.println( ..... ); break;
         .....
         .....
         default: System.out.println( ..... );
      }
   }
}
```

14. Exercises on Algorithms - Sorti
14.1 Linear Search
14.2 Recursive Binary Search
14.3 Bubble Sort
14.4 Selection Sort
14.5 Insertion Sort
14.6 Recursive Quick Sort
14.7 Merge Sort
14.8 Heap Sort
15. Exercises on Algorithms - Num
15.1 Perfect and Deficient Numbe
15.2 Prime Numbers
15.3 Prime Factors
15.4 Greatest Common Divisor (G
16. Final Notes

Try `number = 0`, 1, 2, 3, ..., 9, `10` and verify your results.

1.5 PrintDayInWord (nested-if, switch-case)

Write a program called **`PrintDayInWord`** which prints "`Sunday`", "`Monday`", ... "`Saturday`" if the `int` variable "`dayNumber`" is `0`, `1`, ..., `6`, respectively. Otherwise, it shall print "Not a valid day". Use (a) a "nested-`if`" statement; (b) a "`switch-case-default`" statement.

Try `dayNumber = 0`, 1, 2, 3, 4, 5, 6, 7 and verify your results.

2. Exercises on Number Systems (for Science/Engineering Students)

To be proficient in programming, you need to be able to operate on these number systems:

1. Decimal (used by human beings for input and output)
2. Binary (used by computer for storage and processing)
3. Hexadecimal (shorthand or compact form for binary)

Read "Number Systems" section of "Data Representation", and complete the exercises.

3. 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" (@ https://www.oracle.com/technetwork/java/codeconventions-150003.pdf or google "Java Code Convention").
   - 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.

4. Exercises on Decision and Loop

4.1 SumAverageRunningInt (Decision & Loop)

Write a program called **`SumAverageRunningInt`** to produce the sum of `1`, `2`, `3`, ..., to `100`. Store `1` and `100` in variables `lowerbound` and `upperbound`, so that we can change their values easily. Also compute and display the average. The output shall look like:

```
The sum of 1 to 100 is 5050
The average is 50.5
```

Hints

```
/**
 * Compute the sum and average of running integers from a lowerbound to an upperbound using loop.
 */
public class SumAverageRunningInt {   // Save as "SumAverageRunningInt.java"
   public static void main (String[] args) {
      // Define variables
      int sum = 0;          // The accumulated sum, init to 0
      double average;       // average in double
      final int LOWERBOUND = 1;
      final int UPPERBOUND = 100;

      // Use a for-loop to sum from lowerbound to upperbound
      for (int number = LOWERBOUND; number <= UPPERBOUND; ++number) {
            // The loop index variable number = 1, 2, 3, ..., 99, 100
         sum += number;     // same as "sum = sum + number"
      }
      // Compute average in double. Beware that int / int produces int!
      ......
      // Print sum and average
      ......
   }
}
```

Try

1. Modify the program to use a "while-do" loop instead of "for" loop.

```
      int sum = 0;
      int number = LOWERBOUND;        // declare and init loop index variable
      while (number <= UPPERBOUND) {  // test
         sum += number;
         ++number;                    // update
      }
```

2. Modify the program to use a "do-while" loop.

```
int sum = 0;
int number = LOWERBOUND;          // declare and init loop index variable
do {
   sum += number;
   ++number;                      // update
} while (number <= UPPERBOUND);   // test
```

3. What is the difference between "for" and "while-do" loops? What is the difference between "while-do" and "do-while" loops?
4. Modify the program to sum from `111` to `8899`, and compute the average. Introduce an `int` variable called `count` to count the numbers in the specified range (to be used in computing the average).

```
int count = 0;   // Count the number within the range, init to 0
for ( ...; ...; ... ) {
   ......
   ++count;
}
```

5. Modify the program to find the "sum of the squares" of all the numbers from `1` to `100`, i.e. `1*1 + 2*2 + 3*3 + ... + 100*100`.
6. Modify the program to produce two sums: sum of odd numbers and sum of even numbers from `1` to `100`. Also computer their absolute difference.
   HINTS:

```
// Define variables
int sumOdd  = 0;   // Accumulating sum of odd numbers
int sumEven = 0;   // Accumulating sum of even numbers
int absDiff;       // Absolute difference between the two sums
......
// Compute sums
for (int number = ...; ...; ...) {
   if (......) {
      sumOdd += number;
   } else {
      sumEven += number;
   }
}
// Compute Absolute Difference
if (sumOdd > sumEven) {
   absDiff = ......;
} else {
   absDiff = ......;
}
// OR use one liner conditional expression
absDiff = (sumOdd > sumEven) ? ...... : ......;
```

4.2 `Product1ToN` (or `Factorial`) (Decision & Loop)

Write a program called **`Product1ToN`** to compute the product of integers from `1` to `10` (i.e., `1×2×3×...×10`), as an `int`. Take note that It is the same as factorial of `N`.

Hints

Declare an `int` variable called `product`, initialize to 1, to accumulate the product.

```
// Define variables
int product = 1;      // The accumulated product, init to 1
final int LOWERBOUND = 1;
final int UPPERBOUND = 10;
```

Try

1. Compute the product from 1 to 11, 1 to 12, 1 to 13 and 1 to 14. Write down the product obtained and decide if the results are correct.
   HINTS: Factorial of 13 (=6227020800) is outside the range of `int [-2147483648, 2147483647]`. Take note that computer programs may not produce the correct result even though the code seems correct!
2. Repeat the above, but use `long` to store the `product`. Compare the products obtained with `int` for `N=13` and `N=14`.
   HINTS: With `long`, you can store factorial of up to `20`.

4.3 HarmonicSum (Decision & Loop)

Write a program called **`HarmonicSum`** to compute the sum of a harmonic series, as shown below, where *n*=50000. The program shall compute the sum from *left-to-right* as well as from the *right-to-left*. Are the two sums the same? Obtain the absolute difference between these two sums and explain the difference. Which sum is more accurate?

$$Harmonic(n) = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$$

Hints

```
/**
 * Compute the sum of harmonics series from left-to-right and right-to-left.
 */
public class HarmonicSum {   // Save as "HarmonicSum.java"
   public static void main (String[] args) {
      // Define variables
      final int MAX_DENOMINATOR = 50000;  // Use a more meaningful name instead of n
      double sumL2R = 0.0;         // Sum from left-to-right
      double sumR2L = 0.0;         // Sum from right-to-left
      double absDiff;              // Absolute difference between the two sums

      // for-loop for summing from left-to-right
      for (int denominator = 1; denominator <= MAX_DENOMINATOR; ++denominator) {
          // denominator = 1, 2, 3, 4, 5, ..., MAX_DENOMINATOR
          ......
          // Beware that int/int gives int, e.g., 1/2 gives 0.
      }
      System.out.println("The sum from left-to-right is: " + sumL2R);

      // for-loop for summing from right-to-left
      ......

      // Find the absolute difference and display
      if (sumL2R > sumR2L) ......
      else ......
   }
}
```

4.4 ComputePI (Decision & Loop)

Write a program called **`ComputePI`** to compute the value of π, using the following series expansion. Use the maximum denominator (`MAX_DENOMINATOR`) as the terminating condition. Try `MAX_DENOMINATOR` of `1000`, `10000`, `100000`, `1000000` and compare the PI obtained. Is this series suitable for computing PI? Why?

$$\pi = 4 \times \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \frac{1}{13} - \frac{1}{15} + \cdots\right)$$

Hints

Add to `sum` if the `denominator % 4` is `1`, and subtract from `sum` if it is `3`.

```
double sum = 0.0;
int MAX_DENOMINATOR = 1000;   // Try 10000, 100000, 1000000
for (int denominator = 1; denominator <= MAX_DENOMINATOR; denominator += 2) {
      // denominator = 1, 3, 5, 7, ..., MAX_DENOMINATOR
   if (denominator % 4 == 1) {
      sum += ......;
   } else if (denominator % 4 == 3) {
      sum -= ......;
   } else {   // remainder of 0 or 2
      System.out.println("Impossible!!!");
   }
}
......
```

Try

1. Instead of using maximum denominator as the terminating condition, rewrite your program to use the maximum number of terms (`MAX_TERM`) as the terminating condition.

```
int MAX_TERM = 10000;  // number of terms used in computation
int sum = 0.0;
for (int term = 1; term <= MAX_TERM; term++) {  // term = 1, 2, 3, ..., MAX_TERM
      // term = 1, 2, 3, 4, ..., MAX_TERM
   if (term % 2 == 1) {  // odd term number: add
      sum += 1.0 / (term * 2 - 1);
   } else {              // even term number: subtract
      ......
   }
}
```

2. JDK maintains the value of π in a built-in `double` constant called `Math.PI` (`=3.141592653589793`). Add a statement to compare the values obtained and the `Math.PI`, in percents of `Math.PI`, i.e., `(piComputed / Math.PI) * 100`.

4.5 CozaLozaWoza (Decision & Loop)

Write a program called **`CozaLozaWoza`** which prints the numbers `1` to `110`, `11` numbers per line. The program shall print "Coza" in place of the numbers which are multiples of 3, "Loza" for multiples of 5, "Woza" for multiples of 7, "CozaLoza" for multiples of 3 and 5, and so on. The output shall look like:

```
1 2 Coza 4 Loza Coza Woza 8 Coza Loza 11
Coza 13 Woza CozaLoza 16 17 Coza 19 Loza CozaWoza 22
23 Coza Loza 26 Coza Woza 29 CozaLoza 31 32 Coza
......
```

Hints

```
public class CozaLozaWoza {   // Save as "CozaLozaWoza.java"
   public static void main(String[] args) {
      final int LOWERBOUND = 1, UPPERBOUND = 110;
      for (int number = LOWERBOUND; number <= UPPERBOUND; ++number) {
            // number = LOWERBOUND+1, LOWERBOUND+2, ..., UPPERBOUND
         // Print "Coza" if number is divisible by 3
         if ( ...... ) {
            System.out.print("Coza");
         }
         // Print "Loza" if number is divisible by 5
         if ( ...... ) {
            System.out.print(.....);
         }
         // Print "Woza" if number is divisible by 7
         ......
```

```
         // Print the number if it is not divisible by 3, 5 and 7 (i.e., it has not been processed above)
         if ( ..... ) {
            ......
         }
         // After processing the number, print a newline if number is divisible by 11;
         // else print a space
         if ( ..... ) {
            System.out.println();  // print newline
         } else {
            System.out.print( ..... );  // print a space
         }
      }
   }
}
```

Notes

1. You cannot use nested-if (if ... else if ... else if ... else) for this problem. It is because the tests are not mutually exclusive. For example, `15` is divisible by both `3` and `5`. Nested-if is only applicable if the tests are mutually exclusive.
2. The tests above looks messy. A better solution is to use a `boolean` flag to keep track of whether the number has been processed, as follows:

```
      final int LOWERBOUND = 1, UPPERBOUND = 110;
      boolean printed;
      for (int number = LOWERBOUND; number <= UPPERBOUND; ++number) {
         printed = false;  // init before processing each number
         // Print "Coza" if number is divisible by 3
         if ( ..... ) {
            System.out.print( ..... );
            printed = true;   // processed!
         }
         // Print "Loza" if number is divisible by 5
         if ( ..... ) {
            System.out.print( ..... );
            printed = true;   // processed!
         }
         // Print "Woza" if number is divisible by 7
         ......
         // Print the number if it has not been processed
         if (!printed) {
            ......
         }
         // After processing the number, print a newline if it is divisible by 11;
         // else, print a space
         ......
      }
```

4.6 `Fibonacci` (Decision & Loop)

Write a program called **`Fibonacci`** to print the first `20` Fibonacci numbers `F(n)`, where `F(n)=F(n-1)+F(n-2)` and `F(1)=F(2)=1`. Also compute their average. The output shall look like:

```
The first 20 Fibonacci numbers are:
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765
The average is 885.5
```

Hints

```
/**
 * Print first 20 Fibonacci numbers and their average
 */
public class Fibonacci {
```

```
   public static void main (String[] args) {
      int n = 3;          // The index n for F(n), starting from n=3, as n=1 and n=2 are pre-defined
      int fn;             // F(n) to be computed
      int fnMinus1 = 1;   // F(n-1), init to F(2)
      int fnMinus2 = 1;   // F(n-2), init to F(1)
      int nMax = 20;      // maximum n, inclusive
      int sum = fnMinus1 + fnMinus2;  // Need sum to compute average
      double average;

      System.out.println("The first " + nMax + " Fibonacci numbers are:");
      ......

      while (n <= nMax) {  // n starts from 3
            // n = 3, 4, 5, ..., nMax
         // Compute F(n), print it and add to sum
         ......
         // Increment the index n and shift the numbers for the next iteration
         ++n;
         fnMinus2 = fnMinus1;
         fnMinus1 = fn;
      }

      // Compute and display the average (=sum/nMax).
      // Beware that int/int gives int.
      ......
   }
}
```

Try

1. *Tribonacci numbers* are a sequence of numbers `T(n)` similar to *Fibonacci numbers*, except that a number is formed by adding the three previous numbers, i.e., `T(n)=T(n-1)+T(n-2)+T(n-3)`, `T(1)=T(2)=1`, and `T(3)=2`. Write a program called **`Tribonacci`** to produce the first twenty Tribonacci numbers.

4.7 `ExtractDigits` (Decision & Loop)

Write a program called **`ExtractDigits`** to extract each digit from an `int`, in the reverse order. For example, if the `int` is 15423, the output shall be "3 2 4 5 1", with a space separating the digits.

Hints

The *coding pattern* for extracting individual digits from an integer *n* is:

1. Use `(n % 10)` to extract the last (least-significant) digit.
2. Use `n = n / 10` to drop the last (least-significant) digit.
3. Repeat if `(n > 0)`, i.e., more digits to extract.

Take note that *n* is destroyed in the process. You may need to clone a copy.

```
      int n = ...;
      while (n > 0) {
         int digit = n % 10;  // Extract the least-significant digit
         // Print this digit
         ......
         n = n / 10;  // Drop the least-significant digit and repeat the loop
      }
```

5. Exercises on Input, Decision and Loop

5.1 Add2Integer (Input)

Write a program called `Add2Integers` that prompts user to enter two integers. The program shall read the two integers as `int`; compute their sum; and print the result. For example,

```
Enter first integer: 8
Enter second integer: 9
The sum is: 17
```

Hints

```
import java.util.Scanner;   // For keyboard input
/**
 * 1. Prompt user for 2 integers
 * 2. Read inputs as "int"
 * 3. Compute their sum in "int"
 * 4. Print the result
 */
public class Add2Integers {  // Save as "Add2Integers.java"
   public static void main (String[] args) {
      // Declare variables
      int number1, number2, sum;

      // Put up prompting messages and read inputs as "int"
      Scanner in = new Scanner(System.in);  // Scan the keyboard for input
      System.out.print("Enter first integer: ");  // No newline for prompting message
      number1 = in.nextInt();                     // Read next input as "int"
      ......
      in.close();  // Close Scanner

      // Compute sum
      sum = ......

      // Display result
      System.out.println("The sum is: " + sum);   // Print with newline
   }
}
```

5.2 SumProductMinMax3 (Arithmetic & Min/Max)

Write a program called `SumProductMinMax3` that prompts user for three integers. The program shall read the inputs as `int`; compute the sum, product, minimum and maximum of the three integers; and print the results. For examples,

```
Enter 1st integer: 8
Enter 2nd integer: 2
Enter 3rd integer: 9
The sum is: 19
The product is: 144
The min is: 2
The max is: 9
```

Hints

```
      // Declare variables
      int number1, number2, number3;  // The 3 input integers
      int sum, product, min, max;     // To compute these

      // Prompt and read inputs as "int"
      Scanner in = new Scanner(System.in);  // Scan the keyboard
      ......
      ......
      in.close();
```

```
// Compute sum and product
sum = ......
product = ......

// Compute min
// The "coding pattern" for computing min is:
// 1. Set min to the first item
// 2. Compare current min with the second item and update min if second item is smaller
// 3. Repeat for the next item
min = number1;        // Assume min is the 1st item
if (number2 < min) {  // Check if the 2nd item is smaller than current min
   min = number2;     // Update min if so
}
if (number3 < min) {  // Continue for the next item
   min = number3;
}

// Compute max - similar to min
......

// Print results
......
```

Try

1. Write a program called `SumProductMinMax5` that prompts user for five integers. The program shall read the inputs as `int`; compute the sum, product, minimum and maximum of the five integers; and print the results. Use five `int` variables: `number1`, `number2`, ..., `number5` to store the inputs.

5.3 CircleComputation (double & printf())

Write a program called **`CircleComputation`** that prompts user for the radius of a circle in floating point number. The program shall read the input as `double`; compute the diameter, circumference, and area of the circle in `double`; and print the values rounded to 2 decimal places. Use System-provided constant `Math.PI` for pi. The formulas are:

```
diameter = 2.0 * radius;
area = Math.PI * radius * radius;
circumference = 2.0 * Math.PI * radius;
```

Hints

```
// Declare variables
double radius, diameter, circumference, area;  // inputs and results - all in double
......

// Prompt and read inputs as "double"
System.out.print("Enter the radius: ");
radius = in.nextDouble();  // read input as double

// Compute in "double"
......

// Print results using printf() with the following format specifiers:
//   %.2f for a double with 2 decimal digits
//   %n for a newline
System.out.printf("Diameter is: %.2f%n", diameter);
......
```

Try

1. Write a program called **`SphereComputation`** that prompts user for the `radius` of a sphere in floating point number. The

program shall read the input as `double`; compute the volume and surface area of the sphere in `double`; and print the values rounded to 2 decimal places. The formulas are:

```
surfaceArea = 4 * Math.PI * radius * radius;
volume = 4 /3 * Math.PI * radius * radius * radius;   // But this does not work in programming?! Why?
```

Take note that you cannot name the variable `surface area` with a space or `surface-area` with a dash. Java's naming convention is `surfaceArea`. Other languages recommend `surface_area` with an underscore.

2. Write a program called **`CylinderComputation`** that prompts user for the base `radius` and `height` of a cylinder in floating point number. The program shall read the inputs as `double`; compute the base area, surface area, and volume of the cylinder; and print the values rounded to 2 decimal places. The formulas are:

```
baseArea = Math.PI * radius * radius;
surfaceArea = 2.0 * Math.PI * radius + 2.0 * baseArea;
volume = baseArea * height;
```

5.4 Swap2Integers

Write a program called `Swap2Integers` that prompts user for two integers. The program shall read the inputs as `int`, save in two variables called `number1` and `number2`; swap the contents of the two variables; and print the results. For examples,

```
Enter first integer: 9
Enter second integer: -9
After the swap, first integer is: -9, second integer is: 9
```

Hints

To swap the contents of two variables `x` and `y`, you need to introduce a temporary storage, say `temp`, and do: `temp ⇐ x`; `x ⇐ y`; `y ⇐ temp`.

5.5 IncomeTaxCalculator (Decision)

The progressive income tax rate is mandated as follows:

Taxable Income	Rate (%)
First $20,000	0
Next $20,000	10
Next $20,000	20
The remaining	30

For example, suppose that the taxable income is `$85000`, the income tax payable is `$20000*0% + $20000*10% + $20000*20% + $25000*30%`.

Write a program called **`IncomeTaxCalculator`** that reads the taxable income (in `int`). The program shall calculate the income tax payable (in `double`); and print the result rounded to 2 decimal places. For examples,

```
Enter the taxable income: $41234
The income tax payable is: $2246.80

Enter the taxable income: $67891
The income tax payable is: $8367.30

Enter the taxable income: $85432
The income tax payable is: $13629.60

Enter the taxable income: $12345
The income tax payable is: $0.00
```

Hints

```
// Declare constants first (variables may use these constants)
// The keyword "final" marked these as constant (i.e., cannot be changed).
// Use uppercase words joined with underscore to name constants
final double TAX_RATE_ABOVE_20K = 0.1;
final double TAX_RATE_ABOVE_40K = 0.2;
final double TAX_RATE_ABOVE_60K = 0.3;

// Declare variables
int taxableIncome;
double taxPayable;
......

// Compute tax payable in "double" using a nested-if to handle 4 cases
if (taxableIncome <= 20000) {         // [0, 20000]
   taxPayable = ......;
} else if (taxableIncome <= 40000) {  // [20001, 40000]
   taxPayable = ......;
} else if (taxableIncome <= 60000) {  // [40001, 60000]
   taxPayable = ......;
} else {                              // [60001, ]
   taxPayable = ......;
}
// Alternatively, you could use the following nested-if conditions
// but the above follows the table data
//if (taxableIncome > 60000) {          // [60001, ]
//   ......
//} else if (taxableIncome > 40000) {   // [40001, 60000]
//   ......
//} else if (taxableIncome > 20000) {   // [20001, 40000]
//   ......
//} else {                              // [0, 20000]
//   ......
//}

// Print results rounded to 2 decimal places
System.out.printf("The income tax payable is: $%.2f%n", ...);
```

Try

Suppose that a `10%` tax rebate is announced for the income tax payable, capped at `$1,000`, modify your program to handle the tax rebate. For example, suppose that the tax payable is `$12,000`, the rebate is `$1,000`, as `10%` of `$12,000` exceed the cap.

5.6 `IncomeTaxCalculatorWithSentinel` (Decision & Loop)

Based on the previous exercise, write a program called `IncomeTaxCalculatorWithSentinel` which shall repeat the calculation until user enter `-1`. For example,

```
Enter the taxable income (or -1 to end): $41000
The income tax payable is: $2200.00

Enter the taxable income (or -1 to end): $62000
The income tax payable is: $6600.00

Enter the taxable income (or -1 to end): $73123
The income tax payable is: $9936.90

Enter the taxable income (or -1 to end): $84328
The income tax payable is: $13298.40

Enter the taxable income: $-1
bye!
```

The `-1` is known as the *sentinel value*. (Wiki: In programming, a *sentinel value*, also referred to as a flag value, trip value, rogue value, signal value, or dummy data, is a special value which uses its presence as a condition of termination.)

Hints

The *coding pattern* for handling input with sentinel value is as follows:

```
// Declare constants first
final int SENTINEL = -1;    // Terminating value for input
......

// Declare variables
int taxableIncome;
double taxPayable;
......

// Read the first input to "seed" the while loop
System.out.print("Enter the taxable income (or -1 to end): $");
taxableIncome = in.nextInt();

while (taxableIncome != SENTINEL) {
   // Compute tax payable
   ......
   // Print result
   ......

   // Read the next input
   System.out.print("Enter the taxable income (or -1 to end): $");
   taxableIncome = in.nextInt();
      // Repeat the loop body, only if the input is not the SENTINEL value.
      // Take note that you need to repeat these two statements inside/outside the loop!
}
System.out.println("bye!");
```

Take note that we repeat the input statements inside and outside the loop. Repeating statements is NOT a good programming practice. This is because it is easy to repeat (Cntl-C/Cntl-V), but hard to maintain and synchronize the repeated statements. In this case, we have no better choices!

5.7 PensionContributionCalculator (Decision)

Both the employer and the employee are mandated to contribute a certain percentage of the employee's salary towards the employee's pension fund. The rate is tabulated as follows:

Employee's Age	Employee Rate (%)	Employer Rate (%)
55 and below	20	17
above 55 to 60	13	13
above 60 to 65	7.5	9
above 65	5	7.5

However, the contribution is subjected to a salary ceiling of `$6,000`. In other words, if an employee earns `$6，800`, only `$6，000` attracts employee's and employer's contributions, the remaining `$800` does not.

Write a program called **`PensionContributionCalculator`** that reads the monthly salary and age (in `int`) of an employee. Your program shall calculate the employee's, employer's and total contributions (in `double`); and print the results rounded to 2 decimal places. For examples,

```
Enter the monthly salary: $3000
Enter the age: 30
The employee's contribution is: $600.00
The employer's contribution is: $510.00
```

```
The total contribution is: $1110.00
```

Hints

```
// Declare constants
final int SALARY_CEILING = 6000;
final double EMPLOYEE_RATE_55_AND_BELOW = 0.2;
final double EMPLOYER_RATE_55_AND_BELOW = 0.17;
final double EMPLOYEE_RATE_55_TO_60 = 0.13;
final double EMPLOYER_RATE_55_TO_60 = 0.13;
final double EMPLOYEE_RATE_60_TO_65 = 0.075;
final double EMPLOYER_RATE_60_TO_65 = 0.09;
final double EMPLOYEE_RATE_65_ABOVE = 0.05;
final double EMPLOYER_RATE_65_ABOVE = 0.075;

// Declare variables
int salary, age;     // to be input
int contributableSalary;
double employeeContribution, employerContribution, totalContribution;
......

// Check the contribution cap
contributableSalary = ......

// Compute various contributions in "double" using a nested-if to handle 4 cases
if (age <= 55) {         // 55 and below
   ......
} else if (age <= 60) {  // (60, 65]
   ......
} else if (age <= 65) {  // (55, 60]
   ......
} else {                 // above 65
   ......
}
// Alternatively,
//if (age > 65) ......
//else if (age > 60) ......
//else if (age > 55) ......
//else ......
```

5.8 PensionContributionCalculatorWithSentinel (Decision & Loop)

Based on the previous `PensionContributionCalculator`, write a program called **`PensionContributionCalculatorWithSentinel`** which shall repeat the calculations until user enter -1 for the salary. For examples,

```
Enter the monthly salary (or -1 to end): $5123
Enter the age: 21
The employee's contribution is: $1024.60
The employer's contribution is: $870.91
The total contribution is: $1895.51

Enter the monthly salary (or -1 to end): $5123
Enter the age: 64
The employee's contribution is: $384.22
The employer's contribution is: $461.07
The total contribution is: $845.30

Enter the monthly salary (or -1 to end): $-1
bye!
```

Hints

```
      // Read the first input to "seed" the while loop
      System.out.print("Enter the monthly salary (or -1 to end): $");
      salary = in.nextInt();

      while (salary != SENTINEL) {
         // Read the remaining
         System.out.print("Enter the age: ");
         age = in.nextInt();

         ......
         ......

         // Read the next input and repeat
         System.out.print("Enter the monthly salary (or -1 to end): $");
         salary = in.nextInt();
      }
```

5.9 SalesTaxCalculator (Decision & Loop)

A sales tax of 7% is levied on all goods and services consumed. It is also mandatory that all the price tags should include the sales tax. For example, if an item has a price tag of $107, the actual price is $100 and $7 goes to the sales tax.

Write a program using a loop to continuously input the tax-inclusive price (in double); compute the actual price and the sales tax (in double); and print the results rounded to 2 decimal places. The program shall terminate in response to input of -1; and print the total price, total actual price, and total sales tax. For examples,

```
Enter the tax-inclusive price in dollars (or -1 to end): 107
Actual Price is: $100.00, Sales Tax is: $7.00

Enter the tax-inclusive price in dollars (or -1 to end): 214
Actual Price is: $200.00, Sales Tax is: $14.00

Enter the tax-inclusive price in dollars (or -1 to end): 321
Actual Price is: $300.00, Sales Tax is: $21.00

Enter the tax-inclusive price in dollars (or -1 to end): -1
Total Price is: $642.00
Total Actual Price is: $600.00
Total Sales Tax is: $42.00
```

Hints

```
      // Declare constants
      final double SALES_TAX_RATE = 0.07;
      final int SENTINEL = -1;        // Terminating value for input

      // Declare variables
      double price, actualPrice, salesTax;  // inputs and results
      double totalPrice = 0.0, totalActualPrice = 0.0, totalSalesTax = 0.0;  // to accumulate
      ......

      // Read the first input to "seed" the while loop
      System.out.print("Enter the tax-inclusive price in dollars (or -1 to end): ");
      price =  in.nextDouble();

      while (price != SENTINEL) {
         // Compute the tax
         ......
         // Accumulate into the totals
         ......
         // Print results
         ......
```

```
      // Read the next input and repeat
      System.out.print("Enter the tax-inclusive price in dollars (or -1 to end): ");
      price =  in.nextDouble();
   }
   // print totals
   ......
```

5.10 ReverseInt (Loop with Modulus/Divide)

Write a program that prompts user for a positive integer. The program shall read the input as `int`; and print the "reverse" of the input integer. For examples,

```
Enter a positive integer: 12345
The reverse is: 54321
```

Hints

Use the following *coding pattern* which uses a while-loop with repeated modulus/divide operations to extract and drop the last digit of a positive integer.

```
   // Declare variables
   int inNumber;   // to be input
   int inDigit;    // each digit
   ......

   // Extract and drop the "last" digit repeatably using a while-loop with modulus/divide operations
   while (inNumber > 0) {
      inDigit = inNumber % 10; // extract the "last" digit
      // Print this digit (which is extracted in reverse order)
      ......
      inNumber /= 10;          // drop "last" digit and repeat
   }
   ......
```

5.11 SumOfDigitsInt (Loop with Modulus/Divide)

Write a program that prompts user for a positive integer. The program shall read the input as `int`; compute and print the sum of all its digits. For examples,

```
Enter a positive integer: 12345
The sum of all digits is: 15
```

Hints

See "ReverseInt".

5.12 InputValidation (Loop with boolean flag)

Your program often needs to validate the user's inputs, e.g., marks shall be between 0 and 100.

Write a program that prompts user for an integer between `0-10` or `90-100`. The program shall read the input as `int`; and repeat until the user enters a valid input. For examples,

```
Enter a number between 0-10 or 90-100: -1
Invalid input, try again...
Enter a number between 0-10 or 90-100: 50
Invalid input, try again...
Enter a number between 0-10 or 90-100: 101
Invalid input, try again...
Enter a number between 0-10 or 90-100: 95
You have entered: 95
```

Hints

Use the following *coding pattern* which uses a do-while loop controlled by a `boolean` flag to do input validation. We use a do-while instead of while-do loop as we need to execute the body to prompt and process the input at least once.

```
// Declare variables
int numberIn;      // to be input
boolean isValid;   // boolean flag to control the loop
......

// Use a do-while loop controlled by a boolean flag
// to repeatably read the input until a valid input is entered
isValid = false;   // default assuming input is not valid
do {
   // Prompt and read input
   ......

   // Validate input by setting the boolean flag accordingly
   if (numberIn ......) {
      isValid = true;   // exit the loop
   } else {
      System.out.println(......);  // Print error message and repeat
   }
} while (!isValid);
......
```

5.13 AverageWithInputValidation (Loop with boolean flag)

Write a program that prompts user for the mark (between `0-100` in `int`) of 3 students; computes the average (in `double`); and prints the result rounded to 2 decimal places. Your program needs to perform input validation. For examples,

```
Enter the mark (0-100) for student 1: 56
Enter the mark (0-100) for student 2: 101
Invalid input, try again...
Enter the mark (0-100) for student 2: -1
Invalid input, try again...
Enter the mark (0-100) for student 2: 99
Enter the mark (0-100) for student 3: 45
The average is: 66.67
```

Hints