Write method

Philosophy behind method

- Someone needs to do the work. If the instructions are not in main method, then they should be in the body of some other method.
- Methods are independent of each other. So no method can be nested inside another method.
- To design a method, find out what the input of the method is, what the method will do, and what the return of that method is (a method may return nothing).
- Input parameters of a method is like variable declaration. If you have more than one parameter, each declaration is separated by a comma.

Philosophy behind method: do not underdo

- A method should not underdo the task – as an employee, you need to fulfill your duty.
- If there is a return, make sure that you will return a value in every possible case.
- Only when a method is called, its input parameters are initialized – for example, if you do not use a bread machine, those input slots will not be filled with actual flour and nuts.

Philosophy behind method: do not overdo

- A method should not overdo more than necessary task – as an employee, you do not want to work extra for nothing.
- Unless a method is specialized in input – for example, ensure the user enter an integer in [0, 100] -- we **normally** do not do interactive input inside a method.

Philosophy behind method: do not overdo II

- Unless a method is specialized in output, do not print intermediate contents to screen. For example, sqrt method does not print out “the square root of xyz is blah blah” to the screen.
  - Do what you are asked to do. When we use a recipe to make a cookie, we may print out “we make a cookie”, or we may take a picture of it, or we may throw out the cookie if it does not taste good. It is not the designer of cookie recipe to care how people uses that product, the only thing he/she needs to care is to write ingredients (inputs) and instructions to make that cookie.

Caller and Callee

- Method that calls other methods is caller while the method being called is callee.
  - For example, if you use sqrt method in main method, then main method is caller while sqrt is callee.
  - A method can call more than one method. And a method being called by main can then call another method.

How to call a method

1. Caller saves local variables and returning point. Then pass the values to the input parameters, if any, in callee.

2. Caller yields control to callee.

3. Callee runs code.

4. After callee finishes, it returns result (if necessary) and yields control back to caller.

5.Caller continue to run code from the returning point.

6. It is like switch scenes between stages when someone works in a field, then he/she has a dream, and wakes up from the dream and continues to work in the field.

Exercise I

- Given two integers a and b, return the result of a raised to the power of b. (Hint: you may need to consider when b is positive, zero, or negative)
- Given two integers a and b, return which one is big.
- Given a numerical grade (can contain decimal numbers), return its letter grade (’A’, ‘B’, ‘C’, ‘D’, ‘F’). If the numerical grade is smaller than zero or larger than 100, return a char with space.

Exercise II

- Write a method to return the maximum for three given integers.
- Write a method, for a given string and a given char, find out the number of occurrence of that char in the string.

Exercise III

- Write a method to find out whether an integer is prime or not. Then call this method in main method to print out all the prime integers from 1 to 100.
- By definition, a prime is a positive integer that has only 1 and itself as factors.
  - So 1 is not a prime by definition.
- Given an integer n, define 1 and n as trivial factors of n. The factors that are not trivial are called non-trivial factors.
  - So a prime has no non-trivial factors.
  - TODO: find out whether an integer has non-trivial factors or not.

Relationship between non-trivial factor and prime

- Non-trivial factor candidates of integer n are _____.
- How to find out whether a candidate is ACTUALLY a factor of n?
- What if we find the first non-trivial factor of n?
  - What are non-trivial factor candidates of 25?
  - What is the first non-trivial factor of 25?
- What if none of the non-trivial factor candidates is a factor of n?
  - What are non-trivial factor candidates of 11?

Existence of non-trivial factors or not

- If n has at least a non-trivial factor, then n is _____.
- If n has NO non-trivial factor, then n is _____.

Given a batch of eggs,

- If there is at least a bad egg, then this batch of eggs does not pass quality check.
- If there is no bad egg, then this batch of eggs pass quality check.

Existence of non-trivial factor or not: II

- At least one non-trivial factor

Prime? NO

- None non-trivial factor

Prime? Yes

- Prime number does not allow existence of non-trivial factors.

- At least one bad egg

- No bad egg

- Quality control does not allow existence of bad eggs.

Is the following pseudo-code correct?

```
bool QAPassed(eggs a) {
    for each egg in a {
        if the current egg is bad
            return false;
        else true;
    }
}
```

```
bool QAPassed(eggs a) {
    for each egg in a {
        if the current egg is bad
            return false;
    }

    return true;
}
```

candidates for non-trivial smaller factors of n

- Suppose n is 11, then we would check whether 2, 3, …, 5 (do we need to continue from 6?) are factors of 11 or not. Said differently, given n, the possible candidates of non-trivial factors are in [2, n/2].
- In fact, we can do better. Non-trivial smaller factors are in [2, $\sqrt{n}$]. Why?
  - Factors appear in a pair. Once we know one, we can derive the other. For example, suppose n is 16, if 2 is a factor of 16, then 16/2 is also a factor of 16.
  - Possible ways to factor 16 are: 2 * 8, 4 * 4, 8 * 2. You can see 2 * 8 and 8 * 2 are redundant.
  - So we only need to look at the smaller factor among any factoring. Prove by contradiction that the biggest smaller factor of n cannot be larger than $\sqrt{n}$.
  - Note that [2, $\sqrt{n}$] is a valid range only when n >= 4. Consider n = 1,2,3 separately.

candidates for non-trivial factors of n

- Given an integer n, non-trivial smaller factors are located in $[2, \sqrt{n}]$.
  - When to use this above fact? When we want to find out whether n is a prime or not.
- Given an integer n, non-trivial factors are located in [2, n/2].
  - When to use this fact? Find out whether an integer if perfect or not.
  - An integer n is perfect if all its non-trivial factors added up to n itself?
    - Is 1 perfect?
    - Is any of 2, 3, 4, 5 perfect?
    - Is 6 perfect?
    - Write a method to find out whether an integer is perfect or not. Then use the above method to find out all perfect integers in [1, 1000].

Common mistake

- Print ≠ return
- If return type is not void, make sure return a value fit for the type in every branch.