[AT Ralph] Basic Java Practice

Target: Finish 4 Java program

For Challenge: Complete addition 1 Java program

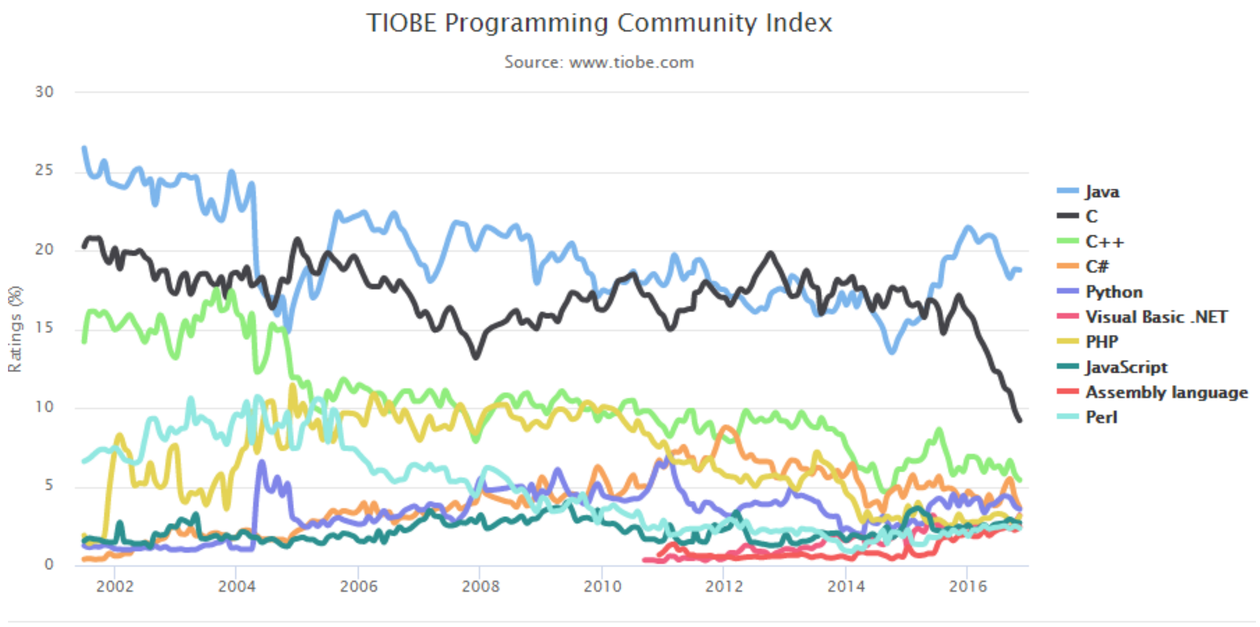
For Cha-Challenge: Finish 1 large calculation Java program in 5 seconds

Part 1: Java Introduce

- Java is easy to learn.
  Java was designed to be easy to use and is therefore easy to write, compile, debug, and learn than other programming languages.
- Java is object-oriented.
  This allows you to create modular programs and reusable code.
- Java is platform-independent.
  One of the most significant advantages of Java is its ability to move easily from one computer system to another. The ability to run the same program on many different systems is crucial to World Wide Web software, and Java succeeds at this by being platform-independent at both the source and binary levels.

TIOBE program language trend:

http://www.tiobe.com/tiobe-index/



Part 2: Variables & Basic Data Type

byte, short, int, long, double, boolean, char, string, reference

- **byte**: The `byte` data type is an 8-bit signed two's complement integer. It has a minimum value of -128 and a maximum value of 127 (inclusive). The `byte` data type can be useful for saving memory in large arrays, where the memory savings actually matters. They can also be used in place of `int` where their limits help to clarify your code; the fact that a variable's range is limited can serve as a form of documentation.
- **short**: The `short` data type is a 16-bit signed two's complement integer. It has a minimum value of -32,768 and a maximum value of 32,767 (inclusive). As with `byte`, the same guidelines apply: you can use a `short` to save memory in large arrays, in situations where the memory savings actually matters.
- **int**: By default, the `int` data type is a 32-bit signed two's complement integer, which has a minimum value of -2^{31} and a maximum value

of 2^{31}-1. In Java SE 8 and later, you can use the `int` data type to represent an unsigned 32-bit integer, which has a minimum value of 0 and a maximum value of 2^{32}-1. Use the Integer class to use `int` data type as an unsigned integer. See the section The Number Classes for more information. Static methods like `compareUnsigned`, `divideUnsigned` etc have been added to the `Integer` class to support the arithmetic operations for unsigned integers.
- **long**: The `long` data type is a 64-bit two's complement integer. The signed long has a minimum value of -2^{63} and a maximum value of 2^{63}-1. In Java SE 8 and later, you can use the `long` data type to represent an unsigned 64-bit long, which has a minimum value of 0 and a maximum value of 2^{64}-1. Use this data type when you need a range of values wider than those provided by `int`. The `Long` class also contains methods like `compareUnsigned`, `divideUnsigned` etc to support arithmetic operations for unsigned long.
- **float**: The `float` data type is a single-precision 32-bit IEEE 754 floating point. Its range of values is beyond the scope of this discussion, but is specified in the Floating-Point Types, Formats, and Values section of the Java Language Specification. As with the recommendations for `byte` and `short`, use a `float` (instead of `double`) if you need to save memory in large arrays of floating point numbers. This data type should never be used for precise values, such as currency. For that, you will need to use the java.math.BigDecimal class instead. Numbers and Strings covers `BigDecimal` and other useful classes provided by the Java platform.
- **double**: The `double` data type is a double-precision 64-bit IEEE 754 floating point. Its range of values is beyond the scope of this discussion, but is specified in the Floating-Point Types, Formats, and Values section of the Java Language Specification. For decimal values, this data type is generally the default choice. As mentioned above, this data type should never be used for precise values, such as currency.
- **boolean**: The `boolean` data type has only two possible values: `true` and `false`. Use this data type for simple flags that track true/false conditions. This data type represents one bit of information, but its "size" isn't something that's precisely defined.
- **char**: The `char` data type is a single 16-bit Unicode character. It has a minimum value of `'\u0000'` (or 0) and a maximum value of `'\uffff'` (or 65,535 inclusive).

```
int decimal = 100;
int octal = 0144;
int hexa =  0x64;
```

Practice 1:

set variables with basic java data type and print them:

int a = -1;
long b = 258258258258258L;
double c = 3.1415926;
char d = 66;
String e = "I love automation";

Part 3: Primitive Operators

*+ - * / % ++ --*

```
// Add two literal values
int result = 5 + 5;

// Add two variables
int a = 5;
int b = 6;
int result = a + b;

// Add two variables and a literal
int result = a + b + 15;
```

Practice 2:

*Give me the result of this (222*33+(11111)/44-666) mod 55=??, ignore the precision, give me the integer result*

Part 4: Conditional And Loops

if else, else

```
void applyBrakes() {
    if (isMoving) {
        currentSpeed--;
    } else {
        System.err.println("The bicycle has already stopped!");
    }
}
```

switch case

```
public class SwitchDemo {
    public static void main(String[] args) {

        int month = 8;
        String monthString;
        switch (month) {
            case 1:  monthString = "January";
                     break;
            case 2:  monthString = "February";
                     break;
            case 3:  monthString = "March";
                     break;
            case 4:  monthString = "April";
                     break;
            case 5:  monthString = "May";
                     break;
            case 6:  monthString = "June";
                     break;
            case 7:  monthString = "July";
                     break;
            case 8:  monthString = "August";
                     break;
            case 9:  monthString = "September";
                     break;
            case 10: monthString = "October";
                     break;
            case 11: monthString = "November";
                     break;
            case 12: monthString = "December";
                     break;
            default: monthString = "Invalid month";
                     break;
        }
        System.out.println(monthString);
    }
}
```

while

```
class WhileDemo {
    public static void main(String[] args){
        int count = 1;
        while (count < 11) {
            System.out.println("Count is: " + count);
            count++;
        }
    }
}
```

for

```
class ForDemo {
    public static void main(String[] args){
         for(int i=1; i<11; i++){
              System.out.println("Count is: " + i);
         }
    }
}
```

Practice 3:

Sum the integer number from 1 to 20161202, print the right result

Part 5: How To Write A Function

public , protected, private

Practice 4:

You take the Mathematics final exam, the full score is 100, is you get not less than 60, you pass the exam, if you get less than 60 score, you fail.

write a function areYouPassExam(int score) to return you are pass or fail the exam.

Challenge:

Practice 5:

Write a method can find the maximum prime number from (33333 - 44444).

Cha-Challenge:

Practice 6:

Count how many prime number from (0-33333333), can you finish the running in 5 seconds

Books recommand:

1.<u>Thinking in Java</u>

2.<u>Core Java</u>

3.<u>HeadFirst</u>