CSci 1133
Lab Exercise 13

Java is a powerful programming language and is arguably the most popular and widely used computer language today. If you've seen the news lately about driverless cars, the first prototypes were actually coded in Java!

In this lab, you will practice working with a few simple Java programs. As you've learned in lecture, there are several important differences between Python and Java. These range from type restrictions (Java is "statically typed" while Python is "dynamically typed"), to things as simple as having to add a semicolon after each line.

Warm up

Let's get started. One of the easiest ways to learn the nuances of a new programming language is to convert a simple program from a language that you already know into the language you are trying to learn. Take comfort in the knowledge that the semantic *principles* are identical, regardless the language!

Consider the following Python program that determines all the *perfect numbers* between 1 and some upper-bound provided by the user. Recall that a *perfect number* is one in which the sum of all of its divisors (except for the number itself) equals itself (e.g. 6 = 1 + 2 + 3). Convert this Python program to its Java equivalent. Name the Java main class `PerfectNum`.

```
def main():
    limit = int(input("Enter the upper limit: "))
    n=1
    while n <= limit:
        i=1
        factorsum = 0
        while i<n:
            if n%i == 0:
                factorsum += i
            i += 1
        if factorsum == n:
            print(n,"is a perfect number!")
        n+=1

if __name__ == '__main__':
    main()
```

Stretch

Now, let's try something more challenging. The following Python program inputs Roman Numerals in the form of Strings and outputs their (decimal) value. It consists of two pure functions and a non-pure `main` function:

```
def convertDigit(letter):
    digits = "IVXLCDM"
    values = [1,5,10,50,100,500,1000]
    i = 0
    while i < len(digits) and letter != digits[i]:
        i += 1
    if i < len(digits):
        return values[i]
    return 0

def convertRoman(istr):
    lastvalue = 0
    decimalval = 0
    for ch in istr:
        currentvalue = convertDigit(ch)
        if lastvalue < currentvalue:
            decimalval -= lastvalue
        else:
            decimalval += lastvalue
        lastvalue = currentvalue
    return decimalval + lastvalue

def main():
    done = False
    while not done:
        romannumeral = input("Enter a Roman Numeral as a string: ")
        if len(romannumeral) == 0:
            done = True
        else:
            print("Decimal value:",convertRoman(romannumeral))
```

Convert this Python program to Java. Here are some helpful "hints":

- Name the Java main class: `ConvertRoman`
- User the Scanner `.nextLine()` method to input a `String` from the keyboard
- To create "pure functions" in Java, you must define them as `public static` methods in the main class and be sure to give them the proper return-value type!
- The literal values for the boolean states *True* and *False* are `true` and `false`, respectively (lower-case keywords)
- To determine the length of a String in Java, use the String method: `.length()`
- Java does not provide an index *operator* for strings (i.e., [ ]). To index individual characters in a String you must use the String method: `.charAt(`*index*`)` where *index* is an integer offset relative to zero.
- If you want to use the Java "foreach" iterator loop variant on a String, you must first convert the String object to a character array. This is easily accomplished using the String method: `toCharArray()`

Java object classes

Since everything in Java is defined in a class, we're already familiar with defining class methods. If we remove the static declaration, then the method definition functions exactly like the Python methods you've been constructing for some time. They must be called by binding to an instance of the class (i.e., object).

Let's take a look at a simple Java class and extend it.

First, copy-paste the following Java code into a new text file, and save it as `Calculator.java`.

```
public class Calculator {
	private double value1;
	private double value2;

	public Calculator(double v1, double v2) {
		value1 = v1;
		value2 = v2;
	}

	public void setValue(double v1, double v2) {
		value1 = v1;
		value2 = v2;
	}

	public double sum() {
		return value1 + value2;
	}

	public double difference() {
		return value1 - value2;
	}


	public static void main(String[] args) {
		Calculator c = new Calculator(0,0);
		c.setValue(3, 4);
		System.out.println("sum " + c.sum());
		System.out.println("diffrence " + c.difference());
	}
}
```

There are several things in this bit of code that we haven't seen before. First, there's a public method definition with the *same name* as the class (`Calculator`), and it appears to violate the type-specification rules! Actually, this is how we specify the class *constructor method* in Java. The constructor method is called when a new class instance (i.e., object) is created. It's similar to the `__init__` method in Python, except we always have to *explicitly* construct objects in Java using the `new` operator. When the statement `new Calculator(0,0)` is called in the `main` method, the `Calculator` constructor method will be called to initialize the object's instance variables.

Note that the instance variables of this class are declared prior to the method definitions and contain the keyword private. Recall that Python does not support (purely) private instance variables and does not require their declaration. However Java, and most other strongly-typed languages do support purely private variables and methods. The keyword private is used to enforce encapsulation and prohibit access to the declared variables of the class.

Do the following:
1) Add two new public Java methods to the class:

- `product()` returns the floating-point product of the stored values
- `quotient()` returns the floating-point quotient of value1 divided by value2

2) Test your new methods by adding code to the main method of the class.

3) Construct an array of 20 Calculator objects with the values, (1,1), (2,2), (3,3), ..., (20,20) using a loop and then determine/print the sum of the products of all the objects in the array.

Workout

Bank Account

Create a Java class that represents a person's bank account. The class must contain the following (private) instance variables:

- `double balance`
- `String ownerName`
- `String accountNumber`

Implement the following methods:

- Accessor methods ("getters" in Java parlance) for each of the instance variables
- Mutator methods ("setters" in Java parlance) for each of the instance variables
- Mutator: `deposit(double amount)` That will update the account balance
- Mutator: `withdraw(double amount)` That will update the account balance
- Accessor: `compoundInterest(int numYears, double rate, int numCompounds)` that returns the balance after `numYears` of compounding the account balance at a rate of `rate` for `numCompounds` compounds per year, according to the formula:

$$A = P(1 + \frac{r}{n})^{nt}$$

  where *P* is the *principal* or the original amount of money, *r* is the *annual* interest rate, *t* is number of years, and n is the number of times to compound annually.

- `simpleInterest(int numYears, double rate)`
  Returns the balance after `numYears` of accruing simple interest at a rate of `rate`, according to :

$$A = P(1 + rt)$$

  where *P* is the principal amount, *r* is the annual interest rate and *t* is the number of years

Provide a `main` method to test your class .