Exercise 1
Learning Java syntax with some exercises

1 The next second...

In a new class Syntax1, create a function: nextSecond(hour, minute, second)
that prints the time at the next second. You will then call this function from the main(),
and test it with several arguments.

The Python code of the nextSecond function is as follow:

```python
def nextSecond(h,m,s):
    if (s<59):
        print("It ’ s %d : %d : %d\%(h,m,s+1))
    elif (m<59):
        print("It ’ s %d : %d : 00\%(h,m+1))
    elif (h<23):
        print("It ’ s %d : 00 : 00\%(h+1))
    else:
        print("It ’ s 00 : 00 : 00")
```

and your Java code will be inspired by:

```java
public class Syntaxe1 {

    public static void main(String [] args) {
        nextSecond (8,10,55);
        nextSecond (8,10,12);
        /* complete hehe*/
    }

    static void nextSecond(int h,int m, int s){
        /* complete hehe */
    }
}
```

Warning: pay attention to borderline cases.

You need to create tests where:
(i) the result only increments by one second (e.g after 8h 12m 3s, we have 8h 12m 4s)
(ii) the result changes the minute (e.g after 8h 12m 59s we have 8h 13m 0s)
(iii) the result changes the hour, and one where the result changes the day.
(iv) also, do not hesitate to create a test at the beginning of the method which checks the
entry parameters: the hours need to be between 0 and 23, the minutes and second should
be each between 0 and 59.
2 Euclidean division

In a new class Syntax2, create a function: `euclideanDivision(dividend, divisor)` which prints the result (quotient and remainder) of the euclidean division of the dividend by the divisor. You will call this function from the main() and test it with several arguments.

The Python code of the function `euclideanDivision` is as follow:

```
def euclideanDivision(num, den):
    rest = num
    quotient = 0
    while (rest >= den):
        quotient = quotient + 1
        rest = rest - den
    print "The quotient is %d, and the rest %d"%(quotient, rest)

euclideanDivision(1264,245)
euclideanDivision(99987456,14587)
```

and your Java code will be inspired by:

```
public class Syntaxe2 {
    public static void main(String[] args) {
        euclideanDivision(54, 8);
        euclideanDivision(1264, 245);
        euclideanDivision(99987456, 14587);
    }
    static void DivisionE(int num, int den){
        int rest = /* complete */;
        int quotient = /* complete */;
        /*complete here */
    }
}
```

Warning: pay attention to borderline cases.

Do not hesitate to create a test at the beginning of the method which checks the input parameters. In fact, it is necessary that both the dividend and divisor are positive (although the dividend may be zero) if you don’t want to be caught in an infinite loop.
3 Sum of squared integers

In a new class Syntax3, create two functions:
sumSquaredFor(n)
and:
sumSquaredWhile(n)
which computes both the sum of the squared n first integers, with respectively a for loop and a while loop. You will call these two functions from the main() and test it with several arguments.

The Python code of the two functions is as follow:

```python
def sumSquaredFor(n):
    somme=0
    for elt in range(0 ,n+1):
        sum=sum + elt * elt
    return sum
def sumSquaredWhile(n):
    sum=0
    elt=0
    while elt<=n:
        sum=sum + elt * elt
        elt=elt + 1
    return sum

print (sumSquaredFor(4))
print (sumSquaredWhile(4))
```

and your Java code will be inspired by :

```java
public class Syntaxe3 {
    public static void main(String[] args){
        System.out.println(sumSquaredFor(4))
        System.out.println(sumSquaredWhile(4));
        /* Complete with other tests. */
    }
    static int sumSquaredWhile(int n){
        /* complete */
    }
    /* Complete with sumSquaredWhile method */
}
```

Warning: pay attention to borderline cases.
Do not hesitate to add a tests at the beginning of the method which checks the input parameters. For example, n should be positive, otherwise you will not reach it by incrementing by one each iteration.