Preparation

Download links and other details can be found via the course web page:

`http://www.win.tue.nl/~wstomv/edu/sc-hse/`

1. In the book *Introduction to Programming Using Java* by David Eck, read §4.7, 8.1.
2. Download the files for Series 4.
3. Read §1 through §3 of the handout on *Test-Driven Development*.

Assignments

1. Review the slides for Lecture 4.
2. Read the source code of `SimpleClass` in the source directory of the NetBeans project in `JUnit4Examples.zip`, and study the source code of `SimpleClassTest` in the test directory. You can ignore the two test cases for exceptions (`divisionByZero` and `divisionByZeroBetter`).

   Note that the code for the first two test cases (`divisionWithoutRemainder` and `divisionWithRemainder`) is very similar, and invites the introduction of an auxiliary method to factor out the code duplication.

   Do this refactoring, and then add some test cases for boundary situations, such as division by 1, and when dividend and divisor are equal (having expected result 1).
3. Study the source code for `TDDForCountDigitsMethod` in the source directory of `TDDExamples.zip`, and `TDDForCountDigitsMethodTest` in the test directory.

   Read §4 of the handout *Test-Driven Development*, and play along. That is, set constant `PHASE = 1`, and run the test cases. They should fail. Now rerun test cases, after incremening `PHASE` by 1. Repeat this until `PHASE = 4`.
4. **Submit in peach**[3] Do Exercise 1 of the handout *Test-Driven Development*. Use the given skeleton code. Submit your program and test cases in peach[3].
5. **Submit in peach**[3] Apply Test-Driven Development to implement the function `powerize(int)` with specification

```
/**
 * Write a number as a power with maximal exponent.
 *
 * @param n  the number to 'powerize'
```



```
 * @pre 2 <= n
 * @post n == \result.base ^ \result.exponent &&
 *     (\forall int b, int e;
 *      2 <= b && 1 <= e && n == b ^ e;
 *      e <= \result.exponent)
 */
public static Power powerize(int n)
```

where `Power` is a *record* class briefly defined by

```
public static class Power {
    public int base; // 0 <= base
    public int exponent; // 0 <= exponent
}
```

Use the given skeleton code. Of course, you also apply *functional decomposition*, and supply contracts and test cases for all functions. Submit your program and test cases in peach[3].

Hint In Java, an integer n is 32 bits and, hence, we have $n \leq 2^{31}$. For given integer exponent e with $1 \leq e \leq 31$, apply a *binary search* to find integer base b with $2 \leq b < 2^{31}$ such that $b^e < n \leq (b+1)^e$. The binary search can start with 1 as lower bound for b and n as upper bound, because we have $1^e < n \leq n^e$. Next, apply a *linear search* to find the largest e and accompanying b such that $n = b^e$.

Alternatively (more work in design and execution, but possibly easier to undestand), you can *factorize* n, to obtain $n = p_1^{e_1} \times p_2^{e_2} \times \cdots \times p_k^{e_k}$. Next, find the *greatest common divisor* d of all e_i. Then, we have $n = \left(p_1^{e_1/d} \times p_2^{e_2/d} \times \cdots \times p_k^{e_k/d}\right)^d$.

Deadline: Thursday 02 October 23:00 (Moscow time)

General Rules

1. In each submitted file, write:
   - full author name,
   - group number,
   - date of latest change.
2. Make sure you work neatly; pay attention to layout, spelling, and grammar. Adhere to the *Coding Standard for 2IP15*.
3. You are responsible for checking your work before submitting it.
4. Submit only your own work.

