תוכנה I בשפת ג'אווה
בית הספר למדעי המחשב
אוניברסיטת תל אביב

Exercise No. 2

Due Date: 27/11/2005

Read pages no. 1-21 in the Eclipse handouts (available on the course web-site) and perform the little experiments in the handouts. Then, solve the following exercises using Eclipse. At the end of this document you can find "*What to hand in?*" section.

Question 1:

1. Create a new java project, named *"ex2"*, with the default properties.
2. Under this project create a new package, named *"il.ac.tau.cs.<your user name>.ex2"*.
3. Under this package create a new class, named *"Exponentiation"*, and copy the following implementation:

```
public class Exponentiation {
   /**
   * Computes the exponential of a given integer.
   * @param base the integer to be raised to the power
   * of the given<code>exponent</code>.
   * @param exponent a positive integer exponent
   * @return the value of <code>base</code> raised to
   * the power of <code>exponent</code>.
   */
   static int exponent(int base, int exponent) {
        if (isEven(exponent)) {
             int tmp = exponent(base, exponent/2);
             return tmp * tmp;
        } else {
             return base * exponent(base, exponent-1);
        }
   }

   /**
   * Determine if the given integer is even
   * @param n the integer to be determined as even or odd
   * @return true if the integer is even.
   * Otherwise, returns false.
   */
   static boolean isEven(int n) {
        return (n % 2) == 0;
   }
}
```

4. Add a `main` method that prints to the standard output the value of 4^5.
5. Run the program. What is the result? Fix the `exponent` method and test it on other arguments by modifying the `main` method.

Question 2:

Prime factorization (also called prime *decomposition*) is the factorization of an integer into its constituent prime numbers. The prime factorization does not include 1, but does include every copy of every prime divisor. For example, the prime factorization of 36 is 2×2×3×3.

1. Under the *"ex2"* package create a new class, named *"PrimeFactorization"*, and copy the following implementation:

```
public class PrimeFactorization {
	/**
	 * Prints to the standard output a list all the prime factors
	 * (divisors) of a given integer.
	 *
	 * @param n the integer to find prime factors for
	 */
	static void printFactors(int n) {
		System.out.print("The prime factors of " + n + " are: ");
		printFactors(n, 2);
	}

	/**
	 * Prints to the standard output a list of all the prime
	 * factors of a given integer, <code>n</code>, that are not
	 * smaller than <code>minFactor</code>.
	 *
	 * @param n the integer to find prime factors for
	 * @param minFactor defines the smallest value of a prime
	 * factor of <code>n</code> that will be printed
	 */
	static void printFactors(int n, int minFactor) {
		if (minFactor > n) {
			return;
		}

		if (isDivisor(n, minFactor)) {
			System.out.print(" " + minFactor);
			printFactors(n / minFactor, minFactor);
		} else {
			printFactors(n, minFactor);
		}
	}

	/**
	 * Determines if the first given integer (<code>n</code>) is
	 * divisible by the second one (<code>divisor</code>).
	 *
	 * @param n the number to divide
	 * @param divisor the potential divisor
	 * @return true if <code>n</code> is <code>divisable</code>
	 * by <code>divisor</code>. Otherwise, returns false.
	 */
	static boolean isDivisor(int n, int divisor) {
		return (n % divisor) == 0;
	}
}
```

2. Add a `main` method that prints to the standard output a list of all the prime factors of 32. Run the program. What is the result? If required, fix the `printFactors` method.
3. Modify the main method so it will prints to the standard output a list of all the prime factors of 27. What is the result? If required, fix the `printFactors` method.

Question 3:

Fibonacci sequence is defined as a sequence of numbers (called *Fibonacci numbers*), where each number is the sum of the previous two. In general, the Fibonacci numbers are defined by the following *F* function:

$F(0) = 1$
$F(1) = 1$
$F(n) = F(n\text{-}1) + F(n\text{-}2)$ for all $n \geq 2$

The first Fibonacci numbers are: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89…

Under the "*ex2*" package create a new class, named "*Fibonacci*", with a method called `computeElement` that **iteratively** implements the above *F* function with only 2 temporary variables (and a loop counter, as explained in recitation 3). Write a `main` method that prints to the standard output the first 20 Fibonacci numbers to demonstrate your function.

Question 4:

The *factorial* operator is defined recursively as follows:

$$\begin{cases} 0! = 1 \\ n! = n \cdot (n-1)! \quad for\ all\ n > 0 \end{cases}$$

Under the "*ex2*" package create a new class, named "*Factorial*", with the following methods:

- `factIter` that implements the factorial operator iteratively
- `factRecursive` that implements the factorial operator recursively
- `main` method that prints to the standard output the first 20 elements in the following series: 0!, 1!, 2!, 3!,... `main` should compute the odd elements (1st, 3rd, 5th, …) using `factIter` and the even ones (2nd, 4th, …) using `factRecursive`.

What to hand in?

1. Softcopy: A jar file named ex2.jar should contain the classes "Exponentiation", "PrimeFactorization", "Fibonacci" and "Factorial", should be placed under ~/software1 as explained in the submission guidelines. Your programs should avoid "pretty printings" and "user friendly" messages. "Fibonacci" and "Factorial" classes should print their output using `System.out.println`, one element in each line. No commas (,) allowed.

2. Hardcopy: printouts of all the source code (.java files). Note that all the rhetorical questions in this exercise are meant to guide you through the development process. In this exercise we are interested only in the source code.