Programming 1

Tutorial 6

**Activity 1**

If we want to find the GCD of *a* and *b*, which is *gcd*(*a*, *b*). The Euclidean algorithm says:

*gcd*(*a*, 0) = a

*gcd*(*a*, *b*) = *gcd*(*b*, *a* mod *b*)

Where *a* mod *b* is the same as a % b in Java. Write a program to find the GCD of two positive integers entered by the user. Make use of a while or do…while loop.

**Expected result:**

Enter A (A > 0): 26

Enter B (B > 0): 13

GCD(26,13) = 13

**Instructions**

Based on the Euclidean algorithm, when b becomes zero, a is the GCD. Until then, we modify a and b according to the second equation:

*gcd*(*a*, *b*) = *gcd*(*b*, *a* mod *b*)

From this equation, we can see that a will take b's previous value and b will take a % b. In order to do that in a program, we can't do something like:

a = b; // a and b have the same value now

b = a % b; // b is zero since a and b are equal

To achieve the desired effect, we can use another variable, for instance, c.

c = a; // make a backup of a

a = b; // ok, now a and b are equal

b = c % b; // we can use c because c holds the previous value of a

This technique is called swapping two numbers. It is possible without the extra variable c but that method would be more difficult to understand:

a = a + b;

b = a - b;

a = a - b; // done swapping a and b

b = b % a; // now, b % a is the previous a % b

# Activity 2

Factoring of integers. Write a program that asks the user for an integer and then prints out all its factors. For example, when the user enters 150, the program should print:

2 3 5 5

(2 \* 3 \* 5 \* 5 = 150)

Use a class FactorGenerator with a constructor FactorGenerator(int numberToFactor) and methods nextFactor and hasMoreFactors. Supply a class FactorPrinter whose main method reads a user input, constructs a FactorGenerator object, and prints the factors.

# Activity 3

Extend the BankAccount example from the lecture.

* Add a method to add annual interest to the account’s balance.
* A bank account should have the name of the account holder. Implement this feature.
* Add a method called toString() which returns a string of this format:

(if the balance is negative, put the – sign before the dollar sign)

Benson, $117.25

Mathew, -$17.50

* Implement a method called transfer to send money from one bank account to another.

There is a $0.5 fee per transfer. Also, if the account does not have enough money to transfer and pay the fee, print out suitable error messages.

# Activity 4

Implement a class Car with the following properties. A car has a certain fuel efficiency (measured in miles/gallon or kilometers/liter—*pick one*) and a certain amount of fuel in the gas tank. The efficiency is specified in the constructor, and the initial fuel level is 0. Supply a method drive that simulates driving the car for a certain distance, reducing the amount of gasoline in the fuel tank. Also supply methods getGasInTank, returning the current amount of gasoline in the fuel tank, and addGas, to add gasoline to the fuel tank.

Sample usage:

Car myHybrid = new Car(50); // efficiency of 50 miles per gallon

myHybrid.addGas(20); // Tank 20 gallons

myHybrid.drive(100); // Drive 100 miles

double gasLeft = myHybrid.getGasInTank(); // Get gas remaining in tank

You may assume that the drive method is never called with a distance that consumes more than the available gas. Create a CarDemo class with a *main* method that uses this Car class and show all of its features.

# Activity 5 \*\*

Create a type named List which behaves like a dynamic array of integers. This class should store the numbers in an array-typed attribute and provide methods to add, get and remove elements.

**Goal:** Implement the list to have all the features of array, but with dynamic length.

\* This is a do-along-with-teacher activity.