COS101

Practical 4

8-12 October 2018

Question 1

Write a superclass *Worker* and subclasses *HourlyWorker* and *SalariedWorker*. Every worker has a name and a salary rate. Write a method computePay (int hours) that computes the weekly pay for every worker. An hourly worker gets paid the hourly wage for the actual number of hours worked, if 'hours' is at most 40. If the hourly worker worked more than 40 hours, the excess is paid at time and a half. The salaried worker gets paid the hourly wage for 40 hours, no matter what the actual number of hours is. Code a java program that uses polymorphism to test these classes.

Question 2

Write a class OurDate that models the date as a set of three integers, one for the day of the month (variable dayOfMonth), one for the month (month) and one for the year (year). The three integers also serve as instance variables for the class. Include a constructor OurDate(int year, int month, int dayOfMonth) and a suitable set of accessor and mutator methods for the class.

The number of days between two dates is important when calculating interest on an investment or savings account. Include a method, numberOfDaysBetween, that returns the number of days elapsed between two dates, date1 and otherDate in the method invocation, date1.numberOfDaysBetween(otherDate). **Hint**: Use a one dimensional array to store the number of days that occur in each month, daysInMonth[] = {31, 28, 31,30, 31,…,31}. Your method should also cater for leap years. For the years 1990 to 2012 a leap year occurs when the value for 'year' is a multiple of 4. Since our dates will not exceed these you may assume that a leap year occurs when 'year' is a multiple of 4. In this case the number of days for February should increase from 28 to 29. Test your method by comparing your output with that in the attached spreadsheet.

In this section you are required to develop a **BankAccount** with instances and methods as indicated in the UML class diagram. The variable dateOpened refers to the actual date when the account was opened at the bank. Methods deposit, withdraw and transfer change the balance of a BankAccount. A deposit will increase the balance while withdraw will reduce the balance. Also, an amount can only be withdrawn if there are sufficient funds in the BankAccount. The transfer method transfers an amount from the BankAccount under consideration to the account referred to as an argument in the method. Define appropriate constructors for the BankAccount class.

In the case of the **SavingsAccount** interest, based on a fixed interest rate, is paid on the daily balance of the account. To ensure that interest is calculated accurately, interest is added each time when a withdrawal or deposit is made. Interest is simply calculated by multiplying the balance of the account $\times$ the number of days since the last interest calculation (as reflected by instance variable dateLastInterestAdded) $\times$ the interest rate per day (i.e. interest per annum / 365 (or 366)). The interest is subsequently added to the balance of the account and dateLastInterestAdded is also updated. The method addInterest(Date) calculates the interest where 'Date' refers to the date when the transaction is performed. The period for the interest covers the number of days since the date when interest was last added to the account and the date when the interest calculation is performed. Although the interest calculated is immediately added to the balance of the account, a separate instance variable, interestAccrued, accumulates the total interest credited to the account.

The **NoticeDepositAccount** is similar to a SavingsAccount except that the accountholder agrees to leave the money in the account for a predetermined period, e.g. 60, 90 or 120 days. There is a penalty of R250 for early withdrawal, i.e. any withdrawal before the predetermined period has expired. The penalty is debited to the account. This approach should be implemented in the NoticeDepositAccount.

UML class diagram