Hacettepe University
Department Of Computer Engineering
Bil236 Programming Laboratory
Experiment 1

Subject	Object Oriented Programming – Inheritance / Polymorphism
Submission Date	20.2.2012
Due Date	07.3.2012
Programming Environment	Java / Eclipse
Advisors	Dr**.** Ayça Tarhan / R.A. Tuğba GÜRGEN

INTRODUCTION

Object-oriented programming (OOP) is a programming paradigm using "objects" – data structures consisting of data fields and methods together with their interactions – to design applications and computer programs. Programming techniques may include features such as data abstraction, encapsulation, messaging, modularity, polymorphism, and inheritance [1].

In this homework you are expected to write a banking system program which targets and includes the usage of OOP paradigms such as inheritance, encapsulation and polymorphism.

SOME CONCEPTS AND BACKGROUND FOR OOP

Inheritance: Inheritance allows the programmer to treat derived class members just like their parent class's members. This type of relationship is called child-Parent or is-a relationship.
"Subclasses" are more specialized versions of a class, which *inherit* attributes and behaviors from their parent classes, and can introduce their own [1].

Polymorphism: Polymorphism allows objects to be represented in multiple forms. Even though classes are derived or inherited from the same parent class, each derived class will have its own behavior. Polymorphism is a concept linked to inheritance and assures that derived classes have the same functions even though each derived class performs different operations [2].

Abstraction: According to Richard Gabriel, abstraction is "the process of identifying common patterns that have systematic variations; an abstraction represents the common pattern and provides a means for specifying which variation to use". An abstract class is a parent class that allows inheritance but can never be instantiated. Abstract classes contain one or more abstract methods that do not have implementation. Abstract classes allow specialization of inherited classes [2].

Interfaces: An *interface* looks like a class, but has no implementation. The only thing it contains are definitions of *events*, *indexers*, *methods* and/or *properties*. The reason *interfaces* only provide definitions is because they are inherited by *classes* and *structs*, which must provide an implementation for each interface member defined [3].

EXPERIMENT

In this experiment, you are going to implement a banking system which simulates traditional banking operations carried out in real world. In this system, there will be only one bank having customers and customer accounts. You are expected to provide different types of bank accounts for customers to choose from and implement basic transactions among these accounts in the system.

In the banking system, you are supposed to satisfy the following requirements;

- Each customer possesses at least one bank account and can close one or all of his accounts.
- A bank account should be one of the following two types; current account or savings account.
- Accounts should hold one of three types of currencies which are Euro, Dollar and Turkish Lira. Money transfer is allowed only between the accounts having the same currency type.
- Customers having savings account could be able to choose one of the following interest rates; daily, monthly or yearly interest rates. Each savings account having one of the previous interest rates should derive from one savings account.
- There will be following transactions provided in proposed system; withdraw, deposit and transfer. There will be one ITransactions interface having the following methods; withdraw, transfer and deposit. Main bank account class must implement this interface.
- Each customer could be able to list current balance of one of his accounts or all of his accounts.
- Transaction history of one account or all accounts of a customer could be listed in the proposed banking system.
- Each customer has unique id, name and surname. Each bank account also has unique id, bank account type, currency type, account opening date and interest start date.
- For the interest calculations, one month should be considered as 30 days.
- Interest is calculated in a cyclic fashion. Interest of the last period is added to the current balance after the interest period is up. Interest is calculated for some amount of money only if it appears in the bank account for the entire interest period. As an example, let's consider a bank account with monthly interest rate of % 1:
    - Account information
        - Account opening time: 02.01.2012
        - Account balance: 100 TL
        - Interest rate: %1
    - Customer deposits 80 TL on 05.02.2012
        - Account balance before deposit: 100 (previous balance) + 1 (interest of 100) = 101
        - Account balance after deposit: 101 (current balance) + 80 (deposit) = 181
    - Customer checks his current balance on 04.03.2012
        - Account balance: 181 + 1 (interest of 101) = 182.01
    - Customer withdraws 20 TL on 05.04.2012
        - Account balance before withdrawal: 182.01 + 1.82 (interest of 182) = 183.83
        - Account balance after withdrawal: 183.83 - 20 (withdrawal) = 163.83

COMMANDS

- Below all the available commands that the system should support are listed:

  ADD CUSTOMER: Using this command new customer is added to the system. Adding a customer having the same id with an existing customer is not allowed and no action should be taken when such a command is encountered.

  ADD ACCOUNT: This command adds an account to an existing customer. Adding an account having the same id with an existing account and adding an account to non-existing customer is not allowed. No action should be taken when such a command is encountered.

  CLOSE ACCOUNT: Using this command an existing account is closed in the system. No action should be taken if non-existing account is attempted to close.

  DEPOSIT: Deposit operation to an existing account is carried out using this command.

  WITHDRAW: Cash withdrawal operation from an existing account is carried out using this command. If there are insufficient funds in the account to be withdrawn, only the existing funds should be withdrawn by the customer.

  TRANSFER: This command denotes money transfer between two different bank accounts. If there are insufficient funds in the source bank account to be transferred, then only the existing funds should be transferred to the target bank account.

  SHOW BALANCE: This command shows the current balance of an existing bank account. It is not considered as a transaction.

LIST TRANSACTIONS: Using this command all transactions of an account or all accounts of a customer is listed till a given date. Even if the account is closed, all the transaction till the given date is listed.

- Only outputs of SHOW BALANCE and LIST TRANSACTIONS commands are written to the output text file.

Sample Input File

```
ADD CUSTOMER 1 Ahmet Aydın
ADD ACCOUNT 1 CUSTOMER 1 CURRENT 07.02.2012 100 EURO
ADD ACCOUNT 2 CUSTOMER 1 SAVINGS 02.01.2012 100 TL MONTHLY 1
ADD ACCOUNT 3 CUSTOMER 1 SAVINGS 07.02.2012 50 DOLLAR DAILY 3
ADD CUSTOMER 2 Fatma Aydın
ADD ACCOUNT 4 CUSTOMER 2 CURRENT 03.02.2012 100 TL
ADD ACCOUNT 5 CUSTOMER 2 SAVINGS 03.02.2012 100 TL YEARLY 5
CLOSE ACCOUNT CUSTOMER 1 ACCOUNT 1 07.02.2012
DEPOSIT CUSTOMER 1 ACCOUNT 2 80 TL 05.02.2012
SHOW BALANCE CUSTOMER 1 ACCOUNT 2 04.03.2012
WITHDRAW CUSTOMER 1 ACCOUNT 2 2O TL 05.04.2012
SHOW BALANCE CUSTOMER 1 ACCOUNT 2 05.04.2012
SHOW BALANCE CUSTOMER 2 ALL 03.02.2014
LIST TRANSACTIONS CUSTOMER 1 ALL 07.04.2012
```

Sample Output File

```
SHOW BALANCE CUSTOMER 1 ACCOUNT 2 04.03.2012
OUTPUT:
CUSTOMER 1 AHMET AYDIN
ACCOUNT 2 SAVINGS TL MONTHLY 1
BALANCE AT 04.03.2012: 182.01 TL

SHOW BALANCE CUSTOMER 1 ACCOUNT 2 05.04.2012
OUTPUT:
CUSTOMER 1 AHMET AYDIN
ACCOUNT 2 SAVINGS TL MONTHLY 1
BALANCE AT 05.04.2012: 163.83 TL

SHOW BALANCE CUSTOMER 2 ALL 03.02.2014
OUTPUT:
CUSTOMER 2 Fatma Aydın
ACCOUNT 4 CURRENT TL
BALANCE AT 03.02.2014: 1O0.00 TL

ACCOUNT 5 SAVINGS TL YEARLY 5
BALANCE AT 03.02.2014: 110.25 TL

LIST TRANSACTIONS CUSTOMER 1 ALL 07.04.2012
OUTPUT:
CUSTOMER 1 AHMET AYDIN
ACCOUNT 1 CURRENT EURO:
CLOSED

ACCOUNT 2 SAVINGS TL MONTHLY 1
DEPOSIT 80.00 TL 05.02.2012
WITHDRAW 20.00 TL 05.04.2012

ACCOUNT 3 SAVINGS DOLLAR DAILY 3
```

NO TRANSACTION

GENERAL RULES & LIMITATIONS

- Your program should use OOP paradigm, so you must use “classes”, “inheritance” and “polymorphism”. Using structures are prohibited.
- Attributes must be accessed through get/set properties! (Not directly).
- In order to run, program expects two parameters. First parameter is the input file containing the commands and the second parameter is the output file that the outputs of the commands are written into. Format of the commands in the input file and the outputs of these commands are well defined in homework definition of this document. Students whose outputs do not match with the given output format will get no points from the homework.
- “commands.txt” file will contain only well-formed commands. Therefore you should feel safe that you will not face invalid commands and related arguments. However, invalid/crashing homeworks will be graded only with the report section and code section will get 0 points.
- You can add your novel methods in methods section of banking system as your own design and approach! This is left to you and all your design considerations must be given in report. Class diagrams should be included also.

NOTES

- SAVE all your work until the experiment is graded.
- The assignment must be original, INDIVIDUAL work. Downloaded or modified source codes will be considered as cheating. Also the students who share their works will be punished in the same way.
- You can ask your question via course’s news group: news://news.cs.hacettepe.edu.tr/dersler.bil236
- Please respect the office hours of your advisor. Office hours are:
  - 12:30 – 13:30, Thursday
  - 09:30 – 10:30, Friday
  - 12:30 – 13:30, Monday

SUBMISSONS

- Your submission will be in the format below

```
<StudentID>
        |-- report
                |-- report.pdf

        |-- source

                |--All Eclipse project files in one zip package (name it as source.zip)
```

- You have to use “Online Experiment Submission System”. http://submit.cs.hacettepe.edu.tr. Other type of submissions especially by e-mail WILL NOT BE ACCEPTED.
- Submission deadline is 07.3.2012, 17.05 pm. No further extension will be given!!!

REFERENCES

1. http://en.wikipedia.org/wiki/Object-oriented_programming
2. http://www.csharpcorner.com/uploadfile/eecabral/oopsand.net211102005075520am/oopsand.net2.aspx
3. http://www.csharp-station.com/Tutorials/Lesson13.aspx