Object Oriented Programming 1
Fall 2015-16
Lab Manual: 05

Lab Task:

1. Lab Review, and start with unfinished classes from Lab_04.
2. Develop Java classes.

Note: Student must follow the name of class, member variables, and functions.
And students should use fully qualified names for these, as well camel notions.
And the syntax alignment has to be as it should be.

1. Lab_04 Classes – Review:

2. Develop Java classes

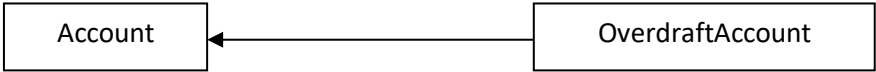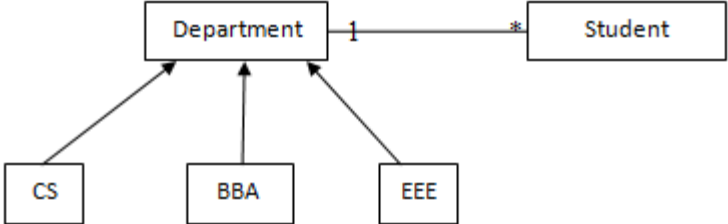| |
|---|
| Now we want to reuse our Account class. The scenario is every student object has an account.
 Tasks that you have to do:
 1. If a student deposit a book after certain duration then for each day 10 taka will be charged by the library authority.
 2. Student can pay that amount (charged amount) from his/her bank account to the library account. So the library also has an account.
 3. Show the total amount a student paid to the library.
 4. Show the total amount library gets from the students (total charged amount).
 5. But if any student has any valid reason for delay then the librarian can excuse the charged amount. Librarian can excuse full or partially (%). |
| In this program we want to reuse our last developed Library and Student Class. Our objective is to develop the scenario, where student can take maximum 5 books (for 5 days each) at a time from the library, and return these books.
 Tasks that you have to do:
 1. Show all the borrowed book information of a student from student class
 2. Show all the student name and book information from library class, which are currently borrowed by students.
 3. Step 2
 4. Preserve borrowed history of student object, which he/she takes in his/her lifetime.
 5. Preserve borrowed history of book objects that are taken by students. |

Inheritance	
Reuse Account class, and OverdraftAccount class extends Account class. OverdraftAccount as member	 **OverdraftAccount** Member Variable: int overdraftLimit Member Function: boolean withdraw(int amount) //overwrite this method

- Implement the above inheritance scenario.
- Test base and derived class's constructor calling.
- Test the type of access modifier's impact on inheritance.

| Test Run time polymorphism example |

Department
Member variable:
    • String deptName;
    • int creditFee
MemberFunction:
    • 2/3 constructor
    • Abstract method void calculateSemesterFee(int credit)

CS, BBA, EEE
Member Variable:
Member Function: overwrite calculateSemesterFee(int credit) in each class
Student:
Member Variable:
    • String stuName
    • Department dept; |

| | MemberFunction:
• 2 constructor
• void showStudentInfo()
• void setDepartment(Department dept)
• void changeDepartment(Department dept)
• void semesterFee(int totalCredit){
    dept. calculateSemesterFee(totalCredit);
} |