

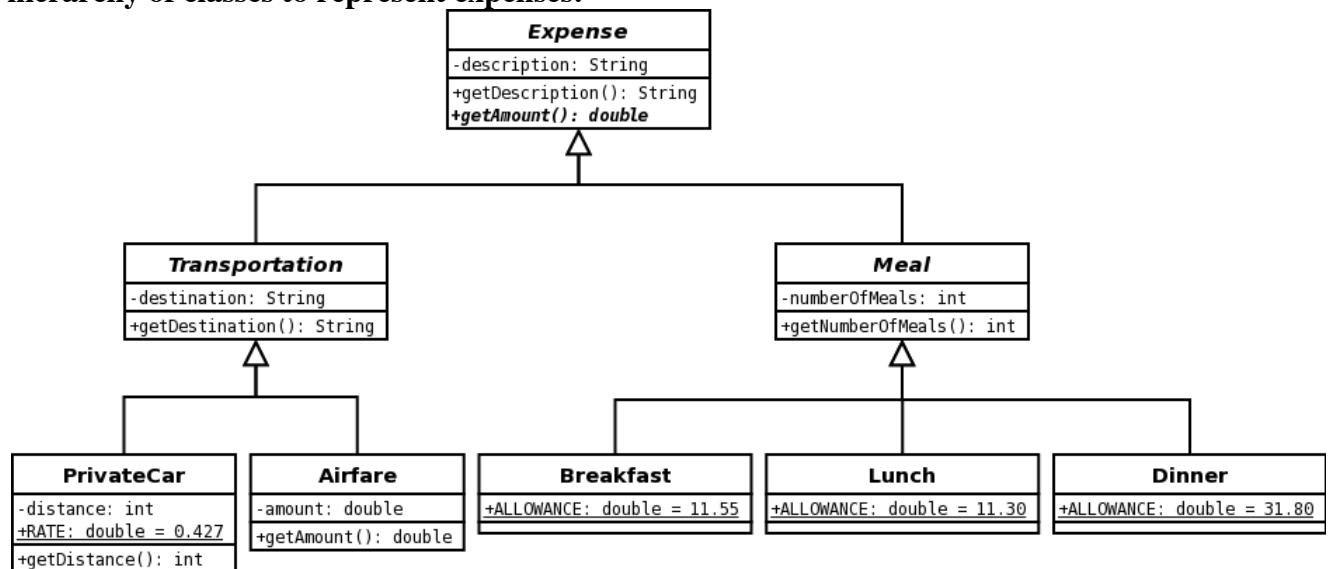
ITI1121: Assignment #1

Due date\time : June4th @23:00 - Submitted via BBL

Part I: Written answer questions (20) (submit a .doc file)

Answer the following questions using no more than the available space:

1. The company Java Financial Solutions is developing a new software system for tracking professional expenses. You are part of the software development team responsible for the hierarchy of classes to represent expenses.



Specifications:

- All expenses have a description (a character string);
- All the transportation expenses have a destination (a character string);
- A transportation expense using a private car has a distance (of type int);
- A transportation expense by air has a fixed amount (of type double) specified when a new transportation expense is created;
- All the meal expenses have an attribute which represents the number of meals.
- All the expenses have a method to calculate the amount represented by this expense:
 - The amount for a transportation expense using a private car is a fixed rate times the distance traveled;
 - The amount for a transportation expense by air is a fixed amount (specified when a new transportation expense is created);
 - The amount for a meal expense is the number of meals times a fixed rate. The rate depends on the kind of meal: Breakfast, Lunch or Dinner;

You must write the Java implementation of the following classes. Make sure to include the constructors, the access methods and where appropriate the method for calculating the amount represented by the expense.

A. Expense 3 marks

```
public abstract class Expense {  
    private String description;  
  
    public Expense(String expDescript) {  
        description = expDescript;  
    }  
    public abstract double getAmount();  
}
```

B. Transportation 2 marks

```
public abstract class Transportation extends Expense {  
    private String destination;  
    public Transportation(String transDescript, String travelDest) {  
        super(transDescript);  
        destination = travelDest;  
    }  
    public String getDesination() { return destination; }  
}
```

2 marks

C. PrivateCar

```
public class PrivateCar extends Transportation {  
    public static final double RATE = 0.427;  
    private int distance;  
  
    public PrivateCar(String transDescript, String travelDestint, int travelDist)  
    {  
        super(transDescript, travelDestint);  
        distance = travelDist;  
    }  
  
    public int getDistance() { return distance; }  
  
    public double getAmount() { return distance*RATE; }  
}
```

2 marks

D. Airfare

```
public class Airfare extends Transportation  
{  
    private double amount;  
  
    public Airfare(String transDescript, String travelDestint, double travelAmount)  
    {  
        super(transDescript, travelDestint);  
        amount = travelAmount;  
    }  
    public double getAmount() { return amount; }  
}
```

E. Meal 2 marks

```
public abstract class Meal extends Expense
{
    private int numberOfMeals;

    public Meal(String transDescript, int numMeals) {
        super(transDescript);
        numberOfMeals = numMeals;
    }

    public int getNumberOfMeals() { return numberOfMeals; }
}
```

F. Breakfast 2 marks

```
public class Breakfast extends Meal
{
    public static final double ALLOWANCE = 11.55;

    public Breakfast(String transDescript, int numMeals) {
        super(transDescript, numMeals);
    }

    public double getAmount() { return getNumberOfMeals() * ALLOWANCE; }
}
```

G. Complete the partial implementation of the class ExpenseTracker below. An ExpenseTracker is used to store Expenses. i) Add the type of the elements of the array expenses. ii) Complete the constructor. iii) Complete the implementation of the method double getTotal(). The method double getTotal() returns the total amount for all the expenses that are currently stored in the ExpenseTracker.

2 mark

```
public class ExpenseTracker {  
    private Expense [] expenses;  
    private int size; // keeps track of the number of elements  
    public ExpenseTracker( int capacity ) {  
        expenses = new Expense[capacity];  
        size = 0;  
    }  
}
```

// a method has been defined for adding expenses to the tracker

```
public boolean add( Expense e ) { ... }
```

```
    public double getTotal() {
```

2 marks

```
        double sum=0;  
        for(int i=0;i<size;i++) sum+=expenses[i].getAmmount();  
        return sum;
```

```
    }  
}
```

Below is a test program to help you understand the work that needs to be done for this question.

```
public class Run {  
    public static void main( String[] args ) {  
        ExpenseTracker epro = new ExpenseTracker( 10 );  
        epro.add( new PrivateCar( "ACFAS 2004", "Montreal (QC)", 430 ) );  
        epro.add( new Airfare( "IWBRA 2005", "Atlanta (GA)", 204.0 ) );  
        epro.add( new Breakfast( "IWBRA 2005", 2 ) );  
        epro.add( new Lunch( "IWBRA 2005", 3 ) );  
        epro.add( new Dinner( "IWBRA 2005", 2 ) );  
        System.out.println( "total = " + epro.getTotal() );  
    }  
}
```

Its output is as follows: **total = 508.21000000000004**

2. Consider the following declarations:

3 marks

```
package pack1;  
public class Class1 {  
    private int v1;  
    protected int v2;  
    int v3;  
    public int v4;  
}  
package pack1;  
    public class Class2 {...}  
    package pack2;  
    public class Class3 extends pack1.Class1 {...}  
    package pack2;  
    public class Class4 {...}
```

- a. **What visibility must variables declared in pack1.Class1 have in order to be visible in pack1.Class2?**

public, protected, or package

- b. **What visibility must variables declared in pack1.Class1 have in order to be visible in pack2.Class3?**

public or protected

- c. **What visibility must variables declared in pack1.Class1 have in order to be visible in pack2.Class4?**

public

Part II: Programming Module (80 marks):

Write a banking program that simulates the operation of your local bank. You should declare the following collection of classes.

Class Account

Data fields: customer (type Customer), balance, accountNumber, transactions array (type Transaction[]). Allocate an initial Transaction array of a reasonable size (e.g., 20), and provide a reallocate method that doubles the size of the Transaction array when it becomes full.

Methods: getBalance, getCustomer, toString, setCustomer

Class SavingsAccount extends Account

Methods: deposit, withdraw, addInterest

Class CheckingAccount extends Account

Methods: deposit, withdraw, addInterest

Class Customer

Data fields: name, address, age, telephoneNumber, customerNumber

Methods: Accessors and modifiers for data fields plus the additional abstract methods getSavingsInterest, getCheckInterest, and getCheckCharge.

Classes Senior, Adult, Student, all these classes extend Customer

Each has constant data fields SAVINGS_INTEREST, CHECK_INTEREST, CHECK_CHARGE, good! and OVERDRAFT_PENALTY that define these values for customers of that type, and each class implements the corresponding accessors.

Class Bank

Data field: accounts array (type Account[]). Allocate an array of a reasonable size (e.g., 100), and provide a reallocate method.

Methods: addAccount, makeDeposit, makeWithdrawal, getAccount

Class Transaction

Data fields: customerNumber, transactionType, amount, date, and fees (a string describing unusual fees)

Methods: processTran

You need to write all these classes and an application class that interacts with the user. In the application, you should first open several accounts and then enter several transactions.

- **Note: Please download and use the classes : BankGUI, BankApp**
 - As well as the interface for the class Bank
- **For each class, add comments (documentation) describing the variables and methods used.**
- **The following are the values for the constants in the classes: Senior, Adult and Student:**

| | SAVINGS_INTEREST | CHECK_INTEREST | CHECK_CHARGE | OVERDRAFT_PENALTY |
|---------|------------------|----------------|--------------|-------------------|
| Adult | 0.03 (3%) | 0.01 | 3 cents | \$25 |
| Student | 0.04 | 0.01 | 2 cents | \$25 |
| Senior | 0.04 | 0.01 | 1 cent | \$10 |