ITI1121: Assignment #1

Can be done individually or by a team of two.

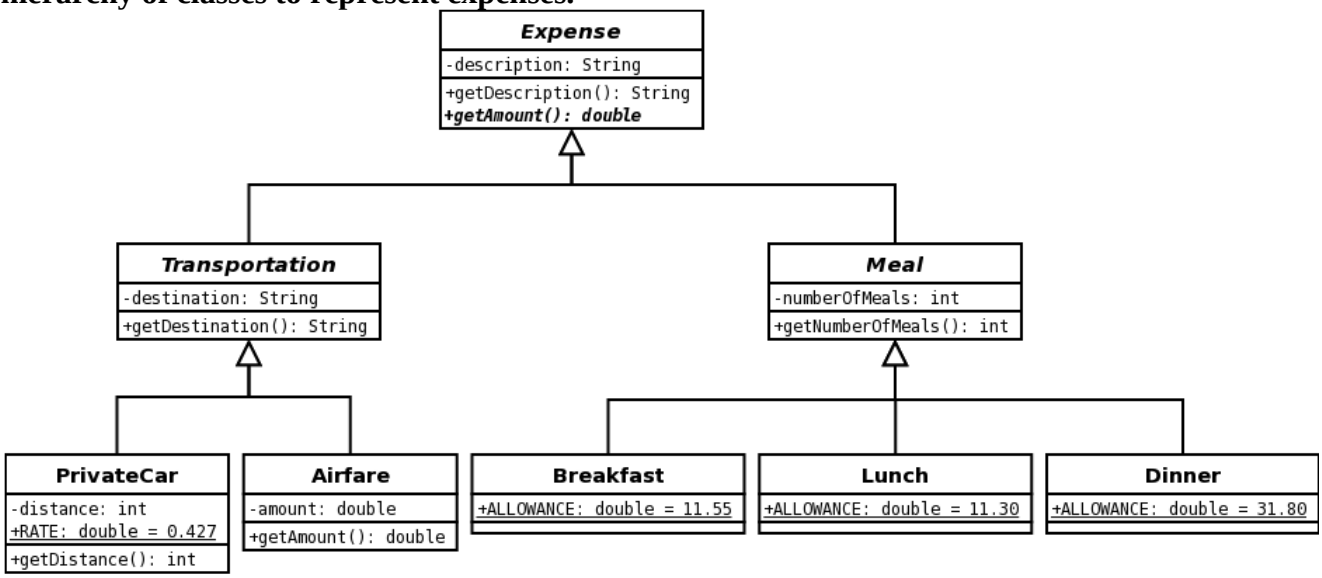
Each file you submit must have your name and student number

Please note any copied component of the solution will result in a zero mark for the entire assignment

Part I: Written answer questions (20)

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

B. Transportation

C. PrivateCar

D. Airfare

E. Meal

F. Breakfast

G. Complete the partial implementation of the class ExpenseTracker below. An Expense-Tracker 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.

```
public class ExpenseTracker {
        private _____  [] expenses;
        private int size; // keeps track of the number of elements
         public ExpenseTracker( int capacity ) {
                _____________________________________;
                size = 0;
         }
// a method has been defined for adding expenses to the tracker
public boolean add( Expense e ) { ... }
  public double getTotal() {
```

```
  }
}
```

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:

```
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?**

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

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

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:

1) An abstract class **Customer**: each customer has**: firstName(String), lastName (String), age** (integer), **customerNumber** (integer)
   - The Class customer has a class variable **lastCustomerNumber** that is initialized to 9999. It is used to initialize the **customerNumber.** Hence, the first customer number is set to 9999 and each time a new customer is created variable **lastCustomerNumber** is increased by one.
   - Define the accessors and modifiers for the data fields as well as the methods **equals** and **toString**. Define as many constructors as needed.
   - Add the following abstract methods: **getSavingsInterest**, **getCheckInterest** and **getCheckCharge** that all return a double. These methods will be implemented in the children classes.
2) The classes **Senior**, **Adult**, **Student** extend the **Customer** class, they all have different values for the constants that are shown in the table below.
   -The **Senior** class has an additional Boolean field that indicates whether the **Senior** is a **VIP** or not.

   The following are the values for the constants in the classes: Senior, Adult and Student:

	SAVINGS_INTEREST	CHECK_INTEREST	CHECK_CHARGE	OVERDRAFT_PENALTY (only up to $500)
Adult	0.03 (3%)	0.01	3 cents	$25
Student	0.07	0.03	2 cents	cannot
Senior	0.08	0.04	1 cent	$10
Senior VIP	0.1	0.04	0	No over draft up to $100 $5 up to $500

3) Class **Transaction**: has a transaction **type** (byte) (0= deposit, 1= withdraw, 2=addedInterest), ***amount*** (double) stores the amount in the transaction, **date** of the transaction (of type Date), **fees**, and a **description** (a string). It has one method: **processTransaction** that returns a string with the transaction information (similar to toString).

4) Class **Account:** An abstract class, each account has a **customer** (type Customer), **balance** (double), **accountNumber** (generated in a similar way to the customer number), **transactions** (an array of Transaction[]).
   - When the Account is initialized you need to create an array with INITSIZE (a class variable set to 25).
   - The account has a method **reallocate** that doubles the size of the transactions array when it is full.

- Other methods; **getBalance (returns double)**, **getCustomer (returns a customer)**, **toString** (returns a string with the account information except for the transactions), **setCustomer** (Customer c).
-

5) Classes **SavingsAccount** and **CheckingAccount** extend **Account.**
They implement the methods: **deposit**, **withdraw** and **addInterest**.
The method deposit adds a positive amount to the account balance and store the transaction information in the transactions array.
The method **withdraw** reduces the balance only if sufficient balance is available. Adults and Seniors can overdraft money up to $500 at an overdraft cost (see the above table). The withdraw information should be stored in a transaction along with the overdraft fee. If the customer withdraws money from a checking account and additional charge is incurred and added to the transaction fee.
The method **addInterest** calculates the Interest for the balance and adds it to balance (for example, if balance =$10 for a checking account of a senior, **addInetrest** calculate the Interest (10*0.04) and adds it to the balance and saves the information in a transaction.

6) The class **Bank** has an accounts array (account []). When a bank is created, allocated a space 100 accounts and provide a reallocate method that can double the size of the accounts array whenever you cannot add a new account. The Bank class has the following methods: addAccount, makeDeposit, makeWithdrawal and geAccount.
7) Some Notes:
   - Please download and use the classes: BankGUI, BankApp, as well as the interface for the class Bank. You can modify the code as needed.
   - For each class, add comments (documentation) describing the variables and methods used.

Create a folder for each question.
For Q1) submit a word document.
For Q2) follow the same steps you follow in your labs.

Create and submit a zip file with the name: ST1#_ST#2_Assign1.