COMP1112 Object Oriented Programming
Spring 2023, Project #1,
Due: 21 April, 23:50 via Blackboard

IŞIK UNIVERSITY
COMPUTER SCIENCE AND ENGINEERING

In this project, you will design and implement a simple version of a **`Communication Management and Payment System`**. In the system we have the following objects:

`Subscriber` – Subscribers can communicate each other. There are two ways of communication: **voice** or **message** communication. Subscribers can also get connected to the **Internet** via their mobile phones.

`ServiceProvider` – ServiceProviders charge Subscribers based on their communication usage. Each communication type has a specific cost (voice communication cost, messaging cost, Internet cost).

`Invoice` – It is paid by the Subscriber. ServiceProviders record the communication actions of Subscribers and create Invoices based on their communication usage.

Details of these classes are described below. You must implement the following fields and methods given below. However, you can also add additional methods and fields if needed.

You will implement **four classes** in this project.

`Subscriber` `class`:

`Subscriber` class must have the following data fields described below:
- `s_id`: int (s_id must be unique and between 1000000 - 9000000)
- `name`: String
- `age`: int
- `isActive`: Boolean (a subscriber is automatically active)
- `s_provider`: ServiceProvider
- `invoice`: Invoice (created whenever a subscriber is created)

`Subscriber` class must have the following methods with the given parameters described below:
- A `constructor` with five parameters: String `name`, int `age`, ServiceProvider `s_provider`, double `usageLimit`. The other data fields must be initialized based on the descriptions.
- `void updateStatus()` for this subscriber to check if the subscriber is active or not. If the subscriber does not pay the invoice before the invoice's lastDayToPay, the subscriber becomes inactive. This method updates isActive field accordingly.
- `void makeVoiceCall(int minute, Subscriber receiver)` for this subscriber to call receiver subscriber via the ServiceProvider. Parameter receiver is the another Subscriber that will receive the voice call. Inactive subscribers cannot make voice calls; thus before call operations, the status of the subscriber must be checked.
- `void sendMessage(int quantity, Subscriber receiver)` for this subscriber to send message to the receiver. Parameter quantity is the number of messages to be sent. receiver is the other subscriber that will receive the messages. Inactive subscribers cannot send messages; thus before the operation, the status of the subscriber must be checked.
- `void connectToInternet(double amount)` for this subscriber to get connected to the Internet. Parameter amount is the number of data in MB that is transferred when this subscriber gets connected to the Internet. Inactive subscribers cannot connect to the Internet; thus before the operation, the status of the subscriber must be checked.
- void changeServiceProvider(ServiceProviders s_provider) for this subscriber to change its ServiceProvider. Before changing it, the invoice must be paid.
- `getter` and `setter` methods for required data fields.

`ServiceProvider` `class`:

`ServiceProvider` class must have the following data fields described below:
- `p_id`: int (must be unique and between 500 and 600)
- `p_name`: String
- `voiceCallCost`: double
- `messagingCost`: double
- `internetCost`: double
- `discountRatio`: int
- `subscribersList`: a list of all subscribers (array or ArrayList)

`ServiceProvider` class must have the following methods with the given parameters described below:

- A `constructor` that takes a separate parameter to set each data field.
- `double calculateVoiceCallCost(int minute, Subscriber caller)` to calculate the total amount to pay for a voice call that is initiated by the caller.
- `double calculateMessagingCost(int quantity, Subscriber sender, Subscriber receiver)` to calculate the total amount to pay for sending messages from sender subscriber to receiver subscriber.
- `double calculateInternetCost(double amount)` to calculate the total amount to pay for Internet usage.
- `boolean addSubscriber(Subscriber s)` to add a new subscriber in the list of subscribers
- `boolean removeSubscriber(Subscriber s)` to remove an existing subscriber from the list of subscribers
- `getter` and `setter` methods for the required data fields.

`Invoice` class:

`Invoice` class must have the following fields:

- `usageLimit`: double
- `currentSpending`: double
- `lastDayToPay`: Date

`Invoice` class must have the following methods with the given parameters described below:

- A `constructor` with parameters `usageLimit` and `lastDayToPay`. When an invoice is created, the `currentSpending` must be initialized as zero and `lastDayToPay` must be assigned 30 days after the invoice is created.
- `boolean isLimitExceeded(double amount)` is to check whether the `usageLimit` is exceeded or not.
- `void addCost(double amount)` is to add spending to the Invoice.
- `void pay(double amount)` is to pay the Invoice with the given amount. If amount is equal to the current spending, reset lastDayToPay accordingly.
- `void changeUsageLimit(double amount)` this method is for changing the limiting Amount.
- `getter` methods for the required data fields.

`SimulateSystem` class:

This class will be used to execute the communication system. You will implement a `main` method in this class. In the main method, you will implement

- An array (or ArrayList) of `Subscribers`
- An array (or ArrayList) of `ServiceProviders`

You will print a menu to let the user of the program do the following actions:

1- Creating a new Service Provider
2- Create a new Subscriber
3- Voice Call: A subscriber calls another subscriber
4- Messaging: A subscriber sends a message to another subscriber
5- Internet: A subscriber connects to the Internet
6- Pay Invoice: A subscriber pays his/her invoice
7- Change ServiceProvider: A subscriber changes his/her provider
8- Change Limit: A subscriber changes his/her usage limit for the Invoice
9- List all Subscribers (show s_id, isActive, s_provider, invoice)
10- List all Service Providers (show p_id, p_name, all costs, and discount)
11- Exit

- Whenever a new `Subscriber` or `ServiceProvider` is created, you will add this new item to the corresponding arrays or ArrayLists.
- To create a new service provider , you must get <`voiceCallCost`> <`messagingCost`> <`internetCost`> <`discountRatio`> from the user.
- To create a new subscriber, <`name`> <`age`> <`p_id`> <`usageLimit`> must be taken from the user.
- To execute a Voice Call, you must get <`caller's s_id`> <`receiver's s_id`> <`time in minutes`> from the user. Both caller and receiver will be charged based on their service providers' costs.

COMP1112 Object Oriented Programming
Spring 2023, Project #1,
Due: 21 April, 23:50 via Blackboard

IŞIK UNIVERSITY
COMPUTER SCIENCE AND ENGINEERING

- To execute messaging, you must get <`sender's s_id`> <`receiver's s_id`> <`quantity`> from the user. Only sender will be charged based on his service provider's cost.
- To execute connection to the Internet, you must get <`s_id`> <`amount in MBs`> from the user. The subscriber will be charged based on his service provider's cost.
- To pay the Invoice, you must get <`s_id`> <`amount of money to pay`> from the user.
- To change service provider, <`s_id`> <`p_id of new subscriber`> must be taken from the user.
- To change usage limit, <`s_id`> <`new limit`> must be taken from the user.

When the user chooses the option for **exit** the program, print the following **output information**:
For each service provider: <p_id> : <total voice call time> <total number of messages> <total MBs of Internet usage>
For each subscriber: <s_id> : <total spending> <current spending>
<s_id> < name> <total voice call time> of the subscriber, who had the most amount of total voice call time in terms of minutes during the simulation
<s_id> < name> <total number of messages> of the subscriber, who sent the most number of messages during the simulation
<s_id> < name> <total amount of Internet usage> of the subscriber, who got connected the Internet the most during the simulation

Use the rules and definitions on the following table to calculate charges:

Subscriber Definitions		
`teenager`	`age`	between 10 and 18 (exclusive)
`elderly`	`age`	higher than or equal to 65
Charge Calculations and Discounts		
`Voice Call Charge`	`the amount of minutes` x the service provider's `voiceCallCost` per minute	For a `teenager` or an `elderly`, a discount will be applied on voice call charge based on `discountRatio`
`Voice Call Charge`	The first 5 minutes are free of charge	for `teenagers`
`Messaging Charge`	`the number of the messages` x the service provider's `messagingCost` per message	For sender and receiver having the same service provider, a discount will be applied on messaging charge based on `discountRatio`
`Messaging Charge`	The first 10 messages are free of charge	for `teenagers`
`Internet Usage Charge`	amount of MBs x `internetCost` per MB	For `teenagers`, the first 5 MBs of the Internet is free of charge
Subscriber's status		
• Before any type of communication, the `usageLimit` of the invoice and `isActive` status of the subscriber should be checked. • If a subscriber's `currentSpending` exceeds the `usageLimit` or he is in active, no more communication can be executed. • A service provider calculates the corresponding costs **only if** the subscriber is its customer.		

Project steps:
- First design your solution, **draw UML diagram**, define class relationships (association, aggregation, composition), multiplicity and direction of the association
- Apply **data encapsulation** and **visibility rules** on your solution
- Implement your solution.
- You will submit **four classes** and your **UML diagram** as a JPEG file.