ITCS209 Object Oriented Programming

Project 3

“BuddyShip”

Submit to

Aj. Suppawong Tuarob

Presented by

6388012 Rujiphart Charatvaraphan

6388019 Sorawanan Jeamjantarasakhon

6388022 Pattanan Korkiattrakool

Group 12 Section 1

Welcome to

BuddyShip

Better life, better press “yes”

About “BuddyShip”

“**BuddyShip**” is a mobile application for users who want to find friends with a matching lifestyle, or users who want to make new friends in their daily activities. BuddyShip is a matching buddy application with 2 types of packages for users to choose which is FreeMember and PremiumMember.

The difference between FreeMember and PremiumMember

Both members can access the same features but for ‘PremiumMember’ can choose every member that they want to chat with, including other ‘PremiumMember’ while ‘FreeMember’ can chat to every member except members who have a badge “Premium” on their photo and account.

5 features in “BuddyShip”

1) **findDriverBuddy –** For users who want a buddy to drive together.

2) **findMealBuddy –** For users who want a buddy to have meal together.

3) **findShoppingBuddy –** For users who want a friend to shop together.

4) **findTravelBuddy -** For users who want a friend to travel together.

5) **findReadingBuddy –** For users who want a friend to read together.

The benefit of BuddyShip Application

1. **A chance to meet new friend from all over country**

   - A user who uses BuddyShip can choose the radius of the distance to find the other user, so you may find your buddy from all over the country depending on your radius of distance.

2. **You can make new friends**

   - Not everyone has a chance or time to travel around a country and more than that not everyone will have a friend who has the same lifestyle just like you. Thus, BuddyShip will lead you to meet a new friend who absolutely has the same lifestyle as you (It can lead you to a serious relationship too).

3. **Intercultural exchange**

   - BuddyShip welcomes all members from all over the world, so 2 members can exchange their language and culture.

4. **Escape from the old world**

   - BuddyShip is a new way to explore a new lifestyle. For example, you can invite a buddy to go for a long drive though you cannot drive.

BuddyShip Requirements

When the user first signs up, the user must create an account

by using

1. Name
2. Usernam
3. Password
4. Age
5. Gender
6. Address

Note* Where both Username and password will be used for login to an account and Username will display name on user profile so every member will see this name.

API Document

BuddyShip

Class: Sticker

public class Sticker

Class description: This class is the object of each Sticker. It contains its name and its unicode.

Field:

Fields	**Description**
public String **stickerName**	Sticker's name
public String **stickerUnicode**	Sticker's unicode

Method:

Modifier and types	**Method name and Description**
public string	**getStickerName()** - This method is for getting StickerName field **Parameters** - None
public char	**getStickerUnicode()** - This method is for getting stickerUnicode field **Parameters** - None

Class: Member

public class Member

Class description: This class is the object of each Member. It contains Member's name, Member's ID, Member's age, Member's genre, Member's username, Member's password, and Member's hometown.

Constructor:

Constructor	**Description**
Member(String CustName, String username, int pwd, int age, String genre, String hometown)	Construct new Member. Member's ID will be initialized by running number **runningID.**

Field:

Fields	**Description**
public String **custName**	Member's name.
public int **custID**	Member's ID.
public int **age**	Member's age.
public String **genre**	Member's genre.
private String **username**	Member's username (for login).
private int **pwd**	Member's password (for login).
private static int **runningID**	The ordered integer, used for assigning the Member's ID.
Protected String **hometown**	Member's hometown.

Method:

Modifier and types	**Method name and Description**
public String	**getCustomerName()** - This method is for getting a member's name.
public int	**getCustomerID()** - This method is for getting a member's name.
public int	**getAge()** - This method is for getting a member's age.
public String	**getGenre()** - This method is for getting a member's genre.
public String	**getUsername()** - This method is for getting a member's username for login.
public int	**getPassword()** - This method is for getting a member's password for login.
public String	**getHometown()** - This method is for getting a member's hometown.

Class: DataManagement

public class DataManagement

Class description: This class is for managing all members' data, sticker sets and each customer that are waiting for each matching queue.

Field:

Fields	**Description**
public static Map <String,Sticker> **StickerSet**	Sticker's set(Collection)
Private static Map <String,Sticker> **StickerSetPremium**	Sticker's set(Collection)
private Map <int,Member> **MemberData**	Data set of **FreeMember**
private Map <int,Member> **PremiumMemberData**	Data set of **PremiumMember**
private Map <int,Member> **MemberDriver**	Data set of Member who would like to find the Driver Buddy.
private Map <int,Member> **MemberMeal**	Data set of Member who would like to find the Meal Buddy.
private Map <int,Member> **MemberShop**	Data set of Member who would like to find the Shopping Buddy.
private Map <int,Member> **MemberTravel**	Data set of Member who would like to find the Travel Buddy.
private Map <int,Member> **MemberRead**	Data set of Member who would like to find the Reading Buddy.

Method:

Modifier and types	**Method name and Description**

public Map <int, Member>	**getMemberCustomerData()** - This method is for getting (only **FreeMember** users) data and information such as Member’s id, username, and password. - This method will get a whole set of data in **MemberData.**
public Member	**getCustomerData(int MemberID)** - This method is an overloading method from **getCustomerData().** (Get only one Member who matches the **MemberID)**
public void	**addCustomerData(Member)** - This method will add data to **MemberData.**
public Map <int, Member>	**getPremiumCustomerData()** - This method is for getting (only **PremiumMember** users) data and information such as Member’s id, username, and password. - This method will get a whole set of data in **PremiumMemberData.**
public Member	**getPremiumCustomerData (int MemberID)** - This method is an overloading method from **getPremiumCustomerData().** (Get only one Member who matches the **MemberID)**
public void	**addPremiumCustomerData(Member member)** - This method will add data to **PremiumMemberData**
public static Map <String, Sticker>	**getStickerSet()** - This method will be used as a

	getter of **StickerSet** that will receive **StickerName** and **StickerUnicode.**
public static Sticker	**getStickerSet(String StickerName)** - This method is an overloading method from **getStickerSet().** (Get only one Sticker which matches the **StickerName)**
public static void	**addStickerSet(Sticker sticker)** - This method will add sticker’s data to **StickerSet.**
public static Map<String, Sticker>	**getStickerPremium(boolean isPremium)** - This method will be used as a getter of **StickerSetPremium** that will receive **StickerName** and **StickerUnicode.** (When **isPremium** is true.)
public Sticker	**getStickerSetPremium(String StickerName)** - This method is an overloading method from **getStickerSetPremium().** - (Get only one Member who matches the **StickerName)**
public void	**setCustomerDriver(Customer)** - If a user(both **FreeMember** and **PremiumMember**) select **features “Driver”** it will use data from public Member **findDriverBuddy** in **FreeMember** and **PremiumMember** classes.
public void	**setCustomerMeal(Customer)** - If a user(both **FreeMember** and **PremiumMember**) select **features “Meal”** it will use data from public Member **findMealBuddy** in **FreeMember** and **PremiumMember** classes.

public void	**setCustomerShop(Customer)** - If a user(both **FreeMember** and **PremiumMember**) select **features “Shop”** it will use data from public Member **findShoppingBuddy** in **FreeMember** and **PremiumMember** classes.
public void	**setCustomerTravel(Customer)** - If a user(both **FreeMember** and **PremiumMember**) select **features “Travel”** it will use data from public Member **findTravelBuddy** in **FreeMember** and **PremiumMember.**
public void	**setCustomerRead(Customer)** - If a user(both **FreeMember** and **PremiumMember**) select **features “Read”** it will use data from **public Member findReadingBuddy** in **FreeMember** and **PremiumMember** classes.

Class: FreeMember

public class FreeMember

Class description: This class contains all features in our application. However, this class is for the free members that there will are some different details compared to premium members. FreeMember is able to find only FreeMember buddy.

Constructor:

Constructor	**Description**
FreeMember(String CustName, String username, int pwd, int age, String genre, String hometown)	Construct new FreeMember. This constructor will be sent to Member.

Field:

Fields	**Description**
Private static boolean **isPremium**	This class is FreeMember, this field will always be false.
private boolean **isMatch**	This field tells that this Member is matched with a buddy. If matched, this field will be true.
private int **ticketNumber**	This member's ticket number, it will be created by randoming hash code.
private Member **myBuddy**	This customer's buddy

Method:

Modifier and types	**Method name and Description**
public static void	**selectOption()** *** HT for Hometown** - This method is for providing the interface to input the feature this Member would like to get - Press 1: findDriverBuddy(HT)* - Press 2: findMealBuddy(HT) - Press 3: findShoppingBuddy(HT)

	- Press 4: findTravelBuddy(HT) - Press 5: findReadingBuddy(HT) - Press 6 to exit the selectOption.
public void	**chat(String message)** - This method is for sending the message to another member for making the match. - If **message** is "Will be buddy me?" it will call **selectMatchOption()**.
public void	**sendSticker(String stickerName)** - This method is for sending sticker to another member. - It will send the sticker's unicode by searching the sticker name. - For the free member, this method can access only the **DataManagement.StickerSet**
public Member	**findDriverBuddy(String Hometown)** - This method is for finding another buddy for driving together. - It will return **myBuddy**. **Process of this method** 1) It will perform the information of all members that are interested in the same feature by calling **performData("DRIVER")**. 2) This customer is added into **DataManagement.MemberDriver<this.custID,this(this member)>**. 3) When we get another FreeMember from selecting the FreeMember in **DataManagement.MemberDriver** with the same **Hometown**, call method **chat(String message)** by inputting the message, call it by while loop until we input "Will be buddy

	me?”. 4) When we are out of loop in 3), we will go to **selectMatchOption()**. 5) If both member and another member press yes, it means this member and another member are buddies. **isMatch** will be set to true, **myBuddy** is equal to another member, the data of this customer in **DataManagement.MemberDriver** is removed, and lastly send the ticket with **ticketNumber** to buddy by calling method **sendTicket(this.isPremium, ticketNumber,myBuddy.customerID)**. Else if either member presses no, isMatch is still false. Then we will get back to **selectOption()**.
public Member	**findMealBuddy(String Hometown)** - This method is for finding another buddy for taking a meal together. - It will return **myBuddy**. **Process of this method** 1) It will perform the information of all members that are interested in the same feature by calling **performData(“MEAL”)**.

	2) This customer is added into **DataManagement.MemberMeal<this.custID,this(this member)>.** 3) When we get another FreeMember from selecting the FreeMember in **DataManagement.MemberMeal** with the same Hometown, call method **chat(String message)** by inputting the message, call it by while loop until we input "Will be buddy me?". 4) When we are out of loop in 3), we will go to **selectMatchOption().** 5) If both member and another member press yes, it means this member and another member are buddies. **isMatch** will be set to true, **myBuddy** is equal to another member, the data of this customer in **DataManagement.MemberMeal** is removed, and lastly send the ticket with **ticketNumber** to buddy by calling method **sendTicket(this.isPremium, ticketNumber,myBuddy.customerID).** Else if either member presses no, isMatch is still false. Then we will get back to **selectOption().**
public Member	**findShoppingBuddy(String Hometown)** - This method is for finding another buddy for shopping together. - It will return **myBuddy.** **Process of this method** 1) It will perform the information of all members that are interested in the same feature by calling **performData("SHOP")**

	2) This customer is added into **DataManagement.MemberShop<this.custID,this(this member)>.** 3) When we get another FreeMember from selecting the FreeMember in **DataManagement.MemberShop** with the same **Hometown**, call method **chat(String message)** by inputting the message, call it by while loop until we input “Will be buddy me?”. 4) When we are out of loop in 3), we will go to **selectMatchOption().** 5) If both member and another member press yes, it means this member and another member are buddies. **isMatch** will be set to true, **myBuddy** is equal to another member, the data of this customer in **DataManagement.MemberShop** is removed, and lastly send the ticket with **ticketNumber** to buddy by calling method **sendTicket(this.isPremium, ticketNumber,myBuddy.customerID).** Else if either member presses no, isMatch is still false. Then we will get back to **selectOption().**
public Member	**findTravelBuddy(String Hometown)** - This method is for finding another buddy for traveling together. - It will return **myBuddy.** **Process of this method** 1) It will perform the information of all members that are interested in the same feature by calling **performData(“TRAVEL”)**

	2) This customer is added into **DataManagement.MemberTravel<this.custID,this(this member)>.** 3) When we get another FreeMember from selecting the FreeMember in **DataManagement.MemberTravel** with the same **Hometown**, call method **chat(String message)** by inputting the message, call it by while loop until we input "Will be buddy me?". 4) When we are out of loop in 3), we will go to **selectMatchOption().** 5) If both member and another member press yes, it means this member and another member are buddies. **isMatch** will be set to true, **myBuddy** is equal to another member, the data of this customer in **DataManagement.MemberTravel** is removed, and lastly send the ticket with **ticketNumber** to buddy by calling method **sendTicket(this.isPremium, ticketNumber,myBuddy.customerID).** Else if either member presses no, isMatch is still false. Then we will get back to **selectOption().**

public Member	**findReadingBuddy(String Hometown)** - This method is for finding another buddy for shopping together. - It will return **myBuddy.** **Process of this method** 1) It will perform the information of all members that are interested in the same feature by calling **performData(“READ”)** 2) This customer is added into **DataManagement.MemberRead<this.custID,this(this member)>.** 3) When we get another FreeMember from selecting the FreeMember in **DataManagement.MemberRead** with the same **Hometown,** call method **chat(String message)** by inputting the message, call it by while loop until we input “Will be buddy me?”. 4) When we are out of loop in 3), we will go to **selectMatchOption().** 5) If both member and another member press yes, it means this member and another member are buddies, **myBuddy** is equal to another member, the data of this customer in **DataManagement.MemberRead** is removed, and lastly send the ticket with **ticketNumber** to buddy by calling method **sendTicket(this.isPremium, ticketNumber,myBuddy.customerID).** Else if either member presses no, isMatch is still false. Then we will get back to **selectOption().**

public boolean	**checkTicket(int buddyTicketNumber)** - This method is for checking that the buddy’s ticket is the same as this member’s. - It will return true if both ticket numbers are corresponding. - Else, return false.
public boolean	**sendTicket(boolean isPremium, int ticketNumber , int myBuddyID)** - This method is for sending this member’s ticketNumber to the buddy for checking, if it is successfully sent, return true, else return false.
public static void	**performData(String feature)** - This method is for performing the data from provided parameter - If **feature** is “**DRIVER**” it will perform the data from **DataManagement.MemberDriver.** Else if **feature** is “**MEAL**” it will perform the data from **DataManagement.MemberMeal.** Else if **feature** is “**SHOP**” it will perform the data from **DataManagement.MemberShop.** Else if **feature** is “**TRAVEL**” it will perform the data from **DataManagement.MemberTravel.** Else if **feature** is “**READ**” it will perform the data from **DataManagement.MemberRead.**

public void	**selectMatchOption()** - This method is for selecting the match option. - If both member and another member press **yes**, it means this member and another member are buddies. (**isMatch** = true) Else if either member presses no, isMatch is still false. Then we will get back to **selectOption()**.

Class: PremiumMember

public class PremiumMember

Class description: This class contains all features in our application. However, this class is for the premium member. They can access our all features with full performance. (They can send the premium sticker sets and find both free and premium member)

Constructor

Constructor	Description
PremiumMember(String CustName, String username, int pwd, int age, String genre, String hometown)	Construct new PremiumMember. This constructor will be sent to Member.

Field:

Fields	Description
private static boolean **isPremium**	This class is FreeMember, this field will always be true.
private boolean **isMatch**	This field tells that this Member is matched with a buddy. If matched, this field will be true.
private int **ticketNumber**	This member's ticket number, it will be created if this customer is Matched
private Member **myBuddy**	This customer's buddy

Method:

Modifier and types	Method name and Description
public static void	**selectOption()** * HT for Hometown - This method is for providing the interface to input the feature this Member would like to get - Press 1: findDriverBuddy(HT)* - Press 2: findMealBuddy(HT) - Press 3: findShoppingBuddy(HT) - Press 4: findTravelBuddy(HT) - Press 5: findReadingBuddy(HT) - Press 6 to exit the selectOption.
public void	**chat(String message)** - This method is for sending the message to another member for making the match. - If **message** is "Will be buddy me?" it will call **selectMatchOption()**.
public void	**sendSticker(String stickerName)** - This method is for sending sticker to another member. - It will send the sticker's unicode by searching the sticker name. - For the premium member, this method can access both sticker sets(**DataManagement.StickerSet & DataManagement.StickerSetPremium)**
public Member	**findDriverBuddy(String Hometown)** - This method is for finding another buddy for driving together. - It will return **myBuddy**. **Process of this method** 1) It will perform the information of all members that are interested in the same feature by calling **performData("DRIVER")**. 2) This customer is added into

	DataManagement.MemberDriver<this.custID,this(this member)>. 3) When we get another FreeMember from selecting the FreeMember in **DataManagement.MemberDriver** with the same **Hometown**, call method **chat(String message)** by inputting the message, call it by while loop until we input “Will be buddy me?”. 4) When we are out of loop in 3), we will go to **selectMatchOption()**. 5) If both members and another member press yes, it means this member and another member are buddies. **isMatch** will be set to true, **myBuddy** is equal to another member, the data of this customer in **DataManagement.MemberDriver** is removed, and lastly send the ticket with **ticketNumber** to buddy by calling method **sendTicket(this.isPremium, ticketNumber,myBuddy.customerID)**. Else if either member presses no, isMatch is still false. Then we will get back to **selectOption()**.
public Member	**findMealBuddy(String Hometown)** - This method is for finding another buddy for taking a meal together. - It will return **myBuddy**. **Process of this method** 1) It will perform the information of

	all members that are interested in the same feature by calling **performData("MEAL").** 2) This customer is added into **DataManagement.MemberMeal<this.custID,this(this member)>.** 3) When we get another FreeMember from selecting the FreeMember in **DataManagement.MemberMeal** with the same Hometown, call method **chat(String message)** by inputting the message, call it by while loop until we input "Will be buddy me?". 4) When we are out of loop in 3), we will go to **selectMatchOption().** 5) If both members and another member press yes, it means this member and another member are buddies. **isMatch** will be set to true, **myBuddy** is equal to another member, the data of this customer in **DataManagement.MemberMeal** is removed, and lastly send the ticket with **ticketNumber** to buddy by calling method **sendTicket(this.isPremium, ticketNumber,myBuddy.customerID).** Else if either member presses no, isMatch is still false. Then we will get back to **selectOption().**
public Member	**findShoppingBuddy(String Hometown)** - This method is for finding another buddy for shopping together.

	- It will return **myBuddy.** **Process of this method** 1) It will perform the information of all members that are interested in the same feature by calling **performData("SHOP")** 2) This customer is added into **DataManagement.MemberShop<this.custID,this(this member)>.** 3) When we get another FreeMember from selecting the FreeMember in **DataManagement.MemberShop** with the same **Hometown**, call method **chat(String message)** by inputting the message, call it by while loop until we input "Will be buddy me?". 4) When we are out of loop in 3), we will go to **selectMatchOption().** 5) If both members and another member press yes, it means this member and another member are buddies. **isMatch** will be set to true, **myBuddy** is equal to another member, the data of this customer in **DataManagement.MemberShop** is removed, and lastly send the ticket with **ticketNumber** to buddy by calling method **sendTicket(this.isPremium, ticketNumber,myBuddy.customerID).** Else if either member presses no, isMatch is still false. Then we will get back to **selectOption().**
public Member	**findTravelBuddy(String Hometown)** - This method is for finding another buddy for traveling together.

	- It will return **myBuddy.** **Process of this method** 1) It will perform the information of all members that are interested in the same feature by calling **performData(“TRAVEL”)** 2) This customer is added into **DataManagement.MemberTravel<this.custID,this(this member)>.** 3) When we get another FreeMember from selecting the FreeMember in **DataManagement.MemberTravel** with the same **Hometown**, call method **chat(String message)** by inputting the message, call it by while loop until we input “Will be buddy me?”. 4) When we are out of loop in 3), we will go to **selectMatchOption().** 5) If both members and another member press yes, it means this member and another member are buddies. **isMatch** will be set to true, **myBuddy** is equal to another member, the data of this customer in **DataManagement.MemberTravel** is removed, and lastly send the ticket with **ticketNumber** to buddy by calling method **sendTicket(this.isPremium, ticketNumber,myBuddy.customerID).** Else if either member presses no, isMatch is still false. Then we will get back to **selectOption().**

public Member	**findReadingBuddy(String Hometown)** - This method is for finding another buddy for reading together. - It will return **myBuddy.** **Process of this method** 1) It will perform the information of all members that are interested in the same feature by calling **performData("READ")** 2) This customer is added into **DataManagement.MemberRead<this.custID,this(this member)>.** 3) When we get another FreeMember from selecting the FreeMember in **DataManagement.MemberRead** with the same **Hometown**, call method **chat(String message)** by inputting the message, call it by while loop until we input "Will be buddy me?". 4) When we are out of loop in 3), we will go to **selectMatchOption().** 5) If both member and another member press yes, it means this member and another member are buddies, **myBuddy** is equal to another member, the data of this customer in **DataManagement.MemberRead** is removed, and lastly send the ticket with **ticketNumber** to buddy by calling method **sendTicket(this.isPremium, ticketNumber,myBuddy.customerID).** Else if either member presses no, isMatch is still false. Then we will get back to **selectOption().**

public boolean	**checkTicket(int buddyTicketNumber)** - This method is for checking that the buddy’s ticket is the same as this member’s. - It will return true if both ticket numbers are corresponding. - Else, return false.
public boolean	**sendTicket(boolean isPremium, int ticketNumber , int myBuddyID)** - This method is for sending this member’s ticketNumber to the buddy for checking, if it is successfully sent, return true, else return false.
public static void	**performData(String feature)** - This method is for performing the data from provided parameter - If **feature** is “**DRIVER**” it will perform the data from **DataManagement.MemberDriver.** Else if **feature** is “**MEAL**” it will perform the data from **DataManagement.MemberMeal.** Else if **feature** is “**SHOP**” it will perform the data from **DataManagement.MemberShop.** Else if **feature** is “**TRAVEL**” it will perform the data from **DataManagement.MemberTravel.** Else if **feature** is “**READ**” it will perform the data from **DataManagement.MemberRead.**

public void	**selectMatchOption()** - This method is for selecting the match option. - If both member and another member press **yes**, it means this member and another member are buddies. (**isMatch** = true) Else if either member presses no, isMatch is still false. Then we will get back to **selectOption()**.

Class: App

public class App

Class description: This class is a display page for users when they activate the application.

Method:

Modifier and types	**Method name and Description**
public void	**login()** - This method is provided for users who have already signed up an account and want to sign in to their account again. - This method will use data from **FreeMember** or **PremiumMember** (Depends on the package that the user has chosen)to check if a user already has an account.
public void	**register()** - **This method is provided for users who don't have an account.** - New users have to fill their information on this page and their data will be sent to **FreeMember** or **PremiumMember**(Depends on the package that the user has chosen).

Class: Features

Interface features

Class description: This class is an interface class which FreeMember and PremiumMember implement. It contains all features in the application.

Method:

Modifier and types	Method name and Description
public static void	**SelectOption()** - This method is for providing the interface to input the feature this Member would like to get. - There are 5 features(input 1-6) - Press 1: findDriverBuddy(HT)* - Press 2: findMealBuddy(HT) - Press 3: findShoppingBuddy(HT) - Press 4: findTravelBuddy(HT) - Press 5: findReadingBuddy(HT) - Press 6 to exit the selectOption.
public Member	**findDriverBuddy(String Hometown)** - This method is for users to find a buddy for driving together. - It will return **myBuddy**. **Process of this method:** 1) It will perform the information of all members that are interested in the same feature by calling **performData(“DRIVER”).** **2)** This user is added into **DataManagement.MemberDriver<this.custID,this(this member)>.** 3) When an application gets another FreeMember from selecting the FreeMember in **DataManagement.MemberDriver** with the same **Hometown,** call

	method **chat(String message)** by inputting the message, call it by while loop until user input message “Will be buddy me?”. 4) When it is out of loop in 3), we will go to **selectMatchOption().** 5) If the other member who received a message “Will you buddy me?” press yes, it means both members are buddies. **isMatch** will be set to true, **myBuddy** is equal to another member, the data of this customer in **DataManagement.MemberDriver** is removed, and lastly send the ticket with **ticketNumber** both member who are buddy by calling method **sendTicket(this.isPremium, ticketNumber,myBuddy.customerID).** Else if either member presses no, isMatch is still false. Then we will get back to **selectOption().**
public Member	**findMealBuddy(String Hometown)** - This method is for users to find a buddy for having a meal together. - It will return **myBuddy.** **Process of this method:** 1) It will perform the information of all members that are interested in the same feature by calling **performData(“MEAL”).** 2) This user is added into **DataManagement.MemberMeal<this**

	.custID,this(this member)>. 3) When an application gets another FreeMember from selecting the FreeMember in **DataManagement.MemberMeal** with the same **Hometown**, call method **chat(String message)** by inputting the message, call it by while loop until user input message “Will be buddy me?”. 4) When it is out of loop in 3), we will go to **selectMatchOption().** 5) If the other member who received a message “Will you buddy me?” press yes, it means both members are buddies. **isMatch** will be set to true, **myBuddy** is equal to another member, the data of this customer in **DataManagement.MemberMeal** is removed, and lastly send the ticket with **ticketNumber** both member who are buddy by calling method **sendTicket(this.isPremium, ticketNumber,myBuddy.customerID).** Else if either member presses no, isMatch is still false. Then we will get back to **selectOption().**
public Member	**findShoppingBuddy(String Hometown)** - This method is for users who want to find a buddy for shopping together. - It will return **myBuddy**. **Process of this method:**

	1) It will perform the information of all members that are interested in the same feature by calling **performData("SHOP")** **2)** This user is added into **DataManagement.MemberShop<this.custID,this(this member)>.** 3) When an application gets another FreeMember from selecting the FreeMember in **DataManagement.MemberShop** with the same **Hometown**, call method **chat(String message)** by inputting the message, call it by while loop until user input message "Will be buddy me?". **4)** When it is out of loop in 3), we will go to **selectMatchOption().** **5)** If the other member who received a message "Will you buddy me?" press yes, it means both members are buddies. **isMatch** will be set to true, **myBuddy** is equal to another member, the data of this customer in **DataManagement.MemberShop** is removed, and lastly send the ticket with **ticketNumber** to buddy by calling method **sendTicket(this.isPremium, ticketNumber,myBuddy.customerID).** Else if either member presses no, isMatch is still false. Then we will get back to **selectOption().**
public Member	**findTravelBuddy(String Hometown)** - This method is for users to

find a buddy for traveling together.

- It will return **myBuddy.**

Process of this method:

1) It will perform the information of all members that are interested in the same feature by calling **performData("TRAVEL").**

2) This customer is added into **DataManagement.MemberTravel<this.custID,this(this member)>.**

3) When an application gets another FreeMember from selecting the FreeMember in **DataManagement.MemberTravel** with the same **Hometown**, call method **chat(String message)** by inputting the message, call it by while loop until user input message "Will be buddy me?".

4) When it is out of loop in 3), we will go to **selectMatchOption().**

5) If the other member who received a message "Will you buddy me?" press yes, it means both members are buddies. **isMatch** will be set to true, **myBuddy** is equal to another member, the data of this customer in **DataManagement.MemberTravel** is removed, and lastly send the ticket with **ticketNumber** to buddy by calling method **sendTicket(this.isPremium, ticketNumber,myBuddy.customerI**

	D). Else if either member presses no, isMatch is still false. Then we will get back to **selectOption().**
public Member	**findReadingBuddy(String Hometown)** - **This method is for users who want a buddy for reading books together.** - It will return **myBuddy**. **Process of this method:** 1) It will perform the information of all members that are interested in the same feature by calling **performData("READ").** 2) This user is added into **DataManagement.MemberRead<this .custID,this(this member)>.** 3) When an application gets another FreeMember from selecting the FreeMember in **DataManagement.MemberRead** with the same **Hometown**, call method **chat(String message)** by inputting the message, call it by while loop until user input message "Will be buddy me?". 4) When it is out of loop in 3), we will go to **selectMatchOption().** **5)** If the other member who received a message "Will you buddy me?" press yes, it means both members are buddies. **isMatch** will be set to true, **myBuddy** is equal to another member, the data of this customer in

	DataManagement.MemberRead is removed, and lastly send the ticket with **ticketNumber** to buddy by calling method **sendTicket(this.isPremium, ticketNumber,myBuddy.customerID).** Else if either member presses no, isMatch is still false. Then we will get back to **selectOption().**

Class: ChatPlatform

Interface ChatPlatform

Class description: This class is an interface class which FreeMember and PremiumMember implement. It contains chat, sending stickers and tickets and a select option.

Method:

Modifier and types	**Method name and Description**
public void	**chat(String message)** - This method is for sending the message to another member to make matching. - If the message is “Will be buddy me?” it will call **selectMatchOption()**.
public void	**sendSticker(String stickerName)** - This method is for sending stickers to another member. - It will send the sticker’s unicode by searching the sticker name. - For the free member, this method can access only the **DataManagement.StickerSet**
public boolean	**sendTicket(boolean isPremium, int ticketNumber, int myBuddyID)** - This method is for sending this member’s ticketNumber to the buddy for checking, if it is successfully sent, return true, else return false.
public boolean	**checkTicket(int buddyTicketNumber)** - This method is for checking that the buddy’s ticket is the same as this member’s. - It will return true if both ticket numbers are corresponding. - Else, return false.
public void	**selectMatchOption()** - This method is for selecting the match option. - If the other member who received

	a message “Will you buddy me?” press **yes**,it means both members are buddies. (**isMatch** = true) Else if either member presses no, isMatch is still false. Then we will get back to **selectOption()**.