Exercise Set: Implementing an Object-Oriented Design

In this exercise set, we have marked questions we think are harder than others with a **[‡]**. We have also marked questions for which solutions are provided at the end of the set (**[SP]**). To check solutions for other questions than those marked with **[SP]**, ask one of the instructors or TAs or post a question to the google group!

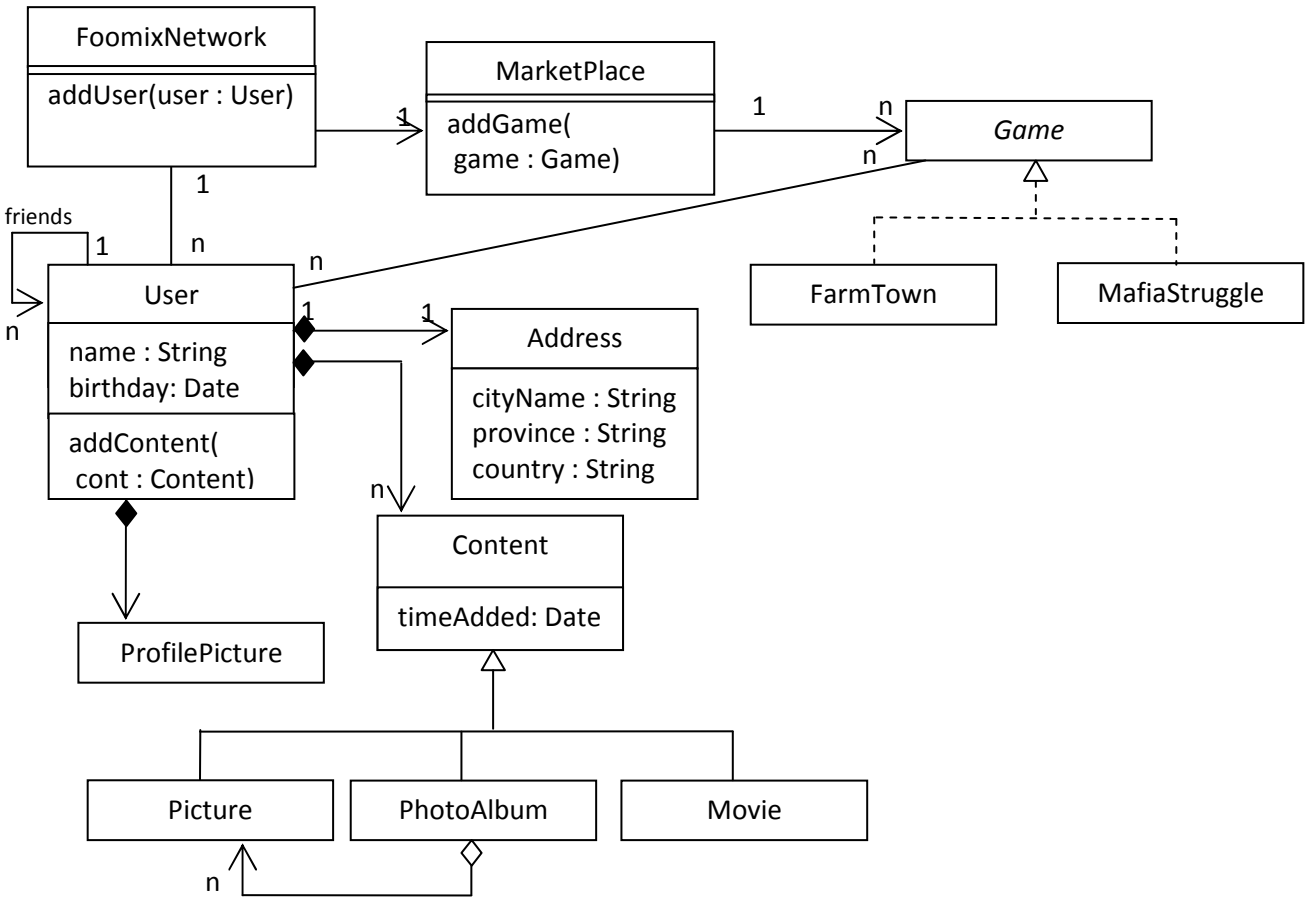
Social Network Application
Given the following partially specified requirements for a social network application Foomix and the Object-Oriented UML Class Diagram:

The Social Network Application Foomix allows friends to connect with each other online and share information such as their photo albums or movies. Foomix allows a user to register and fill in personal information. In particular, it lets the user specify her name, birthday, address and a profile picture.

Once registered, a user can create photo albums that each has a list of pictures that the user has uploaded to Foomix. The user can also upload movies to his profile. For each uploaded content, Foomix stores the time when the content was added.

Each user has friends he is connected to and can add more users as friends. For this, a user can list all or search for users registered with the FoomixNetwork.

Foomix also provides a marketplace that offers a variety of games a user can play. FarmTown and MafiaStruggle are two of these games. New games can be added any time to the marketplace. Once a user plays one of the offered games, the game remembers the user and its highest score in a ranking. Furthermore, users can rank the games.



1. Implement the design in Java. Therefore:
a) implement the classes (describe what packages you would create and what classes go in them);
b) implement the associations (don't forget to implement getter and setter methods; note, I already partially specified three setter methods in the design, `addUser`, `addApplication` and `addContent`). **[‡]**

2. Think about the types of collections from the Java Collections Framework you would use to implement the relations and describe why. In particular, think how you would implement the user rankings in a Game.

3. Implement `equals` and `hashCode` methods for type User (check the reading to make sure you didn't forget anything). **[SP]**

4. Implement `Comparable` for pictures in a photo album so that the pictures will be sorted by the `timeAdded` when `Collections.sort(…)` is used on a collection of `Picture`s

5. Implement `Comparable` for type `User` so that user profiles can be sorted by the `Address` (in alphabetical order first by country, then by province and last by city name, e.g. a user from BC, Canada would be after a user from Alberta, Canada, but before a user from Bavaria, Germany). Therefore, delegate the comparison to `Address`. (Delegation is described in the GUI reading ubc-cpsc-210-GUI.pdf on pages 3-5) **[‡]**

6. Create a class `MemoryLeakDemo` and copy the following code into the class. **[‡]**
Note: Make sure you have the following constructors in your implementation: `FoomixNetwork()`, `MarketPlace()`, `Game(String name)`, `User(String name, java.util.Date birthday)`, and `Content(java.util.Date timeAdded)`.

a) Run the Eclipse Memory Analyzer Tool (MAT) on memory dumped from the MemoryLeakDemo program and determine which objects are taking most space.

b) Draw an object reference graph for the code below and the one you implemented starting from the `main` method.

```
public static void main(String[] args) {
	FoomixNetwork fn = new FoomixNetwork();
	addMarketPlaceAndGames(fn);
	addUsers();

}

public void addMarketPlace(FoomixNetwork fn) {
	MarketPlace mp = new MarketPlace();
// check in your implementation that you called the setter
// setMarketPlace in the FoomixNetwork, otherwise change the method in here
	fn.setMarketPlace(mp);
	addGames(fn);
	addUsers(fn);
}


public void addGames(FoomixNetwork fn) {
// make sure in your implementation that you called the getter getMarketPlace
// in the FoomixNetwork, otherwise change the method in here
	for (int i=0; i<100; i++) {
		fn.getMarketPlace().addApplication(new Game("Game "+i));
	}
}

public void addUsers(FoomixNetwork fn) {
	for (int i=0; i<100; i++) {
		addUser(fn,i);
	}
}

public void addUser(FoomixNetwork fn, int j) {
	List<Date> dates = new ArrayList<Date>();
	Date bday = new Date();
	Bday.setYear(bday.getYear() - j);
	for (int i=0; i<1000; i++) {
		User user = new User("David",bday);
		fn.addUser(user);
```

```
            dates.add(new Date());
            Content album = new PhotoAlbum(dates.get(i));
            user.addContent(album);
        }
    }
```

SOLUTIONS:

4.

```
@Override
public  boolean equals(Object obj) {
	if (obj == null || getClass() != obj.getClass())
		return false;

	User other = (User) obj;

	return getName().equals(other.getName()) &&
			getBirthday().equals(other.getBirthday()) &&
			getAddress().equals(other.getAddress());
}

@Override
public int hashCode() {
	int result = 17;
	result = 37*result + getName().hashCode();
	result = 37*result + getBirthday().hashCode();
	result = 37*result + getAddress().hashCode();
	return result;
}
```