lab1: Spacebook



**US1**

* + A **person** has a **username** and a list of **friends**.
  + Username cannot be null, empty or spaces

**US2**

* + Becoming a friend means adding a bidirectional relationship.
  + Adding yourself should not be possible

**US3**

* + A person can **receive** **messages** from a friend only.
  + A message has a date, sender and body (text)
  + You can ask a person for all received messages

**Us4**

* + You can ask a person for all received messages, **sorted** by date.

**US5**

* + You can ask a person for received messages from a friend, sorted by date.

lab2: BUGFIX



**US**

* + A hunter can go hunting vampires between midnight and 6 am

**BUG**

* + Hunter can go hunting on midnight

EXPECTED: Can go hunting

ACTUAL: Cannot go hunting

lab3: PetShop

[](http://www.google.be/imgres?imgurl=http://watchcartoononline.com/thumbs/Littlest-Pet-Shop-2012-Episode-2-Blythe-s-Big-Adventure--Part-2.jpg&imgrefurl=http://www.watchcartoononline.com/littlest-pet-shop-2012-episode-2-blythes-big-adventure-part-2&h=230&w=336&tbnid=AI_vrGxZyCNKWM:&zoom=1&docid=nEWVSw31b-FqGM&ei=uHZzU5jQD4We0QXk24CYBQ&tbm=isch&ved=0CKMBEDMoPzA_&iact=rc&uact=3&dur=1427&page=2&start=36&ndsp=51)

**US1**

* + A PetShop has a stock. Items can be added

**US2**

* + A customer (which has an e-mail address) can buy an item, in which case the stock is decreased for that item.
  + The customer receives an e-mail that his item will be delivered the next day.

**US3**

* + If the customer tries to order an item that is out of stock, he gets an e-mail that the item will be delivered in the coming 10 days.
  + At the same time another e-mail is sent to order@petshop to order more of <item> and ship one to <customer>

lab4: FLIGHT



**US1**

* + A Flight has a list of Passengers
  + A Flight has an Id (int)
  + Passengers only have a name

**US2**

* + You can add Passengers to a Flight
  + You can get the Number of Passengers on a Flight
  + You can check if a Passenger is on a Flight

**US3**

* + A flight has a Maximum Number of Seats
  + When the flight is full, adding a Passenger causes an exception

**US4**

* + You can use a FlightBookingService to book a seat on a Flight using a FlightId and a person
  + Booking a Seat means adding the Passenger to the Flight
  + If the Flight with the given Id is not found an exception is thrown
  + A Flight is loaded from a database (customer is not sure of which DB to use)

lab5: MovieRental

[](http://www.google.be/url?sa=i&rct=j&q=&esrc=s&frm=1&source=images&cd=&cad=rja&uact=8&docid=SDpn8fa7XCGEUM&tbnid=uQUoF-w4N5pLsM:&ved=0CAUQjRw&url=http://www.slashgear.com/blockbuster-drops-price-of-in-store-rentals-to-compete-with-netflix-and-redbox-27155146/&ei=Lw52U6_HKoOh0QWs0IDoDw&bvm=bv.66699033,d.d2k&psig=AFQjCNHHjtqB9Rb_pQyuaM1xAXCFJRtHDQ&ust=1400332184447087)

Refactor the code using the catalog (<http://refactoring.com/catalog/>)

**US**

* Add a new type of movie (Adult)
* Price is € 5 per day
* Think SOLID
  + S – SRP – Single Reponsibility Principle
  + O – OCP – Open/Closed Principle
  + L – LSP – Liskov Substitution Principle
  + I - ISP - Interface Segregation Principle
  + D – DSP – Depencency Inversion Principle

lab6: The battle



**US1**

* A soldier must have a name. The name cannot be changed.
* A soldier is equipped with one weapon. By default “Bare fist”.
* Weapons do # amount of damage
* “Bare fist” does one damage

**US2**

* Soldiers can fight by comparing weapon damage.
* Every fight has a winner. The soldier with the strongest weapon wins, or if the same the attacker wins.

**US3**

* There are four types of weapons:
  + Axe 3
  + Sword 2
  + Spear 2
  + BareFist 1

**US4**

* I can enlist soldiers to an army so that I can obtain a list of the soldiers of that army
* The front man of an army is the first enlisted soldier
* Armies can engage in a war. Front man vs. Front man. A front man who loses, dies. Dead soldiers are removed from the army. ”Last man standing” wins.

**US5**

public interface IHeadquarters

{

int ReportEnlistment(string soldierName);

void ReportCasualty(int soldierId);

void ReportVictory(int remainingNumberOfSoldiers);

}

* When a soldier is enlisted, this is reported to HQ using the provided interface
* An enlisted soldier gets assigned an Id from HQ
* When a soldier dies, this is also reported to HQ using the soldiers Id
* When an army wins, this is reported with the remaining number of soldiers in the army

**US6**

* There are specialized weapons for the highly trained soldiers:
  + BroadSword 5
  + TwoHandedSword 2 \* damage of sword
  + BroadAxe 2 + Damage of axe
  + Trident 3 \* damage of spear
  + BrassKnuckles 2 \* damage of barefist

**BONUS**

**US7**

* Weapons have specification which grant a damage bonus of 3:
  + Axes over Spears
  + Spears over Swords
  + Swords over Axes
* In a soldier to soldier fight, these bones apply to the attacker
* If no specificiation can be applied (i.e. barefirst or same weapon), no bonus assigned.