lab1: Spacebook



**User Story (=”US”) 1**

* A **person** has a **username.**
* Username cannot be null, empty or spaces

**US2**

* A **person** has a list of **friends**.
* 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



***Already implemented (but clearly badly tested!) User Story:***

* 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 to it.

**US2**

* When a new item is added, the content of the item is appended to a file.

**Of course** your solution should contain unit tests.

Since your code consists of more than one class and of a dependency to the outside with the file writer however, you will also need:

1. an **integration test**  for the file writer. For this, complete the already existing skeleton of the provided integration test.
2. an **end-to-end test** to test your application from front to back, reusing some of the setup and code of the integration test.

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



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.

**US2**

* A soldier is equipped with one weapon. By default “Bare fist”.
* There are four types of weapons:
  + Axe
  + Sword
  + Spear
  + BareFist

**US3**

* A soldier must be able to fight with an other soldier.
* Fighting is done by comparing weapon damage
* Weapons do # amount of damage
  + Axe 3
  + Sword 2
  + Spear 2
  + BareFist 1
* Every fight has a winner. The soldier with the strongest weapon wins
* If the weapons are of equal strength, the attacker wins.

**US4**

* I can enroll soldiers in an army
* The front man of an army is the first enlisted soldier

**US5**

* 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.

**US6**

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

**US7**

* There are specialized weapons for the highly trained soldiers:
  + TwoHandedSword 5
  + BroadAxe 2 + Damage of axe
  + Trident 3 \* damage of spear
  + Magic potion 10 if opponents weapon has even damage

0 if opponents weapon has odd damage

**US8**

* 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 bonuses apply to the attacker
* If no specification can be applied (i.e. barefisted or same weapon), no bonus assigned.