ESIEA Software Architecture Project

01/18/2017

Requirements:

- A program that fits the requirements described in the “Subject” section,
- An implementation of a design pattern that fits this subject,
- A graphical interface.

Schedule :

- **Thursday 18th :** Teams set up, and logged.
- **Friday 19th :** Project set up, tools chosen, resources dispatched, and environment functional.
- **Monday 22nd :** Architecture defined, Software Model final, design pattern implemented.
- **Monday 29th :** Functional software completed, only finishing touches remaining.
- **Tuesday 30th :** Presentation + Code delivery.

Scoring Scale:

- **5 points : functionality**, no bugs,...
- **10 points : Software Architecture**
  - 2 pts : Model, 3 pts Environment (git/test/bugs…), 5 pts OOP concepts (Solid…)
- **5 points : Design Pattern** explanation

Presentation structure :

- Demonstration ~10 min
- Design pattern explanation ~5 min
- Intro/Wrap Up, ~2-3 min

Remember:

- Architecture, and how you thought your project through counts for twice as much as the code does.
- Be SOLID, and think before coding, and proceed step by step (Sprints)
- Ask yourselves “Why ?”
- The client is present during every session, communicate with him.

Subject 1 : Bank logs

The software must allow the management bank accounts operations, and keep a common log of all operations.
There must be at least 4 different accounts.
A file save system will also be implemented.

Subject 2 : Car race

The software must allow a simple race between multiple vehicles.
Vehicles move at random, but have the same probability of movement (ex : car :½ chance to move 2 spaces, bus: ⅓ chance to move 3 spaces, etc…)
The track implementation is free.

Subject 3 : Cell division

The software will modelize the growth of a population of cells.
Each cell has a chance of replicating via mitosis (1 cell gives 2 identical cells).
The population starts with one cell.
Environment factors will impact the replication rate (heat, nutrients, salinity…).

Subject 4 : Order and shipping

The software must allow the management of an order : Order/Billing /Shipping (Check stocks, file order form, Ship).
The important part is the functionality and the workflow, network behavior is NOT expected.

Subject 5 : Company Hierarchy

The software must allow the handling of all employees of a company.
There are 4 departments in this company : Accounting, Sales, Marketing, Manufacturing.
An employee can be CEO, or Head of its department, and thus have subordinates, or not, and thus have a supervisor.
The software should allow the listing of employees, the transfer of employees, and the cost of each department (sum of salaries)

Subject 6 : Auction House

The software must allow the selling of items at an auction house.
Bidders are automated, depending on their available resources, and take time to think if they will bid or not.
The auctioneer starts the bid and waits for the bidders to make an offer. If no offer is made after a certain time, the item is sold to the highest bidder.

Subject 7 : Vending Machine

The software must implement a Vending Machine. The machine can sell items, and be restocked.
It should take into account credit, stock, change.
The implementation of realistic change (only 2,1,0.5,0.20….) is not necessary

Subject 8 : Remote Control

The software must implement a remote control that works on a TV and a Radio.
The actions possible on every device are : ON, OFF, Volume Up, Volume Down, Previous Station, Next Station. The remote also has an ALL ON and ALL OFF button, as well as an Undo button, that negates the last order.

Subject 9 : Unclear Specifications

3 students were asked to deliver a collection of their favorite movies with a few infos (title, main actor, director, year of publication).
Each one of them coded a class returning a collection of MovieInfos, but one offered an ArrayList, the second an Array, and the third a HashTable.
Your software must allow the presentation of the movies in each collection, regardless of it’s type.
Step 1 : develop the first solution as each student did (3 different collections),
Step 2 : unify it all by keeping most of the student’s code (don’t change the collection types)