Skill Matrix

My goal is to create a basic prototype of the skill matrix in C#. This is an exercise in my C# for D2L interview, so I want to make sure I find reasons to use abstract classes, recursion. I’ll reproduce the concept of a database without using a database.

Some basic deliverables I have in mind:

* **Stores actual data (least amount work necessary to accomplish).**
* **Uses one abstract class for all my “table classes” to inherit from**
* **make the crappiest skill dependency I can, as quick as I can**
* That’s it, stop there. No more features allowed. The minimum tables allowed for this.
* Feature two: use recursion to retrieve skills (come up with some kind of skills dependency)
* Undecided: A validation process … hinders my practice abstract class goal… But I’ll imagine it will be in place for now and later.
* accept that this console-based approach is less usable.

out of scope but cool features

* an arbitrary skill size limited to 10ish, at which point they must appear under a sub category… sort of like a rebalancing of a hash table that executes automatically when a particular grouping exceeds a specified number. however the automatic selection of a parent name is unfortunately beyond my skill atm. out of scope.
* use auto complete as users type in a field. this would make this way more usable … well that’s a shiiiit realization for my C# goal... I suppose that’s way outta line now. so that’s an argument for doing this non-console, and probably it’ll be way easier to find code that does this for web than not. out of scope.
* mutually exclusive skills. fuck that noise, waaaay out of scope, waaaaay too difficult.
* shit, I wanna do a tagging system. basically the goal with this consideration is to have a shortcut to circumvent clicking / digging deep into a hierarchy to find a specific skill. ugh. it’s very pleasing an idea to consider, but I must put this out of scope. No Tagging.

So next what I’d like to focus on is resolving some questions I have. So right now I have the following hierarchy: Table > Skill > SkillHave, as well as Table > Person > Member. Now the functionality I want is to model after SQL, so I want any of these classes to be able to invoke Add(). Currently I’m supposing I can do this with **virtual** in Table only, and **override** in all other children classes. Is this true? I clearly want Add() to perform validation logic. … specify it and try it…

* Add() should invoke the ID of that class, so I want a static variable that’s overridden?? in each child?? And supposing override works, can I apply to multiple depths?
* Add(): Also I’m looking for a way for it to “validate locally”, i.e. in order to have a SkillHave, it necessitates having a Skill. I’m leaning towards the fact the code itself handles this, there’s no additional validation I can really implement.
* OK so clearly one requirement I have is a unique static ID at each class.
* I’m thinking I may want base.Add() or something, not sure yet a reason why.
* Pondering How To: I can inherit the fact it starts at 0… but that’s not enough to be valuable
* So far I see the class itself can be abstract, as well as the methods

I’m going to try listing below the kinds of tables I think would be useful. So far basically I’ll need a set of tables that are for data, a group that entirely would later be replaced by SQL later. Separate from this group would be operational classes that control the function of this program.

Skill

any string

level 0 - 5

desire 0 -5

Attempt at hierarchy

code

techniques

debugging

problem-solving

data structures

software (ide)

netbeans

eclipse

visual studio

2013

jetbrains

intellij idea

android studio

web storm

languages

C#

C++

Java, etc.

art

animation (2d)

character animation

animation (3d)

character animation

rigging

uv mapping & unwapping

texturing

style

pixel

cel shading

music

management

software (office)

Word

Excel

PowerPoint

Project

Visio

Google Drive Docs

team delegation