Implementáció - kiegészítő dokumentáció

Az általunk választott projektben egy ***Tamagocsi*** programot kellett megvalósítanunk. Az előzetesen leadott projektterven leírt tervek mind megvalósításra kerültek. A munka az ütemezésben megadottakhoz képest csúszott egy hetet, de határidőre elkészült. A csúszás egyetlen oka, hogy eredetileg 4 fővel indult csapat, majd a tervek elkészülte utánra 2 főre csökkent a csapattagok száma. A leadott projekttervben már a 2 főre előlátott munkamegosztás található.

Felhasznált Design Pattern-ekről részletesen:

- **Builder** - a *User*-hez és a *Pet*-hez készült Builder, melyek az adott objektum létrehozását egyszerűsítik meg. Például mikor egyes műveleteket végzünk és az adott műveletet végző függvény egy, az adott típusú objektumot vár, akkor megtehetjük, hogy létrehozunk egy adott típusú objektumot és amennyiben a függvény nem használja fel az összes paraméterét akkor megtehetjük, hogy csak a szükséges paramétereket állítjuk be, nem kell felesleges random értékeket beírni a többi helyre. Nem is beszélve róla, hogy mennyivel fejlesztő barátiabb módon lehet létrehozni és módosítani adott típusú objektumokat.

- **Decorator** - *Upgrade*Decorator-t tettünk a kedvencek által birtokolható fejlesztésekhez (*Upgrade*). Az alapmegjelenítésen csak a házi kedvenc szerepel és a további bővítésekkel kerülnek az *Upgrade* elemek a főpanelra. Adott elemek vásárlásakor vagy törlésekor a decorator végzi az elemek frissítését panelon.

- **Visitor** - amikor szeretnénk vásárolni a boltban, akkor a *Market* gombra kattintva kilistázódik az összes termék, az ételek (*Food*) és fejlesztések (*Update*) is. Ekkor lehetőségünk van a "Kosárba" pakolni termékeket. Egy *ItemVisitor*-t használtunk az így kiválasztott elemek árainak végig járásához és összeadásához, hogy aztán egy *TotalPrice* formájában megjeleníthessük a felhasználónak.

- **Singleton** - minden adat letárolása adatbázisban történik, így pl. a felhasználók, állatok, ételek, fejlesztések, munkák, állapotok, ... Egy *DAO* osztályban vannak megvalósítva az adatbázis műveleteket végző függvények. Maga a *DAO* osztály lett Singleton. Motivációként, a program futása során nem lehet megnyitva egynél több adatbázis *Connection*. Ha több példány is létrehozható lenne a *DAO*-ból, akkor ez igen nehezen lenne kivédhető, viszont a Singleton megoldja ezt a problémát.

- **Observer** - kedvencfigyelő felületen meg van jelenítve többek között a házi kedvenc és az, hogy éppen milyen "állapotban" van, értve ezalatt az öröm, az energia, a higiénia és az éhség szintjét. Ahogy telik az idő ezek az értékek folyamatosan csökkennek és dinamikusan frissítjük a *PetObserver* segítségével a megjelenített *progressBar*-okon az állapotot (%-ban) és frissíteni az adatbázisban is ezeket az értékeket (ezzel biztosítva a folyamatos mentést). Emellett az ételek elfogyasztásával illetve elvégzett munkákkal (munkának tekintjük pl. a fürdést, evést, játékot, sétát, stb) a felhasználó is tudja növelni az adott állapotok szintjét. Persze komolyabb munkák elvégzése esetén csökkenteni is.

Rövid leírás a program működéséről és útmutató a használatához:

Mikor elindítjuk a programot, egy bejelentkező felület fogad bennünket. Amennyiben már regisztráltunk nincs más dolgunk, mint a meglévő felhasználónév és jelszó párossal belépni. Ellenkező esetben átléphetünk a regisztrációs oldalra. Itt csupán egy felhasználónevet, jelszót, valamint egy email címet kell megadni, majd vissza lépni a beléptető oldalra.

Belépés után egy kedvencválasztó felület fogadja a felhasználót. Amennyiben vannak állatai, akkor választhat közülük, ellenkező esetben hozhat létre újat. Létrehozás esetén 4 féle állat közül választhat: kutya, macska, hal és pingvin, melyeknek vannak különböző típusai. Egy új név megadásával már létre is hozható a kedvenc és megkezdhető vele a játék. Ugyanezen az alakon van lehetőség a saját felhasználói adatok módosítására.

A kedvenc figyelő ablak felső részén található 4 folyamatsáv, melyek a kedvenc 4 fő állapotát jelenítik meg, dinamikusan frissülve. Ilyen állapot az energia, éhség, higiénia és öröm. Ezektől az értékektől függ a kedvenc aktuális kedve. A kedvét egy nagy gondolatbuborékban jelezzük, mely lehet boldog (zöld), kedvetlen (sárga) vagy mérges (piros). Abban az esetben, ha az összes állapotjellemző lecsökken 0%-ra, akkor a kedvencünk elhagy bennünket.

Alatta egy hasonló folyamatsávban az aktuálisan végzett munka található. Amíg a kedvenc munkát végez nem használható más műveletre (ki is szürkül a képe). Ezekkel a munkákkal tudja módosítani az állapotait, illetve pénzt keresni a vásárláshoz.

A jobb oldalon megtekinthetőek az állat által birtokolt termékek (ételek és fejlesztések). Megfelelőt kiválasztva módosulnak a kedvenc állapotadatai, vagy ideiglenesen vagy maradandóan. Így pl. ha elfogyaszt valamilyen ételt, az akkor csak egyszer lesz kihatással az állapotára. Viszont ha vesz egy nagyobb tálat, akkor ezután mindig amikor megeszik valamilyen ételt, akkor az jobban enyhíti majd az éhséget, mintha kisebb tálból enne. Vagy hasonló módon a hal esetén a nagyobb akvárium lassabban csökkenő örömérzetet jelent.

Admin - Admin felhasználónév - jelszó párossal tudunk belépi adminisztrátorként. Az adminisztrátornak joga van minden adat módosításához. Így módosíthatja a felhasználó és a kedvencek adatait, rendelhet hozzájuk munkát, avagy vehet el tőlük. Ugyan így ételt és fejlesztést is. Hozhat létre új ételeket, fejlesztéseket és munkákat, majd tetszése szerint módosíthatja őket. Megváltoztathatja a kedvenc állapotát, illetve állíthatja az általa birtokolt pénz mennyiségét. Az adminisztrátornak van joga módosítani az idő telését. Ez alatt azt értjük, hogy 1 valóéletbeli perc mennyi időnek feleljen meg a játékon belül.