Objektum elvű alkalmazások fejlesztése 3. beadandó

Berezvai Dániel http://elte.3ice.hu/

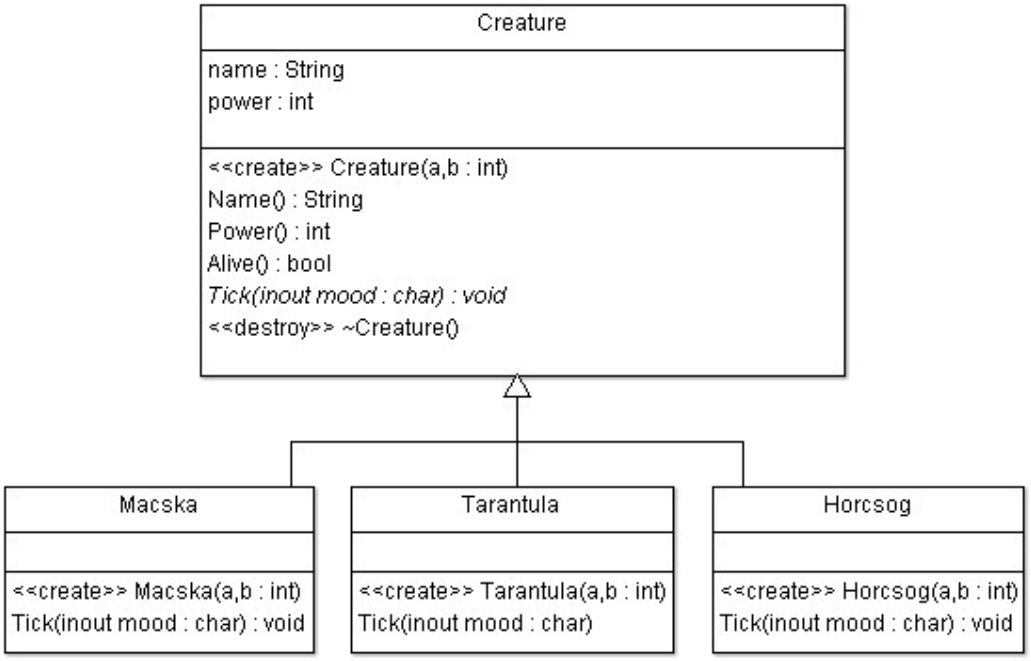
9. Feladat

```
3ICE: I was assigned the 9th task:
Az alábbi feladat megoldásához több olyan osztályt kell használni,amelyek
egy közös ősosztályból származnak és felüldefiniálják az ősosztály
virtuális metódusait. Ezen osztályok objektumait egy gyűjteménybe kell
elhelyezni, majd
ezt a gyűjteményt kell bejárni, a benne levő objektumok megfelelő
metódusait meghívni.
Készítsünk C++ programot a következő feladat megoldására!
Kati házi kedvencei a tarantulák, az arany hörcsögök és a macskák.
Az állatoknak az életerejük megőrzéséhez a táplálékon túl egyéb dolgokra is
szükségük van: a tarantuláknak száraz és meleg terráriumra; az arany
hörcsögöknek puha alomra, ahová befúrhatják magukat, a macskáknak
rendszeres simogatásra. Kati állatainak van neve és ismerhető az
életerejüket mutató 0 és 70 között szám (0 esetén az állat elpusztul).
Katinak vannak jobb és rosszabb napjai. Mikor vidám, egyik állatáról sem
feledkezik meg: ilyenkor a tarantulák életereje 1 - gyel, a hörcsögöké 2 -
vel, a macskáké 3 - mal nő. Átlagos napokon csak a macskáival foglalkozik
(életerejük 3 - mal nő), a többi állat életereje ilyenkor csökken: a
tarantuláké 2 - vel, a hörcsögöké 3 - mal. Amikor szomorú, csak a
legszükségesebb teendőket látja el és ezért minden állat egy kicsit
gyengébb lesz: a tarantulák 3 egységgel, a hörcsögök 5 - tel, a macskák 7 -
tel.
Az állatok adatait egy szöveges állományban találjuk. Az első sor
tartalmazza az állatok számát, amelyet külön - külön sorban az állatok
adatai követnek. Ebben egy karakter azonosítja az állat fajtáját (T –
tarantula, H – hörcsög, M – macska), amit szóköz után az állat neve követ,
majd újabb szóköz után a kezdeti életereje. Az állományban az állatok
felsorolását követő utolsó sorban egy betű sorozat (sztring) írja le Kati
kedvének az egymás utáni napokon való alakulása: v – vidám, a – átlagos, s
– szomorú. Feltehetjük, hogy a fájl formátuma helyes.
Szimulálja az állatok életkedvének változását Kati hangulatváltozásainak
nyomán és írja ki az állatok adatait minden nap végén!
```

Specifikáció

Az állatok leírásához bevezetünk négy osztályt. Attól függetlenül, hogy egy háziállat konkrétan kicsoda vagy mi a fajtája, számos közös tulajdonsággal rendelkeznek. Mindegyiknek van neve és életereje, meg lehet róla kérdezni, hogy hívják ($Name()$), él-e ($Alive()$) még, azaz az életereje nagyobb-e nullánál, mennyi életereje van ($Power()$) és szimulálni lehet annak változásait a napok múlásával. Ez utóbbi művelet ($\mathrm{Tick}()$) módosítja az állat életerejét a neki átadott háziállat gazdi hangulata alapján. Ennek a műveletnek a hatása attól függ, hogy egy állat milyen fajú, ezért ez a művelet az állatok általános jellemzésének szintjén még nem implementálható. Az általános háziállat típusát leíró `Creature` osztály absztrakt lesz, hiszen egyrészt a $Tick()$ metódusa is absztrakt, másrészt

úgysem akarunk ilyen objektumot létrehozni. Ebből származtatjuk a konkrét fajtájú állatok, `Macskák`, `Hörcsögök` és `Tarantulák` osztályait. A speciális osztályok konstruktorai meghívják az ősosztály konstruktorát, ami inicializálja az életerőt és a nevet.



1. ábra C:\ELTE\OAF\GY\3\Sim\Sim.uml, Sim.ClassDiagram.png, Sim.xml.zip, és Sim.zargo

Az $Alive()$, $Name()$, és $Power()$ metódusok az ősosztály szintjén implementálhatók, de a $Tick()$ csak a konkrét osztályok szintjén. Az $Alive()$ metódus akkor ad igaz értéket, ha az életerő pozitív. A $Name()$ metódus a név adattag értékét adja vissza. A $Power()$ metódus az életerő adattag értékét adja vissza. A származtatott osztályokban felüldefiniáljuk a $Tick()$ metódust, ami az alább definiált táblázatok alapján módosítja az életerőt.

Ez megkapja a gazdi hangulatát, és a táblázat megfelelő sora alapján megváltoztatja az aktuális háziállat életerejét.

Hangulat	Hörcsög	Macska	Tarantula
Vidám	+2	+3	+1
Átlagos	-3	-3	-2
Szomorú	-5	-7	-3

2. ábra A $Tick()$ művelet hatását a fenti táblázat foglalja össze.

Most pedig specifikálhatjuk a teljes feladatot. A specifikációban nem kell megkülönböztetni egy-egy háziállat feldolgozása utáni hangulat állapotokat.

Állapottér: $hangulatok: \mathbb{N}^m, háziállatok: Állat^n$

Előfeltétel: $állatok = állat \quad ' \wedge hangulatok = hangulatok'$

Utófeltétel: $állatok = állato \ ' \wedge hangulatok = hangulatok' \wedge$

$$életerők = \bigwedge_{\substack{i=1,j=1 \\ Eredm(állatok[i],hangulat[j])_1, \\ Alive()}}^{n} < állatok[i].név >$$

ahol az $Eredm: Állat \times \mathbb{N}^m \rightarrow Állat \times \mathbb{N}^m$ függvény kiszámolja, hogy egy lény az adott napon a gazdi hangulatától függően hogyan változik (életben marad-e) vagy hogyan változik az életereje. Ezt a számítást egy rekurzív definíciójú függvénnyel lehet leírni úgy, hogy egy kezdeti életerővel rendelkező lény és egy m hosszúságú időszakasz esetén $Eredm(állat, hangulat) = r(m)$, ahol

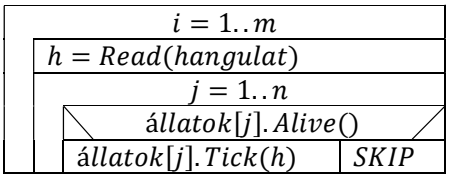
$r: [0..m] \rightarrow Állat \times \mathbb{N}^m \; r(0) = (lény, hangulat) \; \forall j \in [1..m] \; r(j)$
$= \begin{cases} Tick(r(j-1)) \; ha \; r(j-1)_1.Alive() \\ r(j-1) \; ha \; \neg r(j-1)\_1.Alive() \end{cases}$

A $Lépés(r(j-1))$ állítja elő a j-edik nap utáni $r(j)_1$ állatot (csak az életereje változik az előző naphoz képest). Ezeket a változásokat a korábban definiált $Tick()$ függvény alapján számíthatjuk ki. Ez ugyanis az aktuális $r(j-1)_2[j]$ hangulat állapot alapján megadja, hogy mennyivel változik az $r(j-1)_1$ állat életereje.

Absztrakt program

A megoldó programnak két szintje lesz. A felső szinten a fenti specifikációnak megfelelő algoritmus található, az alsó szinten a lények osztályai. struktogram

A külső ciklus a napokon iterál végig, és beépített cikluson keresztül implementálja a lények életerejének előállítását. A specifikáció hangulatfüggő függvényeinek kiszámolása történik, amely egy lépésben átalakítja az $állatok[j]$-t.



3. ábra Struktogram

A belső ciklus egyszerűen végigszalad az állatokon és frissíti az életerejüket az aznapi hangulattól függően. Egy állat életereje attól kezdve már nem változik tovább, ha a vizsgált lény életereje elfogyott.

Megvalósítás

Az absztrakt algoritmust a `main.cpp` állományban elhelyezett mai függvényben találjuk. Az osztályok definíciói a `sim.h` fejállományba, a $Tick()$ metódusok implementációi a `sim.cpp` forrásállományba kerülnek.

A specifikációban és a programkódban használt elnevezések kapcsolata az alábbi:

állat	creature
életerő	power
név	name
él	alive

hangulat	mood
eredmény/lépés	tick

A főprogram

`C:\ELTE\OAF\GY\3\Sim\main.cpp`

Szükséges könyvtárak és header fájlok importálása és névtér deklaráció:

```
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include "Sim.h"
using namespace std;
```

Konstansok az olvashatóságért:

```
const char tarantula = 'T';    const char vidam = 'v'; //T+=1,H+=2,M+=3
const char horcsog = 'H';      const char atlagos = 'a'; //M+=3,T-=2,H-=3
const char macska = 'M';       const char szomoru = 's'; //T-=3,H-=5,M-=7
```

A $main$ függvény az absztrakt program kódján kívül az állatok és a hangulatok beolvasását is tartalmazza.

A szimulációban részt vevő háziállatokra történő hivatkozásokat, azaz $Creature*$ típusú elemeket egy tömbben ($vector<Creature*>$) tároljuk. Így lényegében egy olyan tömböt használunk, amelyik vegyesen tárolhat különböző, de a $Creature$ osztályból származtatott osztályú (típusú) elemeket: a tömb egy elemének tehát alternatív szerkezetű típusa van.

```
int main(){
	cout << "================================" << endl;
	cout << "3ICE's Simulation of animal care" << endl;
	cout << "================================" << endl << endl;

	ifstream f("input.txt");
	int n; f >> n;
```

```
	vector<Creature*> creatures(n);
	for (int i = 0; i < n; ++i){
		char type; string name; int health; f >> type >> name >> health;
		switch (type){
		case tarantula: creatures[i] = new Tarantula(name, health); break;
		case horcsog: creatures[i] = new Horcsog(name, health); break;
		case macska: creatures[i] = new Macska(name, health); break;
		default: cout << "Hibas bementet! " << type << " " << name <<
			" Pedig megigerted, hogy a fajl formatuma helyes."; return -1;
		}
		//3ICE: cout << i << type << name << health << endl;
	}

	for (int i = 0; i < n; i++){
		Creature* c = creatures[i];
		if (!c->Alive()) cout << c->Name() << " mar holtan erkezett. :(" << endl;
		else cout << c->Name() << " kezdeti eletereje: " << c->Power() << endl;
	} cout << endl;
```

A hangulatok egy karakterek vektora. UPDATE: Mégsem. Nincs szükség tárolásra.

A feldolgozás kódja némileg átalakult a tervhez képest, de azzal ekvivalens hatású.

```
	int m; f >> m; char mood;
	for (int i = 1; i <= m; ++i){//3ICE: int i = 0; i < m; ugyan azt produkalna
		f >> mood; switch (mood){
		case vidam: cout << i << ". nap: Kati vidam." << endl; break;
		case atlagos: cout << i << ". nap: Atlagos nap." << endl; break;
		case szomoru: cout << i << ". nap: Kati szomoru." << endl; break;
		default: cout << "Hibas bementet! " << mood <<
			" Pedig megigerted, hogy a fajl formatuma helyes."; return -1;
		}
		for (int j = 0; j < n; j++){
			Creature* c = creatures[j];
			if (c->Alive()){
				c->Tick(mood);
				if (!c->Alive()) cout << c->Name() << " meghalt. :(" << endl;
				else cout << c->Name() << " uj eletereje: " << c->Power() << enc
			}
		} cout << endl;
	}

	cout << "Vege a szimulacionak." << endl;
	for (int i = 0; i < n; i++){
		Creature* c = creatures[i];
		if (!c->Alive()) cout << c->Name() << " nem elte tul... :(" << endl;
		else cout << c->Name() << " tulelte Katit. Vegso eletereje: " << c->Power()
	} cout << endl;
```

A program végén felszabadítjuk a saját memóriafoglalásainkat.

```
	for (int i = 0; i < n; ++i) delete creatures[i];

	//cin >> n;
	cout << "Goodbye!" << endl;
	return 0;
}
```

Creature osztály

C:\ELTE\OAF\GY\3\Sim\Sim.h

Őr (header guard) és egy import:

```
#ifndef Sim_H
#define Sim_H
#include <iostream>
```

Ősosztály:

```
class Creature{
protected:
	std::string name; int power;
	Creature(std::string a, int b) : name(a), power(b) {}
public:
	std::string Name() const { return name; }
	int Power() const { return power; }
	bool Alive() const { return power > 0; }
	virtual void Tick(char &mood) = 0;
	virtual ~Creature() {}
};
```

Speciális háziállatok osztályai:

```
class Horcsog : public Creature{
public:
	Horcsog(std::string a, int b) : Creature(a, b){}
	void Tick(char &mood);
};
class Macska : public Creature{
public:
	Macska(std::string a, int b) : Creature(a, b){}
	void Tick(char &mood);
};
class Tarantula : public Creature{
public:
	Tarantula(std::string a, int b) : Creature(a, b){}
	void Tick(char &mood);
};
#endif // Sim_H
```

A *Tick()* metódus felüldefiniálásai:

C:\ELTE\OAF\GY\3\Sim\Sim.cpp

```
#include "Sim.h"
#include <iomanip>
using namespace std;

const char vidam = 'v'; //T+=1,H+=2,M+=3
const char atlagos = 'a';//M+=3,T-=2,H-=3
const char szomoru = 's';//T-=3,H-=5,M-=7

void Horcsog::Tick(char &mood){
	switch (mood){
	case vidam: power += 2; break;
	case atlagos: power -= 3; break;
	case szomoru: power -= 5; break;
	}
}

void Macska::Tick(char &mood){
```

```
	switch (mood){
	case vidam: power += 3; break;
	case atlagos: power -= 3; break;
	case szomoru: power -= 7; break;
	}
}

void Tarantula::Tick(char &mood){
	switch (mood){
	case vidam: power += 1; break;
	case atlagos: power -= 2; break;
	case szomoru: power -= 3; break;
	}
}
```

Egy példa bemenet (input.txt):

```
8
H Horcsog 49
T Tarantula 57
M Macska 42
H Pofi 14
T Pok 7
M Blacky 38
M Pamacska 16
M Falatka 11
6
vasss
```

Példa futás a fenti bemenetre

```
================================
3ICE's Simulation of animal care
================================

Horcsog kezdeti eletereje: 49
Tarantula kezdeti eletereje: 57
Macska kezdeti eletereje: 42
Pofi kezdeti eletereje: 14
Pok kezdeti eletereje: 7
Blacky kezdeti eletereje: 38
Pamacska kezdeti eletereje: 16
Falatka kezdeti eletereje: 11

1. nap: Kati vidam.
Horcsog uj eletereje: 51
Tarantula uj eletereje: 58
Macska uj eletereje: 45
Pofi uj eletereje: 16
Pok uj eletereje: 8
Blacky uj eletereje: 41
Pamacska uj eletereje: 19
Falatka uj eletereje: 14

[…]

4. nap: Kati szomoru.
Horcsog uj eletereje: 38
Tarantula uj eletereje: 50
```

```
Macska uj eletereje: 28
Pofi uj eletereje: 3
Pok meghalt. :(
Blacky uj eletereje: 24
Pamacska uj eletereje: 2
Falatka meghalt. :(

5. nap: Kati szomoru.
Horcsog uj eletereje: 33
Tarantula uj eletereje: 47
Macska uj eletereje: 21
Pofi meghalt. :(
Blacky uj eletereje: 17
Pamacska meghalt. :(

[…]

Vege a szimulacionak.
Horcsog tulelte Katit. Vegso eletereje: 28
Tarantula tulelte Katit. Vegso eletereje: 44
Macska tulelte Katit. Vegso eletereje: 14
Pofi nem elte tul... :(
Pok nem elte tul... :(
Blacky tulelte Katit. Vegso eletereje: 10
Pamacska nem elte tul... :(
Falatka nem elte tul... :(

Goodbye!

Process returned 0 (0x0)   execution time : 0.00149 s
Press any key to continue.
```

Tesztelési terv

Érvényes fekete doboz tesztesetek:

1. Nincsenek háziállatok.
2. Nulla hosszúságú a szimuláció (minden állat változatlan marad).
3. Egy speciális állat kipróbálása (mindhárom fajtára külön-külön) olyan szimulációban, ahol egymás után mindhárom fajta hangulat előfordul, és ezeken az állat életben marad.
4. Egy speciális állat kipróbálása (mindhárom fajtára külön-külön) olyan szimulációban, ahol a lény életereje elfogy.
5. Külön vizsgálva azt, amikor az utolsó napra érve fogy el egy lény életereje.
6. Hosszú szimuláció: Általános eset sok nappal.
7. Állatkert: Általános eset sok háziállattal.

Érvénytelen adatokra is fel van készítve a fenti program. Nem létező állomány esetén a program ugyan elromlik, de hibás formátumú állomány esetén segítőkész és hasznos

hibaüzenetet dob: `Hibas bementet! N Nyuszi Pedig megigerted, hogy a fajl formatuma helyes.`

Fehér doboz tesztesetek: A fenti esetek tesztelik a program minden utasítását. Dinamikus helyfoglalások miatt viszont tesztelni kellett még a memóriaszivárgást is. (Nincs veszteség.)

A tesztelésnek részét képezik a modulonkénti tesztek (unit tesztek), amelyek egy-egy osztály szolgáltatásait vizsgálják.

Tesztfájlok:

`C:\ELTE\OAF\GY\3\Sim\1nincsenekallatok.txt, 2nullanaphosszuszimulacio.txt, 3specialis+4simateszt.txt, 5utolsonaphalmeg+6hosszu.txt, 7allatkert.txt, 8ervenytelenhaziallat.txt, 9ervenytelenhangulat.txt`

Random tesztfájl generátor `C:\ELTE\OAF\GY\3\Sim\Sim.xlsx` – Excel:

```
Típus =IF(C2<46.666;"H";IF(C2<23.333;"M";"T"))
Név =CONCATENATE(A2;ROW(A2))
Életerő =ROUND(RAND()*70;0)
```

Type	Name	Seed
H	H2	29
H	H3	43
T	T4	63
T	T5	57

Az 1000 soros 7állatkert.txt tesztesethez.

A projekt háromféleképpen is működik:

1) Visual Studio 2010 (→Sim.**sln**)
2) CodeBlocks 12 (→Sim.**cbp**)
3) Parancssor Linux (**g++** -o Sim Sim.cpp main.cpp)
   vagy Windows alatt
   (**gcc** -o Sim Sim.cpp main.cpp **-lstdc++** -enable-auto-import)

Dokumentáció vége. Bónusz: A Verseny feladat megoldása hasonló.

```
    int m;
    f >> m;
    vector<int> field(m);
    for(int j=0; j<m; ++j) f >> field[j];
    for(int i=0; i<n; ++i){
        bool l = true;
        for(int j=0; j<m; ++j){
            if (l = creatures[i]->Alive()) creatures[i]->
Transmute(field[j]);
            else break;
        }
        if (l) cout << creatures[i]->Name() << endl;}
```