Приклад використання класів помилок

// ConsoleApplication21.cpp : Defines the entry point for the console application.

//

#include "stdafx.h"

#include <iostream>

#include<string.h>

using namespace std;

//Примітивний клас тваринки, яку поміщаємо у комірки масиву(у одній комірці не може бути двох тваринок)

class Animal

{

public:

int position;

int type;

Animal(int position, int type) :position(position), type(type) {}

};

//Клас помилки (містить детальну інформацію про помилку, що сталася при розміщенні у одну комірку двох тваринок)

class PutException

{

public:

int position;

Animal \* creature;

string message;

PutException(int position, Animal \* creature) :position(position), creature(creature) {

message = "cell is occupied";

}

};

//Клас масив, у якому розміщуються тваринки (по одну у одну комірку)

class Array {

public:

Animal \*\* data;

int length;

Array(int length)

{

data = new Animal \*[length];

for (int i = 0; i < length; i++)

{

data[i] = 0;

}

this->length = length;

}

void putAnimal(int index, Animal\* animal)

{

if (data[index]==0)

{

data[index] = animal;

}

else

{ //Якщо комірка зайнята, то генеруємо виключну ситуацію (об’єкт класу

// що містить повну інформацію про те, що сталося)

throw new PutException(index, animal);

}

}

};

int main()

{

Array arr(5);

arr.putAnimal(1, new Animal(1, 5));

try

{

arr.putAnimal(1, new Animal(1, 7));

}

catch (PutException\* e) //Перехоплюємо виключні ситуації класу PutException

//

{

cout <<"message: "<< e->message.data() << endl;

cout << "position: " << e->position << endl;

cout << "animal type :" << e->creature->type << endl;

arr.putAnimal(2, e->creature);

}

for (int i = 0; i < arr.length; i++)

{

try {

if (arr.data[i] == 0)

throw "oj";

cout<<arr.data[i]->type<<endl;

}

catch (...)

{

cout << "empty"<<endl;

}

}

system("pause");

return 0;

}

////////////////////////////////////////////////////////////////////////////////////

Приклад з двома класами помилок

// ConsoleApplication21.cpp : Defines the entry point for the console application.

//

#include "stdafx.h"

#include <iostream>

#include<string.h>

using namespace std;

//-----------------------------------------------------------------

class Animal

{

public:

int position;

int type;

Animal(int position, int type) :position(position), type(type) {}

};

class PutException

{

public:

int position;

Animal \* creature;

string message;

PutException(int position, Animal \* creature) :position(position), creature(creature) {

message = "cell is occupied";

}

};

//Клас помилки, яка виникає при звертанні за межі масиву

class OutOfRangeException

{

public :

int wrongIndex;

int nearestRightIndex;

string message;

Animal \* creature;

OutOfRangeException(int wrongIndex, int nearestRightIndex, string message, Animal \* creature) :

wrongIndex(wrongIndex), nearestRightIndex(nearestRightIndex), message(message), creature(creature) {}

};

class Array {

public:

Animal \*\* data;

int length;

Array(int length)

{

data = new Animal \*[length];

for (int i = 0; i < length; i++)

{

data[i] = 0;

}

this->length = length;

}

void putAnimal(int index, Animal\* animal)

{

if (index < 0)

throw new OutOfRangeException(index, 0, "out left", animal);

if (index >=length)

throw new OutOfRangeException(index, length-1, "out right", animal);

if (data[index]==0)

{

data[index] = animal;

}

else

{

throw new PutException(index, animal);

}

}

};

int main()

{

Array arr(5);

arr.putAnimal(1, new Animal(1, 5));

try

{

arr.putAnimal(1, new Animal(1, 7));

}

catch (PutException\* e) //Перехоплення помилки PutException

{

cout <<"message: "<< e->message.data() << endl;

cout << "position: " << e->position << endl;

cout << "animal type :" << e->creature->type << endl;

arr.putAnimal(2, e->creature);

}

catch (OutOfRangeException\* e) //Перехомплення помилки OutOfRangeException

{

cout << e->message.data() << endl;

arr.putAnimal(e->nearestRightIndex, e->creature);

}

for (int i = 0; i < arr.length; i++)

{

try {

if (arr.data[i] == 0)

throw "oj";

cout<<arr.data[i]->type<<endl;

}

catch (...)

{

cout << "empty"<<endl;

}

}

system("pause");

return 0;

}

/////////////////////////////////////////////////////////////////////////////

Приклад шаблону

// ConsoleApplication21.cpp : Defines the entry point for the console application.

//

#include "stdafx.h"

#include <iostream>

#include<string.h>

using namespace std;

//----------- pattern -----------------

class SingleTone

{

private:

int a;

SingleTone() {

cout << "constructor" << endl;

a = rand() % 100;

}

static SingleTone\* obj;

public:

int getA()

{

return a;

}

static SingleTone\* getObject() {

if (obj == 0)

obj = new SingleTone();

return obj;

}

};

SingleTone\* SingleTone::obj=0;

enum Days {

Monday = 1,

Thuesday = 2

};

enum Roles {

Admin=0,

Manager=1,

User=2

};

int main()

{

SingleTone\* s1=SingleTone::getObject();

cout << "s1->a="<<s1->getA() << endl;

SingleTone\* s2 = SingleTone::getObject();

cout << "s2->a=" << s2->getA() << endl;

SingleTone\* s3 = SingleTone::getObject();

cout << "s3->a=" << s3->getA() << endl;

//----------------------

Roles rol;

switch (rol)

{

case Admin:cout << "Hello admin" << endl;

break;

case Manager:cout << "Hello man" << endl;

break;

}

//Визначення дня з використання номера (не зовсім зрозуміло)

int day;

if (day == 1)

{

cout << "go to work" << endl;

}

//Визначення для з використанням типу enum

Days r;

if(r==Monday)

{

cout << "go to work" << endl;

}

system("pause");

return 0;

}