Oblikovni obrasci u programiranju

1. labos – zadaci sa zavodske stranice

ak. god. 2007/08

1)

Prevedi i isprobaj program zadan na stranici predavanja "OMT vs C++".

Promijeni redak:

Derived d;

u

Derived2 d;

Dodaj još i:

#include "derived2.hpp"

Pokreni program.

Implementiraj novu izvedenu klasu, Derived3.

Sučelje stavi u datoteku "derived3.hpp",

implementacijsku datoteku ne moraš imati

(sve metode mogu biti "umetnute" (inline)).

Sada uključi derived3.hpp u main,

te reci "Derived3 d;"

Pokreni program.

Stara implementacija klijenta

radi s novom izvedbom osnovne klase.

Klijent nismo morali ni pipnuti

(čak ni prevesti!).

Koje načelo kaže da \*stari kod\*

treba moći koristiti \*novi kod\*?

2)

Prevedi i isprobaj program zadan na stranici predavanja "generici".

Pokaži da isti mymax radi i za parametre tipa std::string!

Koje načelo kaže da \*stari kod\* treba moći koristiti \*novi kod\*?

3)

Prevedi i isprobaj program:

#include <vector>

#include <list>

#include <algorithm>

#include <iostream>

int main(){

std::vector<int> v(2);

v[0] = 7;

v[1] = v[0] + 3;

v.push\_back(5);

v.insert(v.begin(),1);

std::reverse(v.begin(), v.end());

for (int i = 0; i < v.size(); ++i)

std::cout << "v[" << i << "] = " << v[i] <<"\n";

}

Pokaži da isti vector radi i za parametre tipa std::string!

Koje načelo kaže da \*stari kod\* treba moći koristiti \*novi kod\*?

4)

Prevedi i isprobaj priloženi dopunjeni program s predavanja, stranica "proceduralni OCP?".

Dodaj metodu moveShapes(). Isprobaj.

Dodaj razred Rhomb.

Dodaj jedan Rhomb u listu objekata u main().

Sjeti se, sad moramo promijeniti i drawShapes().

Ovo je domino efekt (krutost), kojeg ćemo kasnije pokušati zauzdati. Za šalu, zaboravi adekvatno promijeniti moveShapes().

Isprobaj ponovo. Sad bi moveShapes trebao "puknuti". To je krhkost uzrokovana redundancijom. Ni to ne želimo imati u programu. Konačno, implementiraj rješenje s predavanja, i komentiraj njegovu funkciju.

#include <iostream>

#include <assert.h>

struct Point{

int x; int y;

};

struct Shape{

enum EType {circle, square};

EType type\_;

};

struct Circle{

Shape::EType type\_;

double radius\_;

Point center\_;

};

struct Square{

Shape::EType type\_;

double radius\_;

Point center\_;

};

void drawSquare(struct Square\*){

std::cerr <<"in drawSquare\n";

}

void drawCircle(struct Circle\*){

std::cerr <<"in drawSquare\n";

}

void drawShapes(Shape\*\* list, int n){

for (int i=0; i<n; ++i){

struct Shape\* s = list[i];

switch (s->type\_){

case Shape::square:

drawSquare((struct Square\*)s);

break;

case Shape::circle:

drawCircle((struct Circle\*)s);

break;

default:

assert(0);

exit(0);

}

}

}

int main(){

Shape\* list[4];

list[0]=(Shape\*)new Circle;

list[0]->type\_=Shape::circle;

list[1]=(Shape\*)new Square;

list[1]->type\_=Shape::square;

list[2]=(Shape\*)new Square;

list[2]->type\_=Shape::square;

list[3]=(Shape\*)new Circle;

list[3]->type\_=Shape::circle;

drawShapes(list, 4);

}

5)

Prevedi i isprobaj program s predavanja, str "OCP - GP". Pokušaj pisati izvan granica niza, u slučajevima da je prevođenje izvršeno s optimiranjem odnosno bez njega.

6)

Primijeni program sa stranice "OCP -- STL map" na ovim uputstvima. Koja riječ je najčešća?

7)

Prevedi i pokreni program na stranici LSP - primjer 2. Dodaj kod za hvatanje iznimke. Kako iznimke pospješuju čitljivost programa?

8)

Prevedi i pokreni program na stranici DI. Objasni načela inverzije i injekcije ovisnosti.

9)

Napišite funkciju za dijalog s korisnikom, kojom korisnik preko standardnog ulaza unosi listu brojeva odvojenih razmacima. Neka je duljina liste proizvoljna, odnosno neka kraj unosa označava znak za novi red '\n'. U prvoj verziji, koristite sljedeću funkciju standardne biblioteke C-a.

char \*gets(char \*s);

Isječak iz dokumentacije:

gets() reads a line from stdin into the buffer

pointed to by s until either a terminating newline or EOF,

which it replaces with '\0'.

Komentirajte lakoću održavanja funkcije, s obzirom na očekivano produžavanje

elemenata liste u skoroj budućnosti. Koje načelo je prekršeno?

Pokušajte poboljšati nadogradivost funkcije korištenjem sljedećih standardnih funkcija:

\* char \*fgets(char \*s, int size, FILE \*stream);

\* istream& getline ( istream& is, string& str );

Komentirajte utjecaj sučelja bibliotečnih funkcija na lakoću održavanja programa.

10)

Upoznaj se s tehnikom neprozirnih tipova.

http://en.wikipedia.org/wiki/Opaque\_pointer

Kako ta tehnika pospješuje načelo zatvorenosti?

Neka su zadane sljedeće komponente

ProzorRoditelj i ProzorDijete:

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

// MainWindow.hpp

#include "ChildWindow.hpp"

class MainWindow{

std::vector<ChildWindow> wnds\_;

public:

MainWindow();

// ...

public:

void addChild();

void numChildren();

};

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

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

// ChildWindow.hpp

#include "MainWindow.hpp"

class ChildWindow{

MainWindow\* parent\_;

public:

ChildWindow(MainWindow\* p):parent\_(p){}

public:

// ...

public:

int numberOfSiblings();

};

Nastao je problem cirkularne ovisnosti jer

ProzorDijete treba znati neke podatke

o ostalim objektima istog roditelja

(npr, koliko ih ukupno ima).

Cirkularna ovisnost je vrlo neželjena pojava

u programskoj arhitekturi jer

onemogućava testiranje komponenti u izolaciji.

Riješi problem cirkularne ovisnosti

tako da pokazivač na roditelja

u razredu ProzorDijete izvedeš

tehnikom neprozirnog objekta.