*// Transfer technologii 2011 - nagłówek implementacji modelu 24-10-2011*

*////////////////////////////////////////////////////////////////////////////////*

*// Klasy do kolejnej startowej wersji modelu (dla danych na poziomie wersji 1.61)*

*////////////////////////////////////////////////////////////////////////////////*

#ifndef \_opiKlasy\_plik\_naglowkowy\_hpp\_

#define \_opiKlasy\_plik\_naglowkowy\_hpp\_

#include "spsModel.h"

#include "spsGenNode.h"

#include "spsGenProc.h"

#include "spsGenLink.h"

#include "spsParaLink.h"

#include "spsGenInfo.h"

#include "spsMatrixNode.h"

#include "INCLUDE/platform.hpp"

#include "INCLUDE/wb\_ptr.hpp"

#include "INCLUDE/wb\_bits.h"

**using** **namespace** wbrtm;

#include <stdlib.h>

#include <cassert>

#include <string>

#include <iostream>

**using** **namespace** std;

**extern** **float** TOWARZYSKA\_NAJMNIEJSZA\_WAGA;*//=0.01;// dla bool KontaktTowarzyski::Poprawny();*

**extern** **float** JEDNOSTKOWA\_WYPLATA;*//=100;//Wartość najmniejszej wpłaty/wypłaty. Sposób użycia niejasny :-) Kasa za jeden raport?*

**extern** **float** WAGA\_NA\_DZIENDOBRY;*//=0.1;//Współczynnik zmniejszania wagi przy pierwszym kontakcie - nie może być za duża*

**extern** **float** TEMPO\_SPADKU\_LINKU;*// np. =0.01;//O jaką część link socjalny zanika przy nie używaniu*

**extern** **float** TEMPO\_WZROSTU\_LINKU;*// np. =0.1;//O jaką część link socjalny rośnie przy odpowiedzi*

**extern** **char**\* **const** KONTO;*//="Ruch konta"; //Marker komunikatu finansowego z przepływem*

**extern** **char**\* **const** DZIENDOBRY;*//="Dzien dobry";//Marker otwierającego komunikatu społecznego*

*//Klasa wewnętrzna skupiająca wspólne właściwości WĘZŁÓW z pracownikami (zespołów)*

*//////////////////////////////////////////////////////////////////////////////////*

**class** \_ZespolRoboczy:**public** GenerycznyWezelSieci

{

**protected**:

**friend** **class** ProcesProdukcyjny;

**friend** **class** ProcesRaportowany;

**friend** **class** ProcesBadawczy;

**friend** **class** ProcesSpoleczny;

*//Ile Etatów Efektywnosc Aktywność Centralnosc Probiznesowosc Prolegalnosc Czasochlonnosc Innowacyjnośc Finans. swobodne Finanse zaplan Dług do nadrz. Udzial w zysku TT Eksperckość w dziedzinie Uwagi*

**unsigned** IleEtatow;*//Ile etatów. Co najmniej 1, 0 - "duch" zespołu - plan utworzenia*

**float** Efektywnosc;*//0..2. Czy działa na 50% wydajności czy na 150%. Zależy od motywacji*

**float** Doswiadczenie;*//POZIOM EKSPERTYZY? Kolejny mnożnik do liczby etatów*

**float** Centralnosc;*//0..1. Jak bardzo wydajność i ewentualnie aktywność zależy od szefa/szefów*

**float** Proaktywnosc;*//Aktywność? 0..1. Podtrzymywanie aktywności zewnętrznej, tworzenie linków itp.*

**float** Czasochlonnosc;*//Jaki jest współczynnik czasochłonności typowej działalności. Wygenerowania pojedynczego wyniku badań?*

**float** Prolegalnosc;*//????Uczciwość??? Jak łatwo podjąć działania "uproszczone"*

*//Procedury podzialu czasu pracy na poszczególne procesy - do wywoływania w procedurach ChwilaDlaCiebie()*

**void** \_ZarzadcaProcesowLosowy(); *//Wybiera proces do wykonania losowo angażując się też losowo*

**void** \_ZarzadcaProcesowLeniwy(); *//Robi tylko najpilniejszy proces albo wcale*

**void** \_ZarzadcaProcesowFantazyjny();*//Robi pilny i jak mu coś zostaje sił to jeszcze jakiś*

**void** \_ZarzadcaProcesowFinezyjny();*//Sprytnie dzieli czas żeby wszystko szlo do przodu*

**void** \_ZarzadcaProcesowSprawiedliwy();*//Dzieli wszystkie siły proporcjonalnie do priorytetów procesów*

**public**:

**virtual** **bool** Poprawny();*//Sprawdza czy wszystkie wskaźniki są poprawnie wpisane (nie przypadkowe i nie 0)*

*//Żeby mogl rozliczac się finansowo z procesami.*

**virtual** **void** PrzyjmijKase(**float** Suma)=0;*//Suma może być ujemna. Może tą kasę "przepuścic" ale proces jest OK*

**virtual** **bool** StacGoNaWydatek(**float** Suma)=0;*//Sprawdza czy taki wydatek jest dopuszczalny*

};

*//Klasa dla badacza, zespołu badawczego*

*////////////////////////////////////////////////////////////////////////////////*

**class** JednostkaBadawcza:**public** \_ZespolRoboczy

{

**friend** **class** ProcesBadawczy; *//Musi mieć dostęp do właściwości, w celach kalkulacyjnych*

*//Implementacja tego co musi być wg. interfejsu typu podstawowego węzła*

**static** KonstruktorElementowModelu<JednostkaBadawcza> WirtKonstr;

ElementModelu::WirtualnyKonstruktor\* VKonstruktor() { **return** &WirtKonstr;}

**public**:

**void** ChwilaDlaCiebie(); *//Endogenne zmiany stanów, zarządzanie procesami itp.*

**void** InterpretujKomunikat(Komunikat\* Co);*//Przyjmowanie komunikatów*

JednostkaBadawcza(){}*//Konstruktor domyślny*

**bool** ZrobWgListy(**const** std::string\* Lista,**unsigned** Ile,**unsigned**& Blad);

**void** AktualizujListeDanych();*//Do zapisu i wyświetlania inspekcyjnego*

*//Ruchy na koncie*

**void** PrzyjmijKase(**float** Suma) {FinanseSwobodne+=Suma;} *//Żeby mogl rozliczac się finansowo z procesami*

**bool** StacGoNaWydatek(**float** Suma){ **return** Suma<FinanseSwobodne;}*//Sprawdza czy taki wydatek jest dopuszczalny*

**protected**:*//Właściwości obiektów klasy i metody pomocnicze*

*//Cechy:*

*//////////////////////////////////*

*// z klasy "ZespolRoboczy":*

*//Ile Etatów Efektywnosc Aktywność Centralnosc Probiznesowosc Prolegalnosc Czasochlonnosc Innowacyjnośc Finans. swobodne Finanse zaplan Dług do nadrz. Udzial w zysku TT Eksperckość w dziedzinie Uwagi*

*// inne - własne:*

*//Jedna skala czy dwie Pronaukowosc vs. Probiznesowsc*

**float** Probiznesowosc;*//0 - czysta nauka, 0.5-badania patentowalne/licenjonowalne, 1 - czysty biznes - szybkie wdrażanie*

**float** Innowacyjnosc;*//0..1. Łatwość startowania projektów bez inspiracji z zewnątrz, zaciekawienie projektami nieco odległymi od dotychczasowych*

**float** FinanseSwobodne;*//Rezerwy finansowe - w skali roku LUB UPROSZCZONA "ZAMOŻNOŚĆ"*

**float** FinanseObiecane;*//Zaplanowane do wydania, ale być może jeszcze nie na koncie (odnawiane/odbierane co 365 kroków)*

**float** Dlugi;*//Ile zespół "wisi" swojemu wydziałowi/instytutowi. Możliwe tylko chwilowo (30 kroków?)*

**float** UdzialWZysku;*//0..1 Jaki procent zysku z innowacji pobiera (reszta idzie dla pracowników)*

*//JAK IMPLEMENTOWAĆ MOTYWACJE????????? --> Zmiany efektywności, ale jaka metoda?*

*//Projekty z listy aktywnych projektów*

*//Kontakty z siatki*

*//Co z kontaktami międzynarodowymi?*

*//połączenie z obiektem gridowym a la rynek?*

};

*//Klasa dla Instytutu, Wydziału, Uniwersytetu itp.*

*////////////////////////////////////////////////////////////////////////////////*

**class** AdministracjaNaukowa:**public** \_ZespolRoboczy

{ *//Implementacja tego co musi być wg. interfejsu typu podstawowego węzła*

**static** KonstruktorElementowModelu<AdministracjaNaukowa> WirtKonstr;

ElementModelu::WirtualnyKonstruktor\* VKonstruktor() { **return** &WirtKonstr;}

**public**:

**void** ChwilaDlaCiebie(); *//Endogenne zmiany stanów, zarządzanie procesami itp.*

**void** InterpretujKomunikat(Komunikat\* Co);*//Przyjmowanie komunikatów*

AdministracjaNaukowa(){}*//Konstruktor domyslny*

**bool** ZrobWgListy(**const** std::string\* Lista,**unsigned** Ile,**unsigned**& Blad);

**void** AktualizujListeDanych();*//Do zapisu i wyświetlania inspekcyjnego*

*//Ruchy na koncie*

**void** PrzyjmijKase(**float** Suma) {FinanseSwobodne+=Suma;} *//Żeby mógł rozliczać się finansowo z procesami*

**bool** StacGoNaWydatek(**float** Suma){ **return** Suma<FinanseSwobodne;}*//Sprawdza czy taki wydatek jest dopuszczalny*

**protected**:*//Właściwości obiektów klasy i metody pomocnicze*

*///////////////////////////////////////////////////////////////////////////////*

*// z KLASY: ZespolRoboczy*

*// unsigned IleEtatow;//Ile etatów. Co najmniej 1, 0 - "duch" zespołu - plan utworzenia*

*//float Efektywnosc;//0..2. Czy działa na 50% wydajności czy na 150%. Zależy od motywacji*

*//float Proaktywnosc;//(Pro)aktywność. Prawdopodobieństwo organizowania konferencji i innych eventów umożliwiających kontakty między naukowcami i gośćmi (czyli co ile kroków następuje taki event)*

*//float Centralnosc;//0..1. Jak bardzo aktywność (zewnętrzna?) zależy od szefa/szefów*

*//float Doswiadczenie;//POZIOM EKSPERTYZY? Jak tego używać? Może to samo co efektywność???*

*//float Czasochlonnosc;// Ile czasu (kroków modelu) się czeka z podjęciem decyzji*

*//float Prolegalnosc;//????Elastyczność??? Jak łatwo podjąć działania obchodzące "bzdurne" przepisy*

**float** Probiznesowosc;*//0...1 Prawdopodobieństwo podjęcia inicjatyw z współudziałem biznesu*

**float** ProInwest;*//0-1 Skłonność inwestowania wolnych środków w rozwój związany z TT*

**float** FinanseSwobodne;*//Rezerwy finansowe - w skali roku (odnawiane/odbierane co 365 kroków) LUB UPROSZCZONA "ZAMOŻNOŚĆ"*

**float** FinanseObiecane;*//Zaplanowane do wydania, ale być może jeszcze nie na koncie*

**float** Dlugi;*//Ile jednostka już "wisi". Np. na budowę, albo na prąd jak dotacja zaległa. Nie może więcej niż urzędowy limit*

**float** UdzialWZysku;*//0..1 Jaki procent zysku z innowacji pobiera (reszta idzie "w dół" hierarchii)*

*//float UdzialPromo; //0-1 Jakie środki są gotowi przeznaczyć na marketing (SKĄD JE BIORĄ?)*

*//float DecyzjaPatentowa;//Ile czasu jest podejmowana taka decyzja: wdrażamy vs. nie interesuje nas to*

};

*//Klasa wykonwacy/producenta ostatecznego produktu*

*////////////////////////////////////////////////////////////////////////////////*

**class** Firma:**public** \_ZespolRoboczy

{ *//Implementacja tego co musi być wg. interfejsu typu podstawowego węzła*

**static** KonstruktorElementowModelu<Firma> WirtKonstr;

ElementModelu::WirtualnyKonstruktor\* VKonstruktor() { **return** &WirtKonstr;}

**public**:

**void** ChwilaDlaCiebie(); *//Endogenne zmiany stanów, zarządzanie procesami itp.*

**void** InterpretujKomunikat(Komunikat\* Co);*//Przyjmowanie komunikatów*

Firma(){}*//Konstruktor domyslny*

**bool** ZrobWgListy(**const** std::string\* Lista,**unsigned** Ile,**unsigned**& Blad);

**void** AktualizujListeDanych();*//Do zapisu i wyświetlania inspekcyjnego*

*//Ruchy na koncie*

**void** PrzyjmijKase(**float** Suma) {FinanseBiezace+=Suma;} *//Żeby mogl rozliczac się finansowo z procesami*

**bool** StacGoNaWydatek(**float** Suma){ **return** Suma<FinanseBiezace;}*//Sprawdza czy taki wydatek jest dopuszczalny*

**protected**:*//Wlasciwości obiektów klasy i metody pomocnicze*

*//////////////////////////////////////////////////////////////////////////////////////*

**void** \_RozeslijNaleznosci(DziedzinaWKolorze Produkt,**float**& wplyw); *//Wg. identyfikatora produktu szuka komu winien dywidende*

*//Z KLASY: ZespolRoboczy*

*// unsigned IleEtatow;//Ile etatów. Co najmniej 1, 0 - "duch" zespołu - plan utworzenia*

*//float Efektywnosc;//0..2. Czy działa na 50% wydajności czy na 150%. Zależy od motywacji*

*//float Proaktywnosc; //Aktywność. 0-1 aktywność w tworzenia nowych znajomości biznesowych*

*//float Centralnosc;//0..1. Jak bardzo aktywność (zewnętrzna?) zależy od szefa/szefów*

*//float Doswiadczenie;//POZIOM EKSPERTYZY? Jak tego używać? Może to samo co efektywność???*

*//float Czasochlonnosc;//Jaki jest współczynnik czasochłonności typowej działalności (np. przygotowanie projektu oraz produktu)*

*//float Prolegalnosc;//?"Pragmatyczność"? Jak łatwo podjąć działania "szarostrefowe" i omijac "nieżyciowe" przepisy*

**float** ProAkademickosc; *//0-1 chęć nawiązywania kontaktów z badaczami i uniwersytetami*

**float** ProInnowac;*//Stosunek do nowości: 0-1 - 0..33% niewiele robi, 33-66% raczej blokuje innych,*

*//powyżej - próbuje, wchodząc na coraz wcześniejszym etapie (prawdopodobienstwo)*

**float** FinanseBiezace; *//Ile firma ma kasy aktualnie LUB UPROSZCZONA "ZAMOŻNOŚĆ"*

**float** FinanseZaplanowane;*//Ile firma ma obiecanych środków kredytowych*

**float** Dlugi;*//Ile firma ma do spłaty. Im więcej tym mniejsza szansa że coś dostanie nowego*

**float** LimitUdzialuInwestTT; *//0-1 Jakie środki finansowe z zarobionych gotów przeznaczyć na inwestycje TT*

*//float UdzialPromo; //0-1 Jakie środki sa gotowi przeznaczyć na marketing*

*//float Budzet*

*//float Wiarygodnosc;//Cecha połączenia? //0-1 na ile dana firma jest atrakcyjna*

*//dla uniwersytetu oraz banków w kontekście podejmowania z nią współpracy*

*//???float Wspolpraca\_Konkurenci; //chęć podejmowania współpracy z innymi firmami 0-1*

*//(te dwie zmienne powinny być od siebie niezależne, relacja o sumie niezerowej*

*//...*

};

*//Klasa organizatora wdrozenia innowacji (BOTT i UOTT)*

*////////////////////////////////////////////////////////////////////////////////*

*/\**

*class OrganizatorWdrozenia:public GenerycznyWezelSieci*

*{*

*//DYLEMAT - CZY TO JEDEN TYP Z "SUWAKAMI" POZWALAJĄCYMI ZMIENIĆ UOTT w niemal BOTT*

*//czy dwa zupełnie różne typy?*

*//Ostatecznie przyjęto że:*

*//Czy BOTT=firma+OTT a UOTT=AdmNaukowa+OTT*

*};*

*\*/*

**class** BOTT:**public** Firma

{ *//Implementacja tego co musi być wg. interfejsu typu podstawowego węzła*

**static** KonstruktorElementowModelu<BOTT> WirtKonstr;

ElementModelu::WirtualnyKonstruktor\* VKonstruktor() { **return** &WirtKonstr;}

**public**:

**void** ChwilaDlaCiebie(); *//Endogenne zmiany stanów, zarządzanie procesami itp.*

**void** InterpretujKomunikat(Komunikat\* Co);*//Przyjmowanie komunikatów*

BOTT(){}*//Konstruktor domyslny*

**bool** ZrobWgListy(**const** std::string\* Lista,**unsigned** Ile,**unsigned**& Blad);

**protected**:*//Wlasciwości obiektów klasy i metody pomocnicze*

*//////////////////////////////////////////////////////////////////////////////*

*//Ile Etatów Efektywnosc Aktywność Centralnosc Pronaukowosc Prolegalnosc Czasochlonnosc Proinnowacyjnosc Finans. bieżące Finanse zaplan Długi Limit ryzyk. TT EksperckoscTT Uwagi*

*//float LimitUdzialuInwestTT; //0-1 Jakie środki finansowe na RYZYKOWNE inwestycje TT*

*// float EksperckoscTT;=DOSWIADCZENIE//Dodatkowy współczynnik 0..2 skuteczności TT wynikający z doświadczenia*

*// float AktywnoscInform;//Aktywne poszukiwanie informacji "bazodanowej" (rynek, bazy, patenty etc)*

*// ... po prostu 0.5 normalnej aktywności???*

*//Kontakty z siatki*

*//Projekty z listy aktywnych projektów*

*//Nowe znajomości przez polecenie, konferencje, poszukiwanie/nawiązywanie kontaktów*

*//Ocena projektu? bezpośrednia i pośrednia*

};

**class** UOTT:**public** AdministracjaNaukowa

{ *//Implementacja tego co musi być wg. interfejsu typu podstawowego węzła*

**static** KonstruktorElementowModelu<UOTT> WirtKonstr;

ElementModelu::WirtualnyKonstruktor\* VKonstruktor() { **return** &WirtKonstr;}

**public**:

**void** ChwilaDlaCiebie(); *//Endogenne zmiany stanów, zarządzanie procesami itp.*

**void** InterpretujKomunikat(Komunikat\* Co);*//Przyjmowanie komunikatów*

UOTT(){}*//Konstruktor domyslny*

**bool** ZrobWgListy(**const** std::string\* Lista,**unsigned** Ile,**unsigned**& Blad);

**protected**:*//Wlasciwości obiektów klasy i metody pomocnicze*

*//////////////////////////////////////////////////////////////////////////////*

*//Ile Etatów Efektywnosc Aktywność Centralnosc Probiznesowosc Prolegalnosc Czasochlonnosc ProTT Finans. swobodne Finanse zaplan Dług do nadrz. Udzial w zysku TT EksperckoscTT Uwagi*

*//float UdzialWZysku;//0..1 Jaki procent zysku z innowacji pobiera i przekształca w motywację pracowników*

*//float EksperckoscTT;==Doswiadczenie//0..2 współczynnik skuteczności TT wynikający z doświadczenia*

*//float AktywnoscInform;//Aktywne poszukiwanie informacji "bazodanowej" (rynek, bazy, patenty etc)*

*// ... po prostu 0.5 normalnej aktywności???*

*//Kontakty z siatki*

*//Projekty z listy aktywnych projektów*

*//Ocena projektu??? - pośrednia przez wewnątrz uczelnianych ekspertów lub zewnętrzne firmy*

*//Trwa to dłużej*

};

*//Klasa powiązania administracyjnego - przepuszcza w górę raporty,*

*// a w dół wytyczne oraz pieniądze*

*////////////////////////////////////////////////////////////////////////////////*

**class** PodlegloscOrganizacyjna:**public** GenerycznePowiazanie

{ *//Implementacja tego co musi być wg. interfejsu typu podstawowego linku*

**static** KonstruktorElementowModelu<PodlegloscOrganizacyjna> WirtKonstr;

*//Dla usprawnienia klasa potomna może zwracać adres swojego wirtualnego konstruktora*

*//Choć można też nie definiować i polegac na gorszej funkcji z klasy bazowej*

ElementModelu::WirtualnyKonstruktor\* VKonstruktor() { **return** &WirtKonstr;}

**public**:

*//void ChwilaDlaCiebie(); //Endogenne zmiany stanów linku - tu brak ...*

**bool** Kierunkowy() { **return** **true**;} *//Wizualnie zachowuje się jak kierunkowy*

**bool** Akceptacja(Komunikat\* Co);*//Ale selekcja komunikatów jest złożona - mogą iść pod prąd*

PodlegloscOrganizacyjna(){}*//Konstruktor domyslny*

**bool** ZrobWgListy(**const** std::string\* Lista,**unsigned** Ile,**unsigned**& Blad);

**protected**:*//Właściwości obiektów klasy i metody pomocnicze*

*//... ???*

};

*// Klasa kooperacji sformalizowanej - przepuszcza w raporty - efekty działań*

*// oraz pieniądze*

*////////////////////////////////////////////////////////////////////////////////*

**class** FormalnaKooperacja:**public** PowiazaniePaboliczne

{ *//Implementacja tego co musi być wg. interfejsu typu podstawowego linku*

**friend** **class** KonstruktorElementowModelu<FormalnaKooperacja>;

**static** KonstruktorElementowModelu<FormalnaKooperacja> WirtKonstr;

ElementModelu::WirtualnyKonstruktor\* VKonstruktor() { **return** &WirtKonstr;}

**public**:

FormalnaKooperacja(**unsigned** Inicjator,**unsigned** Kooperator,**float** Waga,**float** Udzial=1,**unsigned** Termin=(10\*365));*//Konstruktor domyślny*

**void** ChwilaDlaCiebie(); *//Jak "Wykonany" to z czasem się coraz bardziej wygina, a jak termi spadnie do 0 to znika*

**bool** Akceptacja(Komunikat\* Co);*//Selekcja komunikatów*

**bool** Poprawny();*//Jak po terminie to znika*

**bool** ZrobWgListy(**const** std::string\* Lista,**unsigned** Ile,**unsigned**& Blad);

**void** Narysuj();*//Przechwycenie rysowania dla debugingu*

*//Akcesory*

**unsigned** DajTermin(){ **return** Termin;}*//Dostep do terminu*

**void** UstawTermin(**unsigned** NowyTermin){Termin = NowyTermin;}

**bool** CzyWykonany(){ **return** Wykonany;}

**void** UstawWykonano(**bool** TakNie=**true**){Wykonany=TakNie; }

**float** JakiUdzial(){ **if**(Waga<1) **return** Waga; **else** **return** 1;} *//Waga linku kooperacyjnego jest mierzona udzialem w zyskach!!!*

*//Właściwości obiektów klasy i metody pomocnicze*

**protected**:

FormalnaKooperacja(){Wykonany=0;Termin=(10\*365);}*//Konstruktor domyślny*

**bool** Wykonany;*//Czy już został zrealizowany i tylko płyną dywidendy*

**int** Termin;*//Maleje z każdym krokiem i jak 0 to link znika - koniec umowy*

};

*//Klasa powiązania towarzyskiego, zanikającego powoli gdy nie używany*

*// ASYMETRYCZNA - TAKIE LINKI SĄ POD DWA BO ZAUFANIE I WAGA POWIĄZANIA*

*// ZAZWYCZAJ JEST RÓŻNA Z PUNKTU WIDZENIA OBU PARTNERÓW!*

*//Komentarz na temat negatywności (do rozważenia) - hamująca rola węzła w przekazie*

*//informacji. Jak nie lubię bo zazdroszczę, to mu nie powiem,*

*//chociaż jesteśmy powiązani.*

*////////////////////////////////////////////////////////////////////////////////*

**class** KontaktTowarzyski:**public** PowiazaniePaboliczneSkierowane

{ *//Implementacja tego co musi być wg. interfejsu typu podstawowego linku*

**static** KonstruktorElementowModelu<KontaktTowarzyski> WirtKonstr;

ElementModelu::WirtualnyKonstruktor\* VKonstruktor() { **return** &WirtKonstr;}

**public**:

**void** ChwilaDlaCiebie(); *//Endogenne zmiany stanów linku - zanikanie...*

**bool** Poprawny(); *//true jeśli jest dobrze zdefiniowany. Wciąż istnieją oba konce itp.*

*// i WAGA jest większa niż TOWARZYSKA\_NAJMNIEJSZA\_WAGA*

**bool** Akceptacja(Komunikat\* Co);*//Selekcja komunikatów - podstawowa i wzrost wagi*

KontaktTowarzyski(){}*//Konstruktor domyslny*

KontaktTowarzyski(**unsigned** Kogo,**unsigned** ZKim,**double** JakaWaga,**double** JakieWygiecie);

**bool** ZrobWgListy(**const** std::string\* Lista,**unsigned** Ile,**unsigned**& Blad);

*//Specjalne dla tego typu*

**void** ZmienWage(**double** mnoznik);*//Zmienia bezpiecznie wagę powiazania. "mnoznik" może być większy lub mniejszy niż 1*

**protected**:*//Wlasciwości obiektów klasy i metody pomocnicze*

*//float Zaufanie == Waga linku . Dlatego skierowany bo zaufanie może być asymetryczne*

*//float TempoZanikania; //Czasem szybkie, czasem wolne, różne dla różnych osób*

*//Parametr wygięcia linku jest też skorelowane z tempem "parowania" nieużywanego*

*//linku socjalnego - im bardziej "poboczny" czyli wygięty link tym szybciej zanika*

};

*//Klasa komunikatu oficjalnego - raporty (w górę), wytyczne w dół (podleglosci adm.)*

*////////////////////////////////////////////////////////////////////////////////*

**class** KomunikacjaTowarzyska:**public** GeneryczneInfo

{ *//Implementacja tego co musi być wg. interfejsu typu podstawowego linku*

**friend** **class** KonstruktorElementowModelu<KomunikacjaTowarzyska>;

**static** KonstruktorElementowModelu<KomunikacjaTowarzyska> WirtKonstr;

ElementModelu::WirtualnyKonstruktor\* VKonstruktor() { **return** &WirtKonstr;}

**public**:

Komunikat\* Klonuj(); *//Robi na stercie kopie komunikatu do przechowania lub wstawienia*

*//Domyslny konstruktor wymaga kategori i opcjonalnie numeru wezla o którym mowa*

KomunikacjaTowarzyska(**const** **char**\* Rodzaj,**unsigned** OKogoChodzi=-1);

**const** string& Rodzaj();*//Gdzieś w danych jest rodzaj tego komunikatu*

**unsigned**& OKimTaGadka() {**return** OKim;} *//Na temat kogo jest ten komunikat - może o nadawcy, a może nie!*

**bool** Zwrotnie(**float** Szybkosc=0);*//Zwrotne adresowanie sluzy do pogrubiania linku na którego kom. odpowiadamy*

**bool** ZrobWgListy(**const** std::string\* Lista,**unsigned** Ile,**unsigned**& Blad); *//Jak generyczny*

**protected**:*//Wlasciwości obiektów klasy i metody pomocnicze*

**void** RysujKsztalt(**float** X,**float** Y,**float** Rad,**unsigned** R,**unsigned** G,**unsigned** B); *//Rysowanie kształtu zależnego od typu potomnego,*

**unsigned** OKim;

KomunikacjaTowarzyska(){OKim=-1;}*//Konstruktor domyślny - nie wiadomo o kim mowa*

KomunikacjaTowarzyska(**const** KomunikacjaTowarzyska\* Wzor):GeneryczneInfo(Wzor)

{OKim=Wzor->OKim;}*//KONSTRUKTOR KOPIUJĄCY przenosi też numer "obgadywanego" węzłą*

};

*//Klasa komunikatu oficjalnego - raporty (w górę), wytyczne w dół (podleglosci adm.)*

*////////////////////////////////////////////////////////////////////////////////*

**class** KomunikacjaOficjalna:**public** GeneryczneInfo

{ *//Implementacja tego co musi być wg. interfejsu typu podstawowego linku*

**friend** **class** KonstruktorElementowModelu<KomunikacjaOficjalna>;

**static** KonstruktorElementowModelu<KomunikacjaOficjalna> WirtKonstr;

ElementModelu::WirtualnyKonstruktor\* VKonstruktor() { **return** &WirtKonstr;}

**public**:

Komunikat\* Klonuj(); *//Robi na stercie kopie komunikatu do przechowania lub wstawienia*

KomunikacjaOficjalna(**const** **char**\* Rodzaj,**unsigned** Autor);*//Konstruktor z ustalonym rodzajem komunikatu*

**const** string& Rodzaj();*//Gdzieś w danych jest rodzaj tego komunikatu*

**unsigned** KtoJestAutorem(){ **return** Autor;}

**bool** ZrobWgListy(**const** std::string\* Lista,**unsigned** Ile,**unsigned**& Blad);

**protected**:*//Wlasciwości obiektów klasy i metody pomocnicze*

**void** RysujKsztalt(**float** X,**float** Y,**float** Rad,**unsigned** R,**unsigned** G,**unsigned** B); *//Rysowanie kształtu zależnego od typu potomnego,*

KomunikacjaOficjalna(){Autor=-1;}*//Konstruktor domyślny ukryty, żeby byle kto go nie używał :-)*

KomunikacjaOficjalna(**const** KomunikacjaOficjalna\* Wzor):GeneryczneInfo(Wzor) *//KONSTRUKTOR KOPIUJĄCY*

{Autor=Wzor->Autor;}

*//POLE:*

**unsigned** Autor; *//Kto jest autorem*

*//JAKIE RODZAJE:*

*//wytyczne w dół, raporty w górę,*

*//prośba o konsultacje prawną, o zgodę na kontakt/wspolprace*

*//odpowiedzi na to...*

*//...CO JESZCZE???*

};

*//Klasa komunikatu-pakietu do przesyłania realnych produktów z badań lub fabryk*

*// potrzebna główne ze względu na odróznienie wizualizacyjne*

*////////////////////////////////////////////////////////////////////////////////*

**class** PaczkaProduktow:**public** GeneryczneInfo

{ *//Implementacja tego co musi być wg. interfejsu typu podstawowego linku*

**friend** **class** KonstruktorElementowModelu<PaczkaProduktow>;

**static** KonstruktorElementowModelu<PaczkaProduktow> WirtKonstr;

ElementModelu::WirtualnyKonstruktor\* VKonstruktor() { **return** &WirtKonstr;}

**public**:

**friend** **class** ProcesProdukcyjny; *//Chyba musi miec dostęp do wnętrza paczki*

Komunikat\* Klonuj(); *//Robi na stercie kopie komunikatu do przechowania lub wstawienia*

PaczkaProduktow(**const** **char**\* Nazwa,**unsigned** IleWPaczce,**float** CenaZaSztuke,**unsigned** Producent);

**const** string& Rodzaj();*//Tu może być marką czy nazwą handlową - bo może być przesylany dalej!*

**unsigned** IleSztuk() {**return** Ile;}

**float** CenaZaSzt() {**return** Cena;}

**bool** ZrobWgListy(**const** std::string\* Lista,**unsigned** Ile,**unsigned**& Blad);

**protected**:*//Wlasciwości obiektów klasy i metody pomocnicze*

*//Wlasciwości obiektów klasy i metody pomocnicze*

**void** RysujKsztalt(**float** X,**float** Y,**float** Rad,**unsigned** R,**unsigned** G,**unsigned** B); *//Rysowanie kształtu zależnego od typu potomnego,*

PaczkaProduktow(){Ile=1;Cena=1;Producent=-1;}*//Konstruktor domyślny ukryty*

PaczkaProduktow(**const** PaczkaProduktow\* Wzor);*//KONSTRUKTOR KOPIUJĄCY*

*//POLE - LICZBA W PACZCE, CENA, ADRES PRODUCENTA*

**unsigned** Producent;

**unsigned** Ile;

**float** Cena;

*//JAKIE RODZAJE:*

*//\* Dziedzina to produkt ktorego dotyczy.*

};

*//Klasa komunikatu finansowego - obietnice finansowe i kredytowe, umowy, proformy, przelewy*

*////////////////////////////////////////////////////////////////////////////////*

**class** KomunikacjaFinansowa:**public** GeneryczneInfo

{ *//Implementacja tego co musi być wg. interfejsu typu podstawowego linku*

**friend** **class** KonstruktorElementowModelu<KomunikacjaFinansowa>;

**static** KonstruktorElementowModelu<KomunikacjaFinansowa> WirtKonstr;

ElementModelu::WirtualnyKonstruktor\* VKonstruktor() { **return** &WirtKonstr;}

**public**:

*//void ChwilaDlaCiebie(); //Zmiana stanów komunikatu - zaawansowanie przekazu*

Komunikat\* Klonuj(); *//Robi na stercie kopie komunikatu do przechowania lub wstawienia*

KomunikacjaFinansowa(**const** **char**\* Kategoria,**float** JakaKwota);*//Konstruktor domyslny*

**const** string& Rodzaj();*//Gdzieś w danych jest rodzaj tego komunikatu*

**float** JakaKwota() { **return** Kwota;}

**bool** ZrobWgListy(**const** std::string\* Lista,**unsigned** Ile,**unsigned**& Blad);

**protected**:*//Wlasciwości obiektów klasy i metody pomocnicze*

**void** RysujKsztalt(**float** X,**float** Y,**float** Rad,**unsigned** R,**unsigned** G,**unsigned** B); *//Rysowanie kształtu zależnego od typu potomnego,*

KomunikacjaFinansowa(){Kwota=0;}*//Sprywatyzowany konstruktor domyslny*

KomunikacjaFinansowa(**const** KomunikacjaFinansowa\* Wzor):GeneryczneInfo(Wzor)

{}*//KONSTRUKTOR KOPIUJĄCY*

*//JAKIE RODZAJE:*

**float** Kwota;

*//obietnice (propozale umów) finansowe i kredytowe,*

*//umowy, proformy,*

*//Płace, przelewy*

*//Raporty/wyniki/transfer wiedzy na koniec umowy o finansowanie badań*

*//Partie produktów na rynek*

*//pieniadze za produkty z powrotem do producenta*

};

*//Klasa dla świata - realizuje odpowiedzi na wolne poszukiwania, zapytania o produkty*

*// a także wymianę gotowych produktów na konkretną kasę (czyli "zbyt")*

*////////////////////////////////////////////////////////////////////////////////*

**class** RynekZbytu:**public** WezelMacierzowy

{ *//Implementacja tego co musi być wg. interfejsu typu podstawowego węzła*

**static** KonstruktorElementowModelu<RynekZbytu> WirtKonstr;

ElementModelu::WirtualnyKonstruktor\* VKonstruktor() { **return** &WirtKonstr;}

**public**:

RynekZbytu(){}*//Konstruktor domyślny*

**void** ChwilaDlaCiebie(); *//Endogenne zmiany stanów, zarządzanie procesami itp.*

**void** InterpretujKomunikat(Komunikat\* Co);*//Przyjmowanie komunikatów*

**bool** ZrobWgListy(**const** std::string\* Lista,**unsigned** Ile,**unsigned**& Blad);

**protected**:*//Wlasciwości obiektów klasy i metody pomocnicze*

DziedzinaWKolorze \_DaSieSprzedac(**const** DziedzinaWKolorze CoZaProdukt); *//Jak 0 to nie...*

*//GRID NISZ RYNKOWYCH:*

*//CO ROBI:*

*//\* realizuje odpowiedzi na swobodne poszukiwania inspiracji - losowo z filtrem*

*//\* zapytania o zbywalność/zapotrzebowanie na produkty z danym pomysłem*

*// \* Czy jest nisza rynkowa? Jak dokładne jest dopasowanie?*

*// \* Czy jest pusta czy zajęta i jak dokładnie zajęta?*

*//\* a także realizuje wymianę gotowych produktów*

*// na konkretną kasę (czyli "zbyt")*

*// Kolor (DZIEDZINA) rynku określa preferencje*

*//float SilaPreferencji;//0..1 - jak bardzo preferencja zmienia rozkład (?)*

*//albo wczytany grid o określonej zawartości*

};

*//Klasa dla PUBLIKATORA i UPATENTOWEGO - musi przyjmowac tylko komunikaty oficjalne*

*// a poza tym zachowuje się podobnie do klasy bazowej*

*///////////////////////////////////////////////////////////////////////////////////*

**class** SystemInformacyjny:**public** WezelMacierzowy

{ *//Implementacja tego co musi być wg. interfejsu typu podstawowego węzła*

**static** KonstruktorElementowModelu<SystemInformacyjny> WirtKonstr;

*//ElementModelu::WirtualnyKonstruktor\* VKonstruktor() { return &WirtKonstr;} //Wyswietlanie jak klasa bazowa?*

**public**:

SystemInformacyjny(){}*//Konstruktor domyślny*

*//void ChwilaDlaCiebie(); //Endogenne zmiany stanów, zarządzanie procesami itp. - jak klasa bazowa?*

**void** InterpretujKomunikat(Komunikat\* Co);*//Przyjmowanie komunikatów*

*//bool ZrobWgListy(const std::string\* Lista,unsigned Ile,unsigned& Blad); - jak klasa bazowa?*

};

*// Proces kreujący nowe linki - losuje węzły i wg. podobieństwa tworzy link i*

*// wysyła nim zajawkę o sobie, nadaje wagę proporcjonalną do wagi węzła (albo...)*

*// Jak nie utworzy linku to to wysyła przypomnienie do jakiegoś z istniejących*

*// powiązań społecznych zawierające losowe bity ze swojej dziedziny.*

*// Co jakiś czas aktualizuje dziedzinę na podstawie dziedziny swojego węzła.*

*// Jego procedura przechwytywania komunikatów zajmuje się odpowiadaniem na "zaczepki"*

*// o ile pochodzą od kogoś znanego, lub wartego poznania. Kieruje się tu wagą linku*

*// społecznego.*

*////////////////////////////////////////////////////////////////////////////////*

**class** ProcesSpoleczny:**public** GenerycznyProces

{

*//Implementacja tego co musi być wg. interfejsu typu Proces*

**static** KonstruktorElementowModelu<ProcesSpoleczny> WirtKonstr;

ElementModelu::WirtualnyKonstruktor\* VKonstruktor() { **return** &WirtKonstr;}

**public**:

ProcesSpoleczny(){}*//Konstruktor domyslny wykonuje robotę za pomocą domyśnego GenerycznyProces()*

**void** ChwilaDlaCiebie(); *//Tworzenie nowych kontaktów i podtrzymywanie starych*

*//Co jakiś czas (7 dni?) zaczyna się od początku*

**bool** InterpretujKomunikat(Komunikat\* Co);*//Ewentualnie odpowiada na komunikat*

**private**:

**float** Aktywnosc; *//Pobierana z odpowiedniej wartości węzla*

*//Procedury pomocnicze skracajace zapis tego co istotne*

**bool** \_WyslijAutoprezentacje(**unsigned** kanal,**bool** calaprawda); *//Wysyłania informacji o sobie*

**unsigned** \_StworzKanal(**unsigned** IndInny,**float** Waga,**float** Parametr=0);*//I nowy kanal z już obliczona wagą*

*//Jak Parametr =0 to jest losowanie ale z dodatnich!*

**public**:

*//Pomocnicza funkcja statyczna do użycia gdziebądź - znajduje zadane powiązanie spoleczne*

**static** **unsigned** \_JestPowiazanySocjalnie(**unsigned** Startowy,**unsigned** Docelowy);

};

*// Proces obciążający węzeł i na końcu wysyłający raport do odbiorcy lub węzła*

*// nadrzędnego (linkiem Adm.) i wzawiający działanie od poczatku bez nowego procesu*

*/////////////////////////////////////////////////////////////////////////////////////*

**class** ProcesRaportowany:**public** GenerycznyProces

{

*//Implementacja tego co musi być wg. interfejsu typu Proces*

**static** KonstruktorElementowModelu<ProcesRaportowany> WirtKonstr;

ElementModelu::WirtualnyKonstruktor\* VKonstruktor() { **return** &WirtKonstr;}

**public**:

ProcesRaportowany(){}*//Konstruktor domyslny wykonuje robotę za pomocą domyśnego GenerycznyProces()*

**void** ChwilaDlaCiebie(); *//Endogenne zmiany stanów, popychanie pracy do przodu itp.*

*//Na koniec wysyła raport i zaczyna się od początku*

**bool** InterpretujKomunikat(Komunikat\* Co);*//Przesyla dalej raporty, wysyła kase, inkasuje kasę*

**private**:

**bool** \_WyslijRaportOdbiorcy(); *//Pomocnicza procedura wysyłania raportu, gdy proces gotowy*

};

*// Proces zajmujący się dawaniem grantów, doatacji lub pożyczek na procent*

*/////////////////////////////////////////////////////////////////////////////////////*

**class** ProcesGrantowoPozyczkowy:**public** GenerycznyProces

{

*//Implementacja tego co musi być wg. interfejsu typu Proces*

**static** KonstruktorElementowModelu<ProcesGrantowoPozyczkowy> WirtKonstr;

ElementModelu::WirtualnyKonstruktor\* VKonstruktor() { **return** &WirtKonstr;}

**public**:

ProcesGrantowoPozyczkowy();*//Konstruktor domyslny wykonuje robotę za pomocą domyśnego GenerycznyProces()*

**void** ChwilaDlaCiebie(); *//Endogenne zmiany stanów, popychanie pracy do przodu itp.*

**bool** InterpretujKomunikat(Komunikat\* Co);*//Odbiera propozycje, a przydziela kasę*

**private**:

*//Wewnetrzne właściwości. Może używać też cech węzła, np. jego skłonności do ryzyka*

**float** Fundusz; *//Findusze do podziału*

**float** OczekiwanyZwrot;*//0 - dotacja, do 100% pożyczki częścowo zwrotne, pow 100% pozyczki komercyjne*

};

*// Proces badawczy produkuje wyniki naukowe do celów publikacji, patentowania,*

*// lub na czyjeś zamówienie, stąd ma właściwość "ZaplanowanyOdbiorca".*

*// Może się wznawiać orza generować inne procesy badawcze!!!*

*////////////////////////////////////////////////////////////////////////////////*

**class** ProcesBadawczy:**public** GenerycznyProces

{

**friend** **class** JednostkaBadawcza; *//Żeby mógł sobie sam poustawiać pola jak go tworzy od nowa*

*//Implementacja tego co musi być wg. interfejsu typu Proces*

**friend** **class** KonstruktorElementowModelu<ProcesBadawczy>;

**static** KonstruktorElementowModelu<ProcesBadawczy> WirtKonstr;

ElementModelu::WirtualnyKonstruktor\* VKonstruktor() { **return** &WirtKonstr;}

**public**:

*//Konstruktor z parametrami do tworzenia procesu w kodzie*

**enum** EfektNaukowy { NIEWIEM=-1, PUBLIKACJA, PATENT, RAPORT };

ProcesBadawczy(**const** **char**\* Nazwa,EfektNaukowy RodzajProd=NIEWIEM,**unsigned** ZaplanowanyOdbiorca=-1);*//Może od razu znać odbiorce i rodzaj produktu albo nie*

**void** ChwilaDlaCiebie(); *//Endogenne zmiany stanów, popychanie pracy do przodu itp.*

*//Na koniec tworzy swoją kopie i sibie przeznacza do skasowania*

**bool** InterpretujKomunikat(Komunikat\* Co);*//Przyjmowanie komunikatów*

**protected**: *//Ukryty przed innymi konstruktor domyslny wykonuje robotę za pomocą domyśnego GenerycznyProces()*

ProcesBadawczy(){ ZaplanowanyEfekt=NIEWIEM; ZaplOdbiorca=-1;}

**virtual** **float** \_KalkulujKosztBadan(); *//Furtka do "realistycznego" kalkulowania kosztów badań*

**bool** \_WyslijInformacje(**unsigned** link=-1); *//Do rozsylania informacji że się czyms takim zajmujemy - socjalna lub oficjalna*

**bool** \_WyslijPublikacje(); *//Pomocnicza procedura publikowania - domyslnie szuka węzła PUBLIKATOR*

**bool** \_WyslijPatent(); *//Pomocnicza procedura publikowania - domyslnie szuka węzła UPATENTOWY, publikacja wyklucza patent*

**bool** \_WyslijRaport(); *//Pomocnicza procedura raportowania - do odbiorcy albo administracji jak brak*

*//Główna specyficzna zmienna*

**private**:

EfektNaukowy ZaplanowanyEfekt;*//Jakiego rodzaju efekt ma tu być*

**unsigned** ZaplOdbiorca;*//Czy znany jest odbiorca i kto to*

};

*// Proces zajmujący się produkcją i wysyłąniem towarów do odbiorcy lub na rynek*

*// Wysyła pod koniec działania, a potem, jeśli wysłanie się powiodło to tworzy*

*// swoją kopię nieco bardziej efektywną (szybszą)*

*////////////////////////////////////////////////////////////////////////////////*

**class** ProcesProdukcyjny:**public** GenerycznyProces

{

*//Implementacja tego co musi być wg. interfejsu typu Proces*

**friend** **class** KonstruktorElementowModelu<ProcesProdukcyjny>;

**static** KonstruktorElementowModelu<ProcesProdukcyjny> WirtKonstr;

ElementModelu::WirtualnyKonstruktor\* VKonstruktor() { **return** &WirtKonstr;}

**public**:

*//Konstruktor z parametrami do tworzenia procesu w kodzie*

ProcesProdukcyjny(**unsigned** IleSztuk,**float** JakaCena=1,**unsigned** IleWPaczce=-1);*//Dzieli na 10 pak jak -1*

**void** ChwilaDlaCiebie(); *//Endogenne zmiany stanów, popychanie pracy do przodu itp.*

*//Na koniec tworzy swoją kopie i sibie przeznacza do skasowania*

**bool** InterpretujKomunikat(Komunikat\* Co);*//Przyjmowanie komunikatów*

**protected**: *//Ukryty przed innymi konstruktor domyslny wykonuje robotę za pomocą domyśnego GenerycznyProces()*

ProcesProdukcyjny(){Zapotrzebowanie=100;WPaczke=10;Cena=10;Wyslano=0;}

**virtual** **float** \_KalkulujKosztProduktu(); *//Furtka do "realistycznego" kalkulowania ceny produktu*

**bool** \_WyslijProduktyNaRynek(); *//Pomocnicza procedura wysyłania na rynek przy końcu procesu produkcji*

*//Główna specyficzna zmienna*

**private**:

**unsigned** Zapotrzebowanie; *//Na ile sztuk produktu szacujemy zapotrzebowanie*

**float** Cena; *//Jaka cena producenta za sztukę*

**unsigned** WPaczke; *//Po ile sztuk się miesci w paczce*

**unsigned** Wyslano; *//Ile juz poszło*

};

*// Proces poszukujący pomysłów na nowe produkty i "odpalajacy" czasem procesy TT*

*////////////////////////////////////////////////////////////////////////////////*

**class** ProcesPoszukiwanTT:**public** GenerycznyProces

{

*//Implementacja tego co musi być wg. interfejsu typu Proces*

**static** KonstruktorElementowModelu<ProcesPoszukiwanTT> WirtKonstr;

ElementModelu::WirtualnyKonstruktor\* VKonstruktor() { **return** &WirtKonstr;}

**public**:

ProcesPoszukiwanTT();*//Konstruktor domyslny wykonuje robotę za pomocą domyśnego GenerycznyProces()*

**bool** InterpretujKomunikat(Komunikat\* Co);*//Pobiera różne komunikaty i łączy z nich pomysły*

*//Tak żeby były zgodne z profilem firmy. Od czasu do czasu odpala*

*//zwykły ProcesTT*

**void** ChwilaDlaCiebie(); *//Endogenne zmiany stanów - nigdy się nie kończy...*

**private**:

*//...*

};

*// Proces przygotowujący nowy produkt - musi skompletować wszystkie składniki*

*//////////////////////////////////////////////////////////////////////////////////*

**class** ProcesTransferuTech:**public** GenerycznyProces

{

*//Implementacja tego co musi być wg. interfejsu typu Proces*

**friend** **class** KonstruktorElementowModelu<ProcesTransferuTech>;

**static** KonstruktorElementowModelu<ProcesTransferuTech> WirtKonstr;

ElementModelu::WirtualnyKonstruktor\* VKonstruktor() { **return** &WirtKonstr;}

**public**:

ProcesTransferuTech(DziedzinaWKolorze Pomysl);

*//Endogenne zmiany stanów - kończy się sukcesem a lbo porażką (z braku czasu lub środków)*

*//Tu wysyła różne zapytania w zależności od stanu checklisty. Sykces jest wtedy jak odchaczy wszystko*

**void** ChwilaDlaCiebie();

*//Poszukuje odpowiedzi na swoje pytania. W razie sukcesu odchacza,*

*//a w przypadku konieczności ustalenia formalnej koopereacji tworzy linki kooperacyjne.*

*//W razie porażki zamienia linki koop. na słabe powiązania socjalne (gdy ich brak)*

*//W razie sukcesu przetwarza linki koop. mocne socjalne (lub wzmacnia)*

*//I jeśli nie działa na bezpośrednim producencie to sieć kooperacji może przerzucić na produ*

*//na producenta bo jest konieczna do płacenia opłat licencyjnych z zysku.*

**bool** InterpretujKomunikat(Komunikat\* Co);

**private**:

ProcesTransferuTech(); *//Konstruktor domyślny wykonuje robotę za pomocą domyślnego GenerycznyProces()*

**void** \_ObsluzPorazke(); *//W razie porażki zamienia linki koop. na słabe powiązania socjalne (gdy ich brak) lub obniża wagi*

**void** \_ObsluzSukces(); *//Przekazuje produkt do produkcji i przekształca sieć linków*

*//Pola chekclisty*

**struct** ElementListy {

**double** Kiedy;**unsigned** Wykonawca;**bool** Check;

ElementListy(){Kiedy=-1;Wykonawca=-1;Check=**false**;}

};

**static** **const** **unsigned** PRODUCENT=24; *//Gdzie zaznaczamy kto wyprodukuje*

**static** **const** **unsigned** FINANSOWANIE=25; *//Gdzie zaznaczamy kto to finansuje*

ElementListy Fragmenty[32];*//Skąd są poszczególne fragmenty/elementy produktu*

};

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

*/\* OPI version 2011 \*/*

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

*/\* THIS CODE IS DESIGNED & COPYRIGHT BY: \*/*

*/\* W O J C I E C H B O R K O W S K I \*/*

*/\* Instytut Studiow Spolecznych Uniwersytetu Warszawskiego \*/*

*/\* WWW: http://wwww.iss.uw.edu.pl/borkowski/ \*/*

*/\* \*/*

*/\* (Don't change or remove this note) \*/*

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*

#endif