#include<iostream>

using namespace std;

#pragma region Example 1

//class Animal {

//protected:

// string name;

//public:

// Animal(const string&name):name(name){}

//

// string GetName()const {

// return name;

// }

//

// virtual void Speak() = 0 {

// cout << "I cry like all animal" << endl;

// }

//};

//

//class Dragon :public Animal {

//public:

// Dragon(const string&name):Animal(name)

// {

//

// }

// // Inherited via Animal

// virtual void Speak() override

// {

// Animal::Speak();

// }

//};

//

//

//

//void main() {

//

// //Animal\* animal = new Animal("New Animal");

// Animal\* dragon = new Dragon("Bomba Ejdaha");

// dragon->Speak();

//

//}

#pragma endregion

#pragma region Example 2

//class AbstractDatabase {

//public:

// virtual string GetById(int id) = 0;

// virtual void DeleteById(int id) = 0;

// virtual void Add(int id, string data) = 0;

// virtual void Update(int id, string data) = 0;

//};

//

//class OracleDb :public AbstractDatabase {

// // Inherited via AbstractDatabase

// virtual string GetById(int id) override

// {

// cout << "get from Oracle Database" << endl;

// return "";

// }

// virtual void DeleteById(int id) override

// {

// cout << "Delete from Oracle Database" << endl;

// }

// virtual void Add(int id, string data) override

// {

// cout << "Add to Oracle Database" << endl;

// }

// virtual void Update(int id, string data) override

// {

// cout << "Update in Oracle Database" << endl;

// }

//};

//

//class MsSql :public AbstractDatabase {

//public:

// // Inherited via AbstractDatabase

// virtual string GetById(int id) override

// {

// cout << "Get from MsSql Database" << endl;

// return "";

// }

//

// virtual void DeleteById(int id) override

// {

// cout << "Delete from MsSql Database" << endl;

// }

//

// virtual void Add(int id, string data) override

// {

// cout << "Add to MsSql Database" << endl;

// }

//

// virtual void Update(int id, string data) override

// {

// cout << "Update in MsSql Database" << endl;

// }

//};

//

//

//class Website {

// string url;

// AbstractDatabase\* database;

//public:

// Website(AbstractDatabase\* db)

// {

// database = db;

// }

//

// void MakeSomething() {

// database->Add(1, "New Data");

// database->DeleteById(2);

// database->DeleteById(116);

//

// //some big codes

//

// database->GetById(10);

//

// }

//};

//

//void main() {

// Website website(new MsSql());

// website.MakeSomething();

//}

#pragma endregion

#pragma region Example 3

//class Musician {

//protected:

// string name;

// string instrument;

//public:

// Musician() = delete;

// Musician(string name, string instrument) :name(name), instrument(instrument) {

//

// }

// virtual ~Musician() = 0

// {

// }

//};

//

//

//class A :public Musician {

//public:

// A(string s,string s2):Musician(s,s2){}

//

//};

//

//void main() {

// A a("", "");

//}

#pragma endregion

#pragma region Example 4

//class Musician {

//protected:

// string name;

// string instrument;

//public:

// Musician() = delete;

// Musician(string n,string i):name(n),instrument(i)

// {

// }

//

// virtual void perform() = 0 {

// cout << "Sazina qurban .. . . . . Sozune qurban" << endl;

// }

// virtual ~Musician()=0//pure virtual destructor

// {

// }

//};

//

//class Guitarist :public Musician {

// string nickname;

//public:

// Guitarist() = delete;

// Guitarist(string name, string instrument, string nickname)

// :Musician(name, instrument) {

// this->nickname = nickname;

// }

//

// void perform()override {

// cout << "Ohhhh guitar sound . . ." << endl;

// }

//};

//

//class Ashiq :public Musician {

// bool isEloglu;

//public:

// Ashiq() = delete;

// Ashiq(string name, string instrument)

// :Musician(name,instrument)

// {

// this->isEloglu = true;

// }

//

// void perform()override {

//

// Musician::perform();

// }

//};

//

//

//class Band {

// Musician\*\* musicians;

// int count = 0;

//public:

// void AddMusician(Musician\* musician) {

// auto newdata = new Musician \* [count + 1]{};

// for (size\_t i = 0; i < count; i++)

// {

// newdata[i] = musicians[i];

// }

// newdata[count] = musician;

// ++count;

// musicians = newdata;

// newdata = nullptr;

// }

//

// void Start() {

// for (size\_t i = 0; i < count; i++)

// {

// musicians[i]->perform();

// }

// }

//};

//

//

//void main() {

//

// Band stars;

// stars.AddMusician(new Guitarist("Jordon", "guitarBBGTR123", "John"));

// stars.AddMusician(new Ashiq("Telli", "Saz"));

// stars.AddMusician(new Ashiq("Arif Telsiz", "Saz"));

//

// stars.Start();

//

//}

#pragma endregion

#pragma region HomeTask

class Ingredient {

string name;

double fats;

double protein;

double carbohydrates;

int kcal;

public:

//ctor destructor

//getter setter

//pure virtual destr

};

class Mashroom :public Ingredient {

public:

Mushroom(string name,fats,protein,carbohydrates,kcal):

Ingredient{

}

};

class Meat :public Ingredient {

};

class Spices :public Ingredient {

};

class Cheese :public Ingredient {

};

class Tomatoes :public Ingredient {

};

struct RecipeItem {

Ingredient\* ingredient;// 2 dene ingredient

int amount = 0;// neche dene

};

//Yemek DISH

class Dish {

RecipeItem\*\* items;

int count;

string name;

public:

void AddIngredient(RecipeItem\* item) {

}

virtual void Taste() {

cout << "I do not know what is this ? " << endl;

}

virtual void PrintRecipe() {

}

virtual ~Dish() = 0 { //pure vitual destructor

}

};

class Pizza :public Dish {

};

class Kabab :public Dish {

};

class Dolma :public Dish {

};

class CookingDevice {

protected:

Dish\* dish;

bool hasCooked;

int second;

virtual void Cook() = 0;

};

class Oven :public CookingDevice {

};

class Brazzier :public CookingDevice {

};

class Pot :public CookingDevice {

};