# КУРС: «ПЛАТФОРМА MICROSOFT .NET ТА МОВА ПРОГРАМУВАННЯ C#

Тема: Успадкування

**Модуль 6**

**Завдання 1**

Запрограмуйте клас Money (об'єкт класу оперує однією валютою) для роботи з грошима.

У класі мають бути передбачені: поле для зберігання цілої частини грошей (долари, євро, гривні тощо) і поле для зберігання копійок (центи, євроценти, копійки тощо)

Реалізувати методи виведення суми на екран, задання значень частин.

На базі класу Money створити клас Product для роботи з продуктом або товаром. Реалізувати метод, який дозволяє зменшити ціну на задане число.

Для кожного з класів реалізувати необхідні методи і поля.

**Завдання 2**

Створити базовий клас «Пристрій» і похідні класи:

«Чайник», «Мікрохвильовка», «Автомобіль», «Пароплав». За допомогою конструктора, встановити назву до кожного пристрою і його характеристики.

Реалізуйте для кожного з класів методи:

* Sound — видає звук пристрою (пишемо текстом у консоль);
* Show — відображає назву пристрою;
* Desc — відображає опис пристрою.

**Завдання 3**

Створити базовий клас «Музичний інструмент» і похідні класи: «Скрипка», «Тромбон», «Укулеле», «Віолончель». За допомогою конструктора, встановити назву до кожного музичного інструменту і його характеристики.

Реалізуйте для кожного з класів методи:

* Sound — видає звук музичного інструменту (пишемо текстом у консоль);
* Show — відображає назву музичного інструменту;
* Desc — відображає опис музичного інструменту;
* History — відображає історію створення музичного інструменту.

**Завдання 4**

Створити абстрактний базовий клас Worker (працівник) з методом Print(). Створіть чотири похідні класи: President, Security, Manager, Engineer. Перевизначте метод.

Print() для виведення інформації, яка відповідає кож- ному типу працівника.

Завдання 1

using System;

namespace test11

{

class Money

{

protected int wholepart;

protected int fractionalpart;

protected string formatwhole;

protected string formatfractional;

public int Wholepart

{

set{ this.wholepart = value; }

get{ return this.wholepart; }

}

public int Fractionalpart

{

set { this.fractionalpart = value; }

get { return this.fractionalpart; }

}

public string Formatwhole

{

set { this.formatwhole = value; }

get { return this.formatwhole; }

}

public string Formatfractional

{

set { this.formatfractional = value; }

get { return this.formatfractional; }

}

public Money(int wholepart, int fractionalpart, string formatwhole, string formatfractional)

{

this.fractionalpart = fractionalpart;

this.wholepart = wholepart;

this.formatwhole = formatwhole;

this.formatfractional = formatfractional;

}

public Money()

{

this.fractionalpart = 0;

this.wholepart = 0;

this.formatwhole = "unknown";

this.formatfractional = "unknown";

}

public void PrintInfo()

{

Console.WriteLine(wholepart + " " + formatwhole + " " + fractionalpart + " " + formatfractional);

}

}

class Product:Money

{

string name;

string material;

public string Name

{

set { this.name = value; }

get { return this.name; }

}

public string Material

{

set { this.material = value; }

get { return this.material; }

}

public Product(string name, string material,int wholepart, int fractionalpart, string formatwhole, string formatfractional):base(wholepart,fractionalpart,formatwhole,formatfractional)

{

this.name = name;

this.material = material;

}

public void PrintInfo()

{

Console.WriteLine(name);

Console.WriteLine("Матеріал: " + material);

base.PrintInfo();

}

public void ReducePrice(int wholepart, int fractionalpart)

{

int moneybasic = this.wholepart \* 100 + this.fractionalpart;

int moneyreduce = wholepart \* 100 + fractionalpart;

if (moneybasic > moneyreduce)

{

moneybasic -= moneyreduce;

this.wholepart = moneybasic / 100;

this.fractionalpart = moneybasic % 100;

}

else

Console.WriteLine("Суму скидки більша за ціну продукта");

}

}

class Program

{

public static void Main()

{

System.Console.OutputEncoding = System.Text.Encoding.UTF8;

Money money=new Money();

money.Wholepart = 12;

money.Fractionalpart = 99;

money.Formatwhole = "Dolars";

money.Formatfractional = "cent";

money.PrintInfo();

Console.WriteLine("------------------------------------");

Product product = new Product("Iphone 14", "titan", 1499, 98, "dolars", "cent");

product.PrintInfo();

Console.WriteLine("------------------------------------");

product.ReducePrice(100, 99);

product.PrintInfo();

}

}

}

Завдання 2

using System;

namespace test11

{

abstract class Device

{

protected string name;

protected int year\_of\_manufacture;

public Device(string name, int year\_of\_manufacture)

{

this.name = name;

this.year\_of\_manufacture = year\_of\_manufacture;

}

public virtual void Sound(){}

public virtual void Show() {}

public virtual void Desk() {}

}

class Kettle:Device

{

int power;

string material;

float size;

public Kettle(string name, int year\_of\_manufacture, int power, string material, float size):base(name, year\_of\_manufacture)

{

this.power = power;

this.material = material;

this.size = size;

}

public override void Desk()

{

Console.WriteLine("Чайник: " + this.name);

Console.WriteLine("Рік виготовлення: " + this.year\_of\_manufacture);

Console.WriteLine("Потужність: " + this.power+"W");

Console.WriteLine("Матеріал: " + this.material);

Console.WriteLine("Ємкість: " + this.size+"л");

}

public override void Show()

{

Console.WriteLine("Чайник: " + this.name);

}

public override void Sound()

{

Console.WriteLine("Пшпшпшпш чайник закипів");

}

}

class Microwave:Device

{

int power;

int levelpower;

string modes;

public Microwave(string name, int year\_of\_manufacture, int power, int levelpower, string modes): base(name, year\_of\_manufacture)

{

this.power = power;

this.levelpower = levelpower;

this.modes = modes;

}

public override void Desk()

{

Console.WriteLine("Мікрохвильовка: " + this.name);

Console.WriteLine("Рік виготовлення: " + this.year\_of\_manufacture);

Console.WriteLine("Потужність: " + this.power+"W");

Console.WriteLine("Рівнів потужностей: " + this.levelpower);

Console.WriteLine("Режими: " + this.modes);

}

public override void Show()

{

Console.WriteLine("Мікрохвильовка: " + this.name);

}

public override void Sound()

{

Console.WriteLine("Піп піп піп мікрохвильовка закінчила роботу");

}

}

class Car:Device

{

string engine;

string gearbox;

string drive;

public Car(string name, int year\_of\_manufacture, string engine, string gearbox, string drive): base(name, year\_of\_manufacture)

{

this.engine = engine;

this.gearbox = gearbox;

this.drive = drive;

}

public override void Desk()

{

Console.WriteLine("Машина: " + this.name);

Console.WriteLine("Рік виготовлення: " + this.year\_of\_manufacture);

Console.WriteLine("Двигун: " + this.engine);

Console.WriteLine("Коробка передач: " + this.gearbox);

Console.WriteLine("Привід: " + this.drive);

}

public override void Show()

{

Console.WriteLine("Машина: " + this.name);

}

public override void Sound()

{

Console.WriteLine("vrum vrum vrum газує машина");

}

}

class Steamer:Device

{

string type;

float cargocapacity;

public Steamer(string name, int year\_of\_manufacture, string type, float cargocapacity) : base(name, year\_of\_manufacture)

{

this.type = type;

this.cargocapacity = cargocapacity;

}

public override void Desk()

{

Console.WriteLine("Пароплав: " + this.name);

Console.WriteLine("Рік виготовлення: " + this.year\_of\_manufacture);

Console.WriteLine("Вид пароплава: " + this.type);

Console.WriteLine("Вантажність: " + this.cargocapacity+"т");

}

public override void Show()

{

Console.WriteLine("Пароплав: " + this.name);

}

public override void Sound()

{

Console.WriteLine("too too too гуде пароплав");

}

}

class Program

{

public static void Main()

{

System.Console.OutputEncoding = System.Text.Encoding.UTF8;

Device[] devices = new Device[4];

devices[0] = new Kettle("Tefal", 2007, 2200, "пластик", 1.7F);

devices[1] = new Microwave("Ergo", 2020, 700, 3, "авторозмороження, гриль");

devices[2] = new Car("Mersedes", 2015, "2.4 дизель", "механіка", "задній");

devices[3] = new Steamer("Фултон", 1920, "колісний", 15);

for(int i=0;i<devices.Length;i++)

{

devices[i].Show();

}

Console.WriteLine("--------------------------------");

Console.WriteLine("--------------------------------");

for (int i = 0; i < devices.Length; i++)

{

devices[i].Desk();

Console.WriteLine("--------------------------------");

}

Console.WriteLine("--------------------------------");

for (int i = 0; i < devices.Length; i++)

{

devices[i].Sound();

}

}

}

}

Завдання 3

using System;

namespace test11

{

abstract class MusicInstrument

{

protected string name;

protected string history;

public MusicInstrument(string name, string history)

{

this.name = name;

this.history = history;

}

public virtual void Sound(){}

public virtual void Show() { }

public virtual void Desc() {}

public virtual void History() {}

}

class Violin: MusicInstrument

{

int number\_of\_strings;

public Violin(string name, string history,int number\_of\_strings) :base(name, history)

{

this.number\_of\_strings = number\_of\_strings;

}

public override void Sound()

{

Console.WriteLine("Звучить скрипка");

}

public override void Show()

{

Console.WriteLine("Скрипка: " + name);

}

public override void Desc()

{

Console.WriteLine("Скрипка: " + name);

Console.WriteLine("Кількість струн: " + number\_of\_strings+" штук");

}

public override void History()

{

Console.WriteLine("Історія скрипки " + name);

Console.WriteLine(history);

}

}

class Trombone:MusicInstrument

{

int bell\_diameter;

public Trombone(string name, string history, int bell\_diameter):base(name, history)

{

this.bell\_diameter = bell\_diameter;

}

public override void Sound()

{

Console.WriteLine("Звучить тромбон");

}

public override void Show()

{

Console.WriteLine("Тромбон: " + name);

}

public override void Desc()

{

Console.WriteLine("Тромбон: " + name);

Console.WriteLine("Діаметр розтруба: " + bell\_diameter + "мм");

}

public override void History()

{

Console.WriteLine("Історія тромбону " + name);

Console.WriteLine(history);

}

}

class Ukulele:MusicInstrument

{

int number\_of\_strings;

public Ukulele(string name, string history, int number\_of\_strings) : base(name, history)

{

this.number\_of\_strings = number\_of\_strings;

}

public override void Sound()

{

Console.WriteLine("Звучить укулеле");

}

public override void Show()

{

Console.WriteLine("Укулеле: " + name);

}

public override void Desc()

{

Console.WriteLine("Укулеле: " + name);

Console.WriteLine("Кількість струн: " + number\_of\_strings + " штук");

}

public override void History()

{

Console.WriteLine("Історія укулеле " + name);

Console.WriteLine(history);

}

}

class Violoncello:MusicInstrument

{

int number\_of\_strings;

public Violoncello(string name, string history, int number\_of\_strings) : base(name, history)

{

this.number\_of\_strings = number\_of\_strings;

}

public override void Sound()

{

Console.WriteLine("Звучить віолончель");

}

public override void Show()

{

Console.WriteLine("Віолончель: " + name);

}

public override void Desc()

{

Console.WriteLine("Віолончель: " + name);

Console.WriteLine("Кількість струн: " + number\_of\_strings + " штук");

}

public override void History()

{

Console.WriteLine("Історія віолончелі " + name);

Console.WriteLine(history);

}

}

class Program

{

public static void Main()

{

System.Console.OutputEncoding = System.Text.Encoding.UTF8;

MusicInstrument[] musicInstruments = new MusicInstrument[4];

musicInstruments[0] = new Violin("Parkons", "З'явився у середині XVI століття.", 4);

musicInstruments[1] = new Trombone("J.MICHAEL TB-600VJ (S)", "З'явився у XV століття.", 220);

musicInstruments[2] = new Ukulele("Fzone FZU", "З'явився у XIX століття.", 4);

musicInstruments[3] = new Violoncello("Віолончель Hora C120", "Поява віолончелі відноситься до початку XVI століття.", 4);

for(int i=0;i< musicInstruments.Length;i++)

{

musicInstruments[i].Show();

Console.WriteLine("-------------------------------------------");

}

Console.WriteLine("-------------------------------------------");

for (int i = 0; i < musicInstruments.Length; i++)

{

musicInstruments[i].Desc();

Console.WriteLine("-------------------------------------------");

}

Console.WriteLine("-------------------------------------------");

for (int i = 0; i < musicInstruments.Length; i++)

{

musicInstruments[i].Sound();

Console.WriteLine("-------------------------------------------");

}

Console.WriteLine("-------------------------------------------");

for (int i = 0; i < musicInstruments.Length; i++)

{

musicInstruments[i].History();

Console.WriteLine("-------------------------------------------");

}

}

}

}

Завдання 4

using System;

namespace test11

{

abstract class Worker

{

protected string name;

protected string lastname;

protected string surname;

protected string responsibilities;

public Worker(string name, string lastname, string surname, string responsibilities)

{

this.name = name;

this.lastname = lastname;

this.surname = surname;

this.responsibilities = responsibilities;

}

public virtual void Print(){}

}

class President : Worker

{

public President(string name, string lastname, string surname, string responsibilities):base(name, lastname, surname, responsibilities){}

public override void Print()

{

Console.WriteLine("Президент: " + name + " " + lastname + " " + surname);

Console.WriteLine("Обов'язки:\n" + responsibilities);

}

}

class Security : Worker

{

public Security(string name, string lastname, string surname, string responsibilities) : base(name, lastname, surname, responsibilities) { }

public override void Print()

{

Console.WriteLine("Охоронець: " + name + " " + lastname + " " + surname);

Console.WriteLine("Обов'язки:\n" + responsibilities);

}

}

class Manager : Worker

{

public Manager(string name, string lastname, string surname, string responsibilities) : base(name, lastname, surname, responsibilities) { }

public override void Print()

{

Console.WriteLine("Менеджер: " + name + " " + lastname + " " + surname);

Console.WriteLine("Обов'язки:\n" + responsibilities);

}

}

class Engineer : Worker

{

public Engineer(string name, string lastname, string surname, string responsibilities) : base(name, lastname, surname, responsibilities) { }

public override void Print()

{

Console.WriteLine("Інженер: " + name + " " + lastname + " " + surname);

Console.WriteLine("Обов'язки:\n" + responsibilities);

}

}

class Program

{

public static void Main()

{

System.Console.OutputEncoding = System.Text.Encoding.UTF8;

Worker[] workers = new Worker[4];

workers[0] = new President("Лащ", "Антон", "Вадимович", "Керувати проєктом");

workers[1] = new Manager("Деркач", "Іван", "Русланович", "Допомагати організувути роботу, допомагати президенту");

workers[2] = new Security("Швайко", "Микола", "Васильович", "Охороняти об'єкт, та співробітників по проєкту");

workers[3] = new Engineer("Жилко", "Василь", "Юрієвич", "Підтримувати роботоспособність техніки");

for(int i=0;i<workers.Length;i++)

{

workers[i].Print();

Console.WriteLine("------------------------------------------");

}

}

}

}