PROGRAMOZÁS II
6. ÓRA: VIRTUÁLIS FÜGGVÉNYEK, ABSZTRAKT OSZTÁLYOK

VIRTUÁLIS METÓDUSOK

Virtualitás: mi és miért?

```
class Person
{
    public string Name { get; }
    public Person(string name)
    { Name = name; }
    public void Print()
    { Console.WriteLine(Name); }
}

class Student : Person
{
    public string NeptunCode { get; }
    public Student(string name, string neptunCode): base(name)
    { NeptunCode = neptunCode; }
    public void Print()
    { Console.WriteLine($"{Name} : {NeptunCode}"); }
}
```

Virtualitás: mi és miért?

- Mi történik?

```
Person p = new Person("Adam West");
p.Print();
Student s = new Student("Jane Doe", "ABCDEF");
s.Print();
```

Virtualitás: mi és miért?

- Mi történik?

```
Person p = new Person("Adam West");
p.Print();
Student s = new Student("Jane Doe", "ABCDEF");
s.Print();
```

Adam West

Jane Doe : ABCDEF

Virtualitás: mi és miért?

- Mi történik?

```
Person p = new Person("Adam West");
p.Print();
Student s = new Student("Jane Doe", "ABCDEF");
s.Print();
```

Adam West

Jane Doe : ABCDEF

- És most?

```
Person p2 = new Student("Jim East", "XZ1234");
p2.Print();
```

Virtualitás: mi és miért?

- Mi történik?

```
Person p = new Person("Adam West");
p.Print();
Student s = new Student("Jane Doe", "ABCDEF");
s.Print();
```

Adam West

Jane Doe : ABCDEF

- És most?

```
Person p2 = new Student("Jim East", "XZ1234");
p2.Print();
```

Jim East

Mi történt?

- Ősosztály referenciával hivatkoztunk gyerek objektumra
- Meghívtunk egy függvényt → ősosztály verziója hívódott meg
- Mi van, ha személyeket általánosan szeretnénk kezelni, de a gyerekosztályoknál a saját kiírásukat szeretnénk látni → virtuális metódusok

Virtuális metódusok

```
class Person
{
    public string Name { get; }
    public Person(string name)
    { Name = name; }
    public virtual void Print()
    { Console.WriteLine(Name); }
}

class Student : Person
{
    public string NeptunCode { get; }
    public Student(string name, string neptunCode): base(name)
    { NeptunCode = neptunCode; }
    public override void Print()
    { Console.WriteLine($"{Name} : {NeptunCode}"); }
}
```

Virtuális metódusok

- Szerepük: ősosztály referenciával hivatkozunk gyerek objektumra, akkor a metódus gyerekosztály-beli verziója hívódjon meg
- Elemei:
  - Ősosztályban: virtual
  - Gyerekosztályban: override

```
class Person
{
    …
    public virtual void Print()
    { … }
}
class Student : Person
{
    …
    public override void Print()
    { … }
}
```

```
Person p2 = new Student("Jim East", "XZ1234");
p2.Print();
```

Jim East : XZ1234

Virtual property

- Property is lehet virtuális
  - Mivel a property csak egy rövid szintaktika a getter-setter metódusokra

```
class Person
{
    public virtual string Name { get; }
    …
}

class PersonWithPrefix : Person
{
    public string Prefix { get; set; }
    public PersonWithPrefix(string prefix, string name) : base(name)
    { Prefix = prefix; }
    public override string Name
    {  get { return Prefix+" "+base.Name; } }
}
```

Felülírja az ős-beli Name property getterét

Virtual property

```
Person pf1 = new PersonWithPrefix("Dr.", "Amelia Dean");

Console.WriteLine(pf1.Name);
```

```
Dr. Amelia Dean
```

Virtual property

```
class Person
{
    public virtual string Name { get; }
    public Person(string name)
    { Name = name; }
    public virtual void Print()
    { Console.WriteLine(Name); }
}
```

```
Person pf1 = new PersonWithPrefix("Dr.", "Amelia Dean");

pf1.Print();
```

Dr. Amelia Dean

A Print metódust a gyerekosztály nem írta felül, de mivel a metóduson belül a this továbbra is gyerek objektumra hivatkozik, a virtuális Name getter ezt figyelembe veszi.

Virtual kulcsszó

- Fontos: A virtual kulcsszó nem használható egyszerre a static, private, abstract, override kulcsszavakkal.

ABSZTRAKT OSZTÁLYOK

Absztrakt osztály

- Mi történik, amikor az ős annyira általánosít, hogy bizonyos műveleteknek nincs értelme?
- Példa:
  - Egy körnek ki tudjuk számolni a területét
  - Egy téglalapnak úgyszintén
  - A kettő általánosítható alakzat típussal
  - Mi egy alakzat területe?
    - Nem értelmezhető, ha nem tudjuk, milyen alakzat

Absztrakt osztály

abstract kulcsszó

```
abstract class Shape
{
    public abstract double Area();
}

class Circle : Shape
{
    double _radius;
    public Circle(double radius)
    { _radius = radius; }
    public override double Area()
    { return _radius*_radius*3.141592; }
}
```

```
class Rectangle : Shape
{
    double _width;
    double _height;
    public Rectangle(double width,
                     double height)
    {
        _width = width;
        _height = height;
    }
    public override double Area()
    { return _width * _height; }
}
```

```
Shape sh1 = new Circle(5.4);
Shape sh2 = new Rectangle(2.3, 6.7);
Console.WriteLine(sh1.Area());
Console.WriteLine(sh2.Area());
```

Absztrakt osztály

- Absztrakt osztályt nem lehet példányosítani
- Ha származtatunk nem absztrakt gyerekosztályt, akkor minden absztrakt metódusnak kell, hogy legyen kifejtése
- Származtatható olyan osztály, amely nem fejti ki az ős absztrakt metódusát, ekkor a gyerek osztály is absztrakt lesz

```
abstract class NamedShape : Shape
{
    public string Name { get; }
    public NamedShape(string name)
    {
        Name = name;
    }
}
```

TOSTRING

Objektumok kiíratása

```
class Coordinate
{
    public double X { get; }
    public double Y { get; }
    public Coordinate(double x, double y)
    {
        X = x;
        Y = y;
    }
}
```

```
Coordinate c1 = new Coordinate(4.5, 7.6);
Coordinate c2 = new Coordinate(1.2, 9.1);
Console.WriteLine(c1); //Examples.Coordinate
Console.WriteLine(c2); //Examples.Coordinate
```

Objektumok kiíratása

- A parancssorra kiíratott objektumnak a típusa jelenik meg
- Ha az értékét szeretnénk kiíratni, akkor vagy megjelenítjük egyesével, vagy készítünk az osztályba erre metódust
- Van erre általános módszer:
  - ToString

ToString metódus

- C#-ban minden osztály az object ősosztályból származik
- A ToString az object egyik virtuális metódusa, amelynek célja, hogy vezérelje az objektum szöveggé alakítását
- A Console.WriteLine hívásakor a ToString hívódik meg, hogy a kimeneten megjelenő adat formáját meghatározza

ToString metódus

```
class Coordinate
{
    public double X { get; }
    public double Y { get; }
    public Coordinate(double x, double y)
    {
        X = x;
        Y = y;
    }
    public override string ToString()
    {
        return $"({X},{Y})";
    }
}
```

```
Coordinate c1 = new Coordinate(4.5, 7.6);
Coordinate c2 = new Coordinate(1.2, 9.1);
Console.WriteLine(c1); //(4.5,7.6)
Console.WriteLine(c2); //(1.2, 9.1)
```

A c1.ToString() által visszaadott szöveget írja ki

ToString metódus

- Ez történik szöveg interpoláció esetén is

```
Coordinate c1 = new Coordinate(4.5, 7.6);
Coordinate c2 = new Coordinate(1.2, 9.1);

string data = $"c1: {c1}, c2: {c2}";
Console.WriteLine(data);
//c1: (4.5, 7.6), c2: (1.2, 9.1)
```

NAGY PÉLDA: KÉRDÉSBANK

Nagy példa: kérdésbank

- Vizsgáztató kérdéssor rendszer
  - A kérdés sor kérdésekből áll
  - A kérdéseknek több fajtája van
    - Feleletválasztós
    - Igaz-hamis
    - Szám válasz
  - A kérdéseket egy kérdésbank tárolja, így újrahasználhatóak
  - A kérdéseket egységesen szeretnénk tárolni

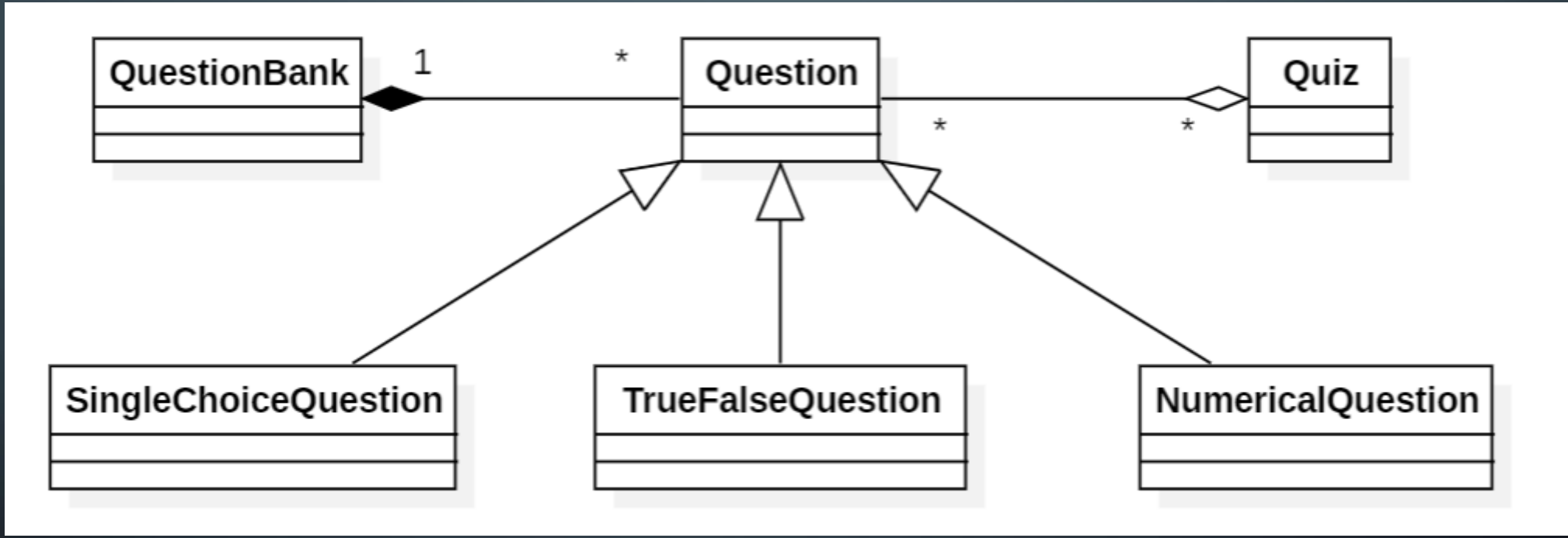
Milyen osztályok legyenek a kérdésekhez

- Egyes kérdés típusok külön osztályokban
  - SingleChoiceQuestion
  - TrueFalseQuestion
  - NumericalQuestion
- Van közös tudás és felelősség
  - Azonosító
  - Kérdés szövege
  - Megjelenítés
  - Válasz kiértékelés
- Célszerű egy közös ősosztály
  - Question

Kérdésbank és tesztek felépítése

- A kérdésbank tárolja az összes kérdést
  - Tartalmazás
- A kérdés sorok ezekből a kérdésekből épülnek fel
  - Hivatkozás

Osztályok


QuestionBank
1
*
Question
*
*
Quiz
SingleChoiceQuestion
TrueFalseQuestion
NumericalQuestion

Attribútumok

- Minden kérdésnek van
  - Azonosító
  - Kérdés szövege
- Feleletválasztós kérdés
  - Választási lehetőségek
  - Helyes válasz
- Igaz-hamis kérdés
  - Helyes válasz
- Szám válaszos kérdés
  - Helyes válasz

Helyes válasz mindenhol van, de a típusa és/vagy jellege más

Attribútumok

- Kérdésbank
  - Az összes kérdés
- Kérdés sor
  - Kérdések (adott sorrendben)

Felelősségei

- Kérdések
  - Kérdés szövegének visszaadása
  - Válasz értékelése
- Teszt
  - Kérdések kilistázása
  - Válaszok bekérése
  - Összesítés
- Kérdésbank
  - Kérdések kezelése (hozzáadás, törlés, módosítás …)
  - Kérdések lekérdezhetősége

Kérdés

```
abstract class Question
{
    public string Id { get; }
    public virtual string Text { get; }
    public Question(string id, string text)
    {
        Id = id;
        Text = text;
    }

    public abstract bool CheckAnswer(string answer);
}
```

Nem tárol helyes választ, sem részleteket

A kérdés szövege virtuális, mert lehet, hogy valamelyik kérdés típus módosítani akar

Helyes válasz nélkül nehéz lenne ellenőrizni

Igaz-hamis kérdés

```
class TrueFalseQuestion : Question
{
    private bool _correctAnswer;
    public TrueFalseQuestion(string id, string text, bool correctAnswer):
        base(id, text)
    {
        _correctAnswer = correctAnswer;
    }
    public override bool CheckAnswer(string answer)
    {
        return bool.Parse(answer) == _correctAnswer;
    }
}
```

Szám válaszos kérdés

```
class NumericalQuestion : Question
{
    private double _correctAnswer;
    public NumericalQuestion(string id, string text, double correctAnswer) :
        base(id, text)
    {
        _correctAnswer = correctAnswer;
    }
    public override bool CheckAnswer(string answer)
    {
        return double.Parse(answer) == _correctAnswer;
```

Feleletválasztós kérdés

```
class SingleChoiceQuestion : Question
{
    List<string> _options;
    int _correctOptionIndex;

    public SingleChoiceQuestion(string id, string text, List<string> options,
int correctOptionIndex):
        base(id, text)
    {
        _options = options;
        _correctOptionIndex = correctOptionIndex;
    }

    public override bool CheckAnswer(string answer)
    {
        return int.Parse(answer) - 1 == _correctOptionIndex;
    }
    …
}
```

Ez a helyes válasz indexe a listában

Feleletválasztós kérdés

```
class SingleChoiceQuestion : Question
{
    …
    public override string Text
    {
        get
        {
            string questionText = base.Text + "\n";
            for (int i = 0; i < _options.Count; i++)
            {
                questionText += $"  {i + 1}. {_options[i]}\n";
            }
            return questionText;
        }
    }
}
```

A kérdés szövegével együtt meg kell jelennie a választási lehetőségeknek is, így felülírjuk a virtuális Text property-t

Kérdés sor

```
class Quiz
{
    List<Question> _questions = new List<Question>();

    public void AddQuestion(Question question)
    {
        _questions.Add(question);
    }

    public int NumberOfQuestions()
    {
        return _questions.Count;
    }
    …
}
```

A kérdéseket általánosan az ősosztályt használva tároljuk

A hozzáadás során bármelyik gyerekosztályt elfogadja, és hozzáadja

Kérdés sor

```
class Quiz
{
    …
    public int Attempt()
    {
        int correctAnswers = 0;
        foreach (Question question in _questions)
        {
            Console.WriteLine(question.Text);
            string userAnswer = Console.ReadLine();
            bool isCorrect = question.CheckAnswer(userAnswer);
            if (isCorrect) { correctAnswers++; }
        }
        return correctAnswers;
    }
}
```

A Text és a CheckAnswer is virtuális, így a megfelelő gyerekosztály metódusa fog meghívódni

Kérdés bank

```
class QuestionBank
{
    List<Question> _questions = new List<Question>();

    public Question GetQuestionById(string id)
    {
        foreach (Question question in _questions)
        {
            if (question.Id == id) { return question; }
        }
        return null;
    }

    …
}
```

A tárolás itt is ősosztályként történik

A kérdésbank nem használ semmit a Question-ből, ami virtuális

Kérdés bank

```
class QuestionBank
{
    …
    public bool AddQuestion(Question newQuestion)
    {
        foreach (Question question in _questions)
        {
            if (question.Id == newQuestion.Id)
            {
                Console.WriteLine($"Question ID \"{newQuestion.Id}\" already
exists in the database!");
                return false;
            }
        }
        _questions.Add(newQuestion);
        return true;
    }
}
```

Alkalmazás

Ezt persze jobb lenne fájlból betölteni, de ahhoz meg kellene határozni egy formátumot is

```
QuestionBank questionBank = new QuestionBank();

questionBank.AddQuestion(new TrueFalseQuestion("class-object 1", "Are class and object the same?", false));

questionBank.AddQuestion(new SingleChoiceQuestion("virtual keyword", "What is the correct keyword to define a virtual method?", new List<string> { "new", "abstract", "virtual", "override" }, 2));

questionBank.AddQuestion(new NumericalQuestion("multiply 1", "What is 2.5*4.2?", 10.5));

questionBank.AddQuestion(new NumericalQuestion("the answer", "What is the answer to Life, the Universe, and Everything?", 42));

questionBank.AddQuestion(new TrueFalseQuestion("do-you", "Do you understand programming?", true));

questionBank.AddQuestion(new NumericalQuestion("multiply 2", "What is 9.9*10?", 99));
```

Alkalmazás

```
Quiz quiz = new Quiz();

AddQuestionToQuizById(quiz, questionBank, "class-object 1");
AddQuestionToQuizById(quiz, questionBank, "multiply 2");
AddQuestionToQuizById(quiz, questionBank, "virtual keyword");

int successCount = quiz.Attempt();

Console.WriteLine($"You have successfully answered {successCount} out of
{quiz.NumberOfQuestions()} questions.");
```

```
static void AddQuestionToQuizById(Quiz quiz, QuestionBank questionBank,
string questionId)
{
    Question question = questionBank.GetQuestionById(questionId);
    if (question == null) { Console.WriteLine($"Question with id
\"{questionId}\" does not exist"); }
    else { quiz.AddQuestion(question); }
}
```

Segéd metódus a kérdések kiválogatásához és kérdés sorhoz rendeléséhez

Normális esetben ez a UI-hoz köthető