Турсунов Баходурхон

#15. .NET Object-Oriented Programming (OOP). Primary Constructors. Static Constructors.

**1. Можно ли объявлять первичные параметры конструктора как ref или out? Если нет, то почему?**

Параметры конструкторов не могут быть объявлены с использованием ключевых слова ref и out. Это связано с тем, что при создании экземпляра объекта, конструктор вызывается для инициализации полей и свойств объекта до того, как сам объект будет создан. Эти ключевые слова предназначены для передачи аргументов методам, и они предполагают, что переменные, переданные по ссылке или как out-параметры, уже инициализированы перед передачей в метод. В конструкторе создается объект, и ему предоставляются значения для инициализации его членов. Поэтому использование ref или out в контексте конструктора не имеет смысла, так как конструктор создает объект и инициализирует его члены, а не изменят значения существующих переменных.

**2. Когда вызывается статический конструктор и как он связан с конструкторами экземпляров?**

Статический конструктор используется для инициализации любых статических данных или для выполнения определенного действия, которое требуется выполнить только один раз. Он вызывается автоматически перед созданием первого экземпляра или ссылкой на какие-либо статические члены.

Конструкторы экземпляров, с другой стороны, вызываются при создании нового объекта этого класса с использованием оператора **new.** Эти конструкторы инициализируют экземпляр класса и его члены. Каждый раз, когда создается новый объект, вызывается конструктор экземпляра для этого объекта.

**3. Обсудите ограничения на типы параметров для первичных конструкторов в C# и причины их существования.**

Первичные конструкторы (primary constructors) вводят ограничения на типы параметров по сравнению с обычными конструкторами. Первичные конструкторы предоставляют краткую форму для инициализации свойств класса, и их цель – сделать код более лаконичным. Однако, из-за ограничений на типы параметров, они могут оказаться неудобными в некоторых ситуациях.

В первичных конструкторах разрешены только следующие типы параметров:

* Простые типы данных (int, double, char, etc.)
* Строки
* Перечисления

Ограничения на типы параметров связано с тем, что первичные конструкторы преобразуют параметры в поля класса, и не все типы данных могут быть преобразованы в поля с использованием этого механизма. Например, сложные типы данных, такие как массивы, списки, словари, и пользовательские классы, не могут быть использованы в первичных конструкторах.

Practice:

1. Создайте класс C# с именем "Студент" со следующими атрибутами: Имя (строка), Возраст (int), Средний балл (double).

2. Реализуйте первичный конструктор для класса "Студент", принимающий в качестве параметров все три атрибута.

3. Создайте экземпляр класса "Студент" с помощью первичного конструктора и инициализируйте его данными из примера.

4. Реализуйте статический конструктор для класса "Студент", который при инициализации класса устанавливает статическое поле "TotalStudents" в 0.

5. Создайте несколько экземпляров класса "Студент", используя первичный конструктор, и увеличивайте значение "TotalStudents" в статическом конструкторе каждый раз, когда создается новый экземпляр.

6. Напишите метод в классе "Студент", который выводит данные о студенте (имя, возраст, средний балл) и общее количество студентов (TotalStudents) с помощью статического метода.

1. namespace HW15\_1

2. {

3. class Student

4. {

5. public string Name { get; set; }

6. public int Age { get; set; }

7. public double AverageScore { get; set; }

8. private static int TotalStudents { get; set; }

9.

10. public Student(string name, int age, double averageScore)

11. {

12. Name = name;

13. Age = age;

14. AverageScore = averageScore;

15. TotalStudents++;

16. }

17.

18. static Student()

19. {

20. TotalStudents = 0;

21. }

22.

23. public static void PrintStudentInfo(Student student)

24. {

25. Console.WriteLine($"Name: {student.Name}, Age: {student.Age}, Average Score: {student.AverageScore}\n");

26. }

27. public static void PrintTotalStudent()

28. {

29. Console.WriteLine($"Total Students: {TotalStudents}");

30. }

31. }

32.

33. class Program

34. {

35. static void Main(string[] args)

36. {

37. Student student1 = new("John", 21, 70.3);

38. Student.PrintStudentInfo(student1);

39.

40. Student student2 = new("Max", 22, 60.4);

41. Student.PrintStudentInfo(student2);

42.

43. Student student3 = new("CJ", 19, 70.6);

44. Student.PrintStudentInfo(student3);

45.

46. Student.PrintTotalStudent();

47. }

48. }

49. }

50.

Здесь в программе, если мы не создадим ни одного студента, будет вызван метод PrintTotalStudent и на консоли будет 0 студентов