Armen Melkumyan • 1st
Technical / Solutions Architect
1yr • Edited •

From C# interview questions:

Please explain records in context of C#

A record in C# is a reference type that provides a simplified syntax for creating immutable objects. Records are useful for representing data structures that are primarily intended to store values with little to no behavior. C# 9 introduced records to provide a concise way to define immutable types without needing to write boilerplate code for operations like value equality, copying, and printing.

Here are some key characteristics of record in C#:

Immutability:

Records are immutable by default, which means their property values cannot be changed once they are set.

Value-based Equality:

Records override the default reference-based equality behavior and provide value-based equality. Two records are equal if their types and property values are equal.

With-Expression:

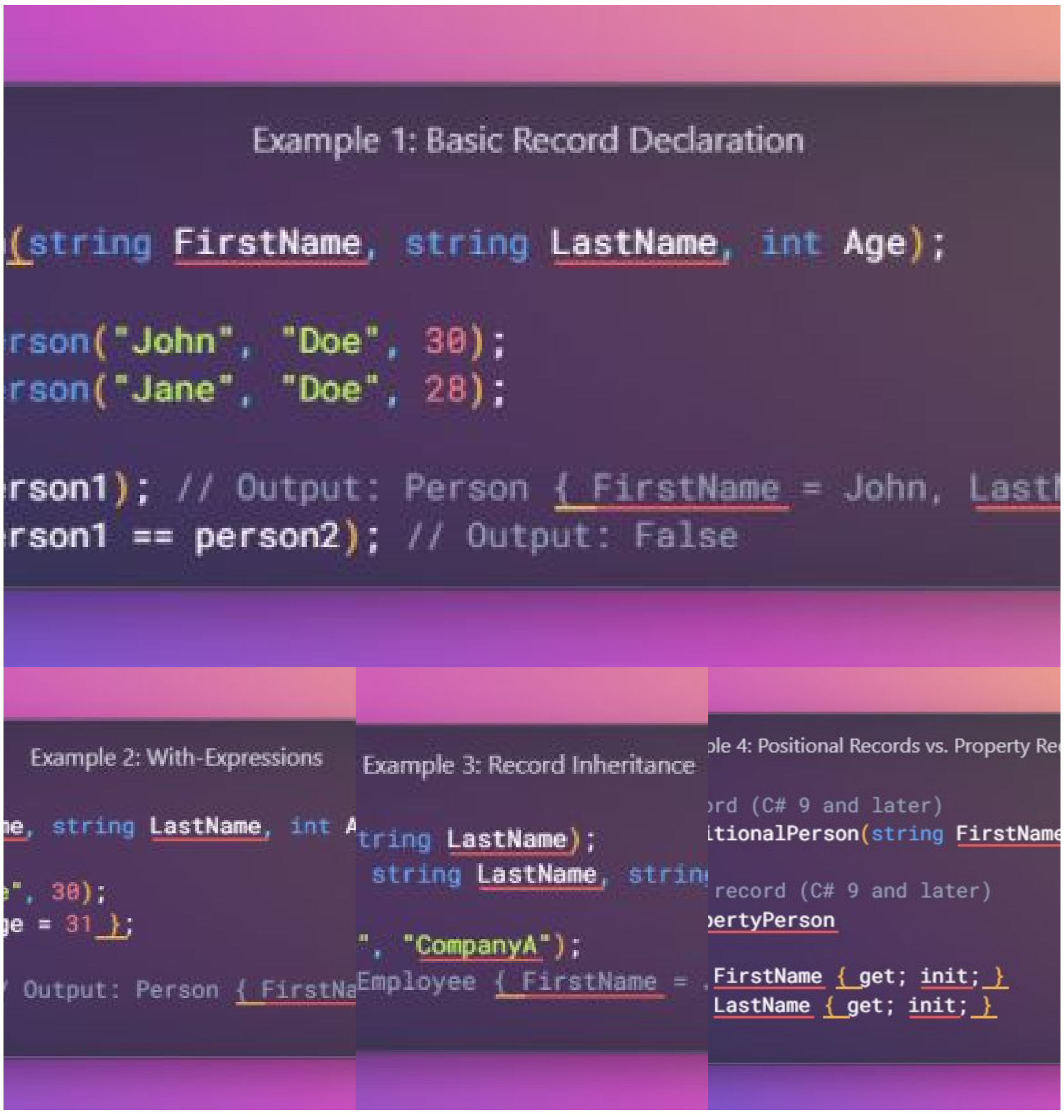
With-expressions provide a way to create a copy of a record with some of its properties changed.

Synthesized Members:

Records automatically generate methods for equality checks, hashing, and printing.

Let's look at some code examples to illustrate how to use records in C#.

#interviewquestions



Example 1: Basic Record Declaration

```
(string FirstName, string LastName, int Age);

rson("John", "Doe", 30);
rson("Jane", "Doe", 28);

rson1); // Output: Person { FirstName = John, Last
rson1 == person2); // Output: False
```

Example 2: With-Expressions

```
ne, string LastName, int A

e", 30);
ge = 31 };

/ Output: Person { FirstNa
```

Example 3: Record Inheritance

```
tring LastName);
 string LastName, strin

", "CompanyA");
Employee { FirstName = 
```

ple 4: Positional Records vs. Property Re

```
ord (C# 9 and later)
itionalPerson(string FirstName

 record (C# 9 and later)
pertyPerson

 FirstName { get; init; }
 LastName { get; init; }
```

8