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

🔍💡 C# .Net Interview Puzzle: Understanding Closures

🤔 What's the Output and Why?

```
for (int i = 0; i < 3; i++)
{
delegates[i] = delegate { Console.WriteLine(i); };
}

foreach (var printDelegate in delegates)
{
printDelegate();
}
```

Diving Deep into Closures in C#:

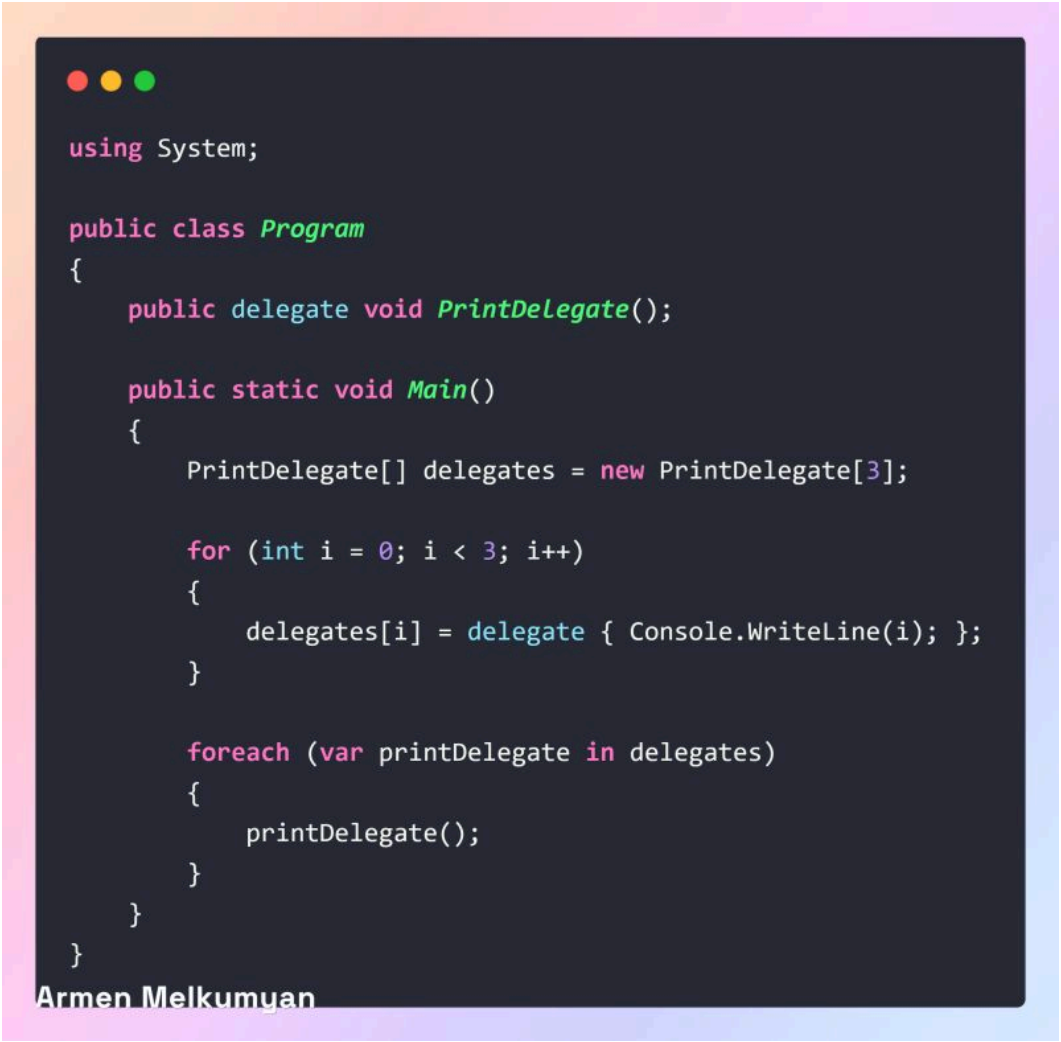
Closures in C# might behave differently than you first expect! When looping and capturing a variable like i in a delegate, C# doesn't capture the variable's value at each iteration. Instead, it captures a reference to i.

🔁 The Loop Mystery:

After looping, all delegates reference the same i — which ends up being 3 when the loop is done. So, when each delegate is finally invoked, they all print i's value at that time: 3.

🎯 Key Takeaway:

This illustrates a crucial aspect of how closures capture variables in C#, leading to outputs that might surprise you!


using System;
public class Program
{
public delegate void PrintDelegate();
public static void Main()
{
PrintDelegate[] delegates = new PrintDelegate[3];
for (int i = 0; i < 3; i++)
{
delegates[i] = delegate { Console.WriteLine(i); };
}
foreach (var printDelegate in delegates)
{
printDelegate();
}
}
}
Armen Melkumyan

36 1 repost