Using yield in switch expression

In Java, the yield keyword is used in the context of concurrency and thread scheduling, specifically within a switch statement in a switch expression. It's used to yield control to other threads or to allow the scheduler to determine which thread should run next.
Here's how you can use yield:

code

```
public class Example
{
	public static void main(String[] args)
	{
		int number = 2;
		int result = switch(number)
		{
			case 1 -> { System.out.println("Case 1"); yield 10; }
			case 2 -> { System.out.println("Case 2"); yield 20; }
			default ->
			{
				System.out.println("Default case"); yield 30;
			}
		};
		System.out.println("Result: " + result);
	}
}
```

In this example, when number is 2, the case 2 is executed, printing "Case 2" and yielding the value 20. The value 20 is then assigned to result, and the program continues executing after the switch expression.
It's important to note that the use of yield in this context is only valid within a switch expression, and it's used for thread scheduling and not for returning values from methods like return. The yield statement allows a thread to voluntarily relinquish control to other threads with equal priority, or to allow threads with higher priority to execute. It's typically used in cooperative multitasking scenarios where threads cooperate to share resources and execution time.