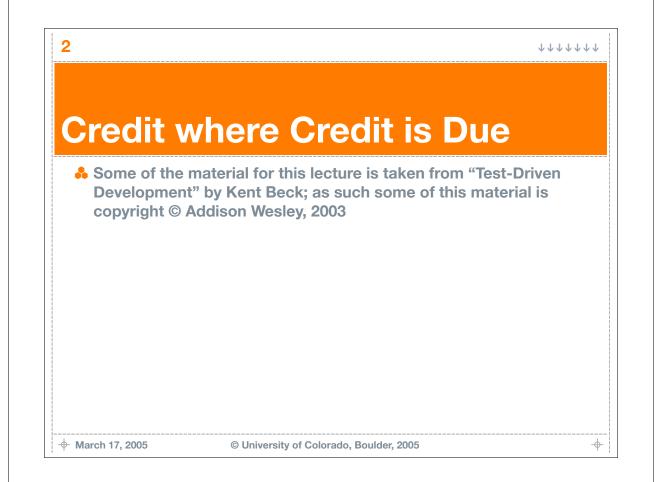
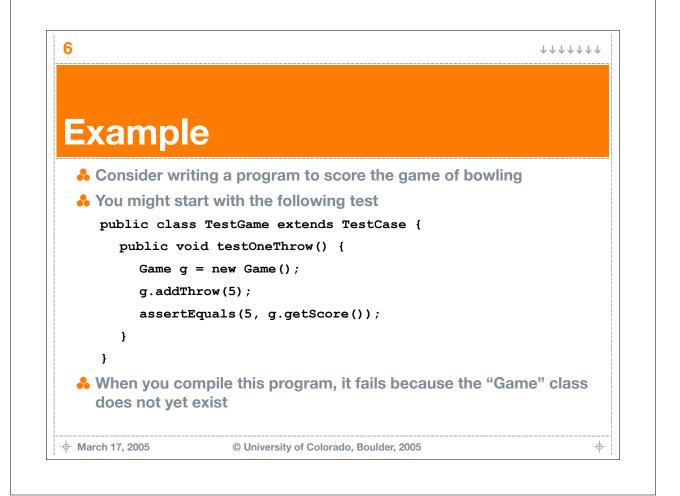
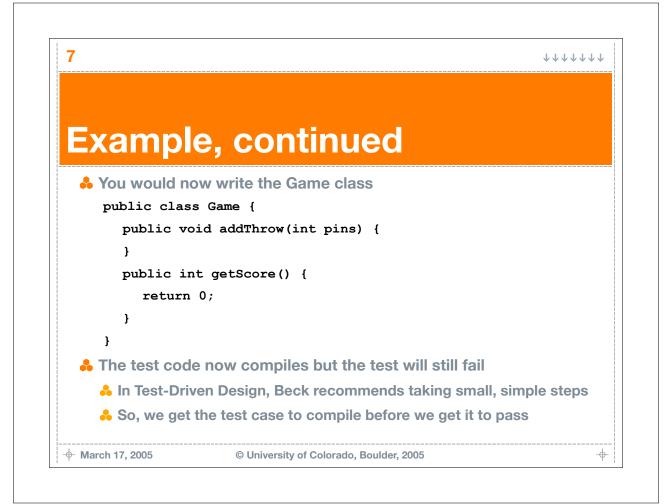
1	$\uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \uparrow \downarrow$
Lecture 20: Test-Driver	۱
	1
Development	)
Development Kenneth M. Anderson	<b>ו</b>

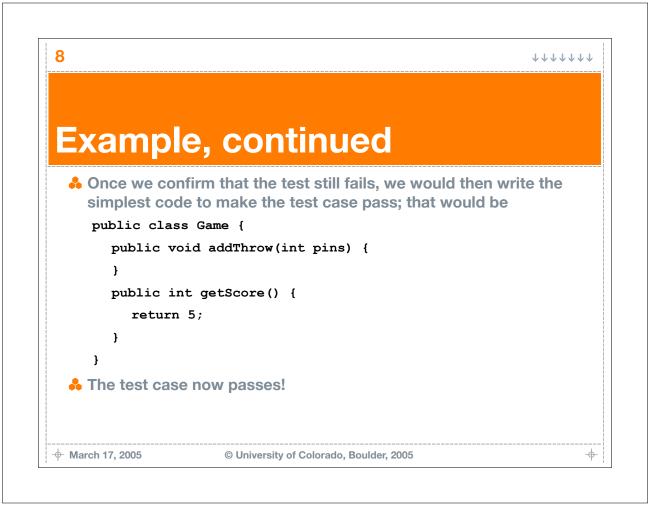


<ul><li>Introduce the</li><li>Present an ex</li></ul>	e concept of Test-Driven Developmei xample	nt (TDD)
♦ March 17, 2005	© University of Colorado, Boulder, 2005	-\$
4		$\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow\uparrow$
	iven Developme	

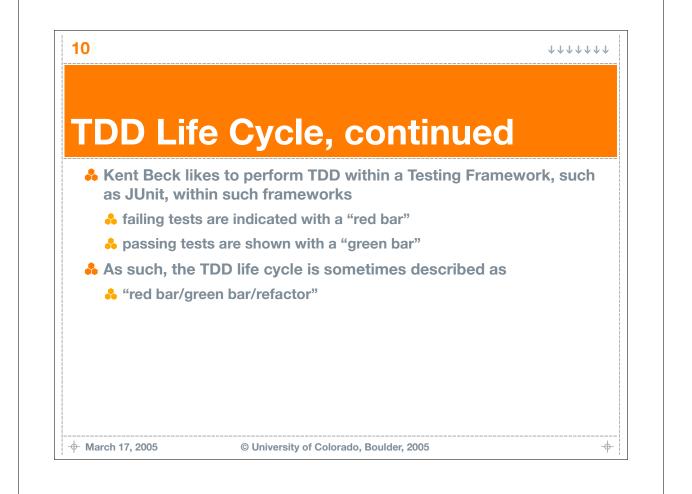
Writing	J Test Case	es First
	s that when you first wri loes not exist	te a test case, you may be testin
And since case "fai		e will not compile, obviously the tes
-	write the skeleton code f vill now compile, but also	for the objects referenced in the tes may not pass
So, then pass	you write the simplest co	de that will then make the test case

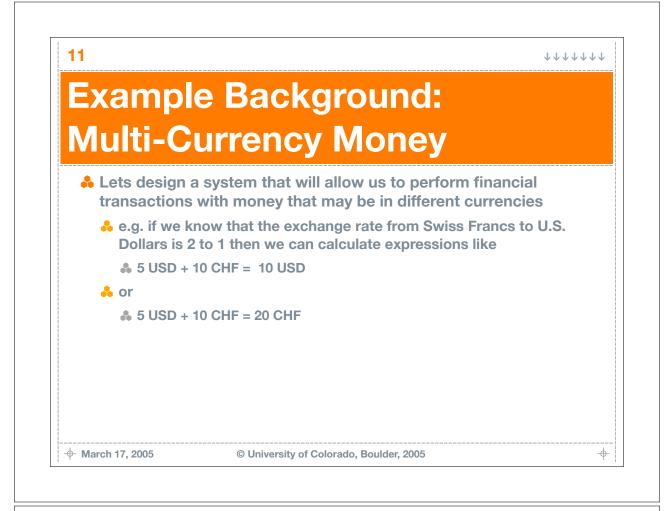


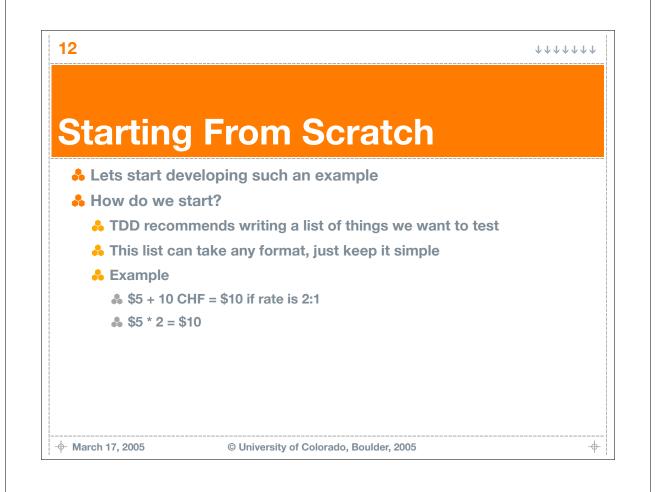




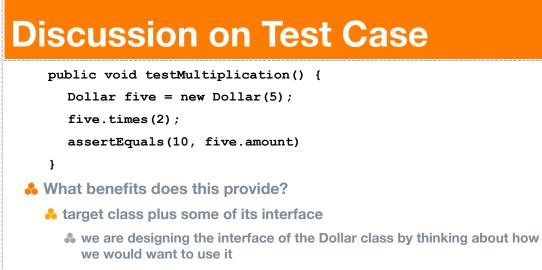
Γ	DD Life Cycle
•	The life cycle of test-driven development is
	👶 Quickly add a test
	Run all tests and see the new one fail
	👃 Make a simple change
	Run all tests and see them all pass
	Refactor to remove duplication
*	This cycle is followed until you have met your goal; note that this cycle simply adds testing to the "add functionality; refactor" loop of refactoring covered in the last two lectures
- M	arch 17, 2005 © University of Colorado, Boulder, 2005 $\diamond$







The first test 5 USD * 2 =	case looks a bit complex, lets start with t	ne second
Å First, we writ	e a test case	
public void	<pre>testMultiplication() {</pre>	
Dollar fi	ve = new Dollar(5);	
five.time		
_	als(10, five.amount)	
}		
→ March 17, 2005	© University of Colorado, Boulder, 2005	 -



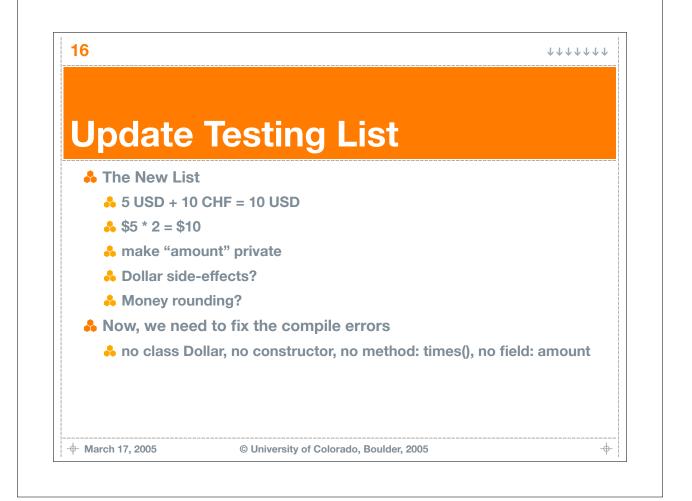
We have made a testable assertion about the state of that class after we perform a particular sequence of operations

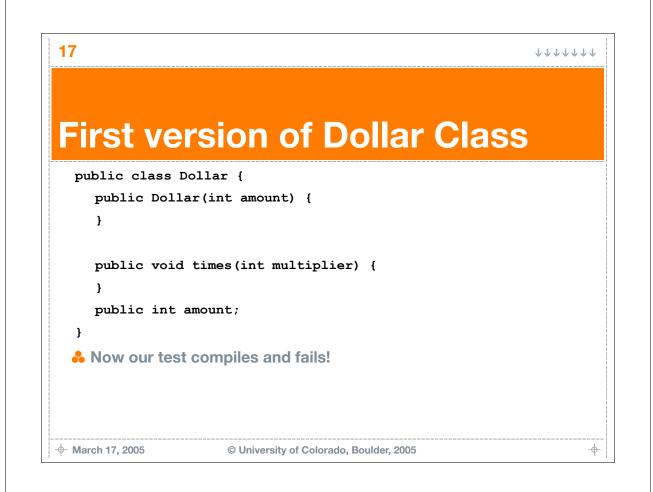
+ March 17, 2005

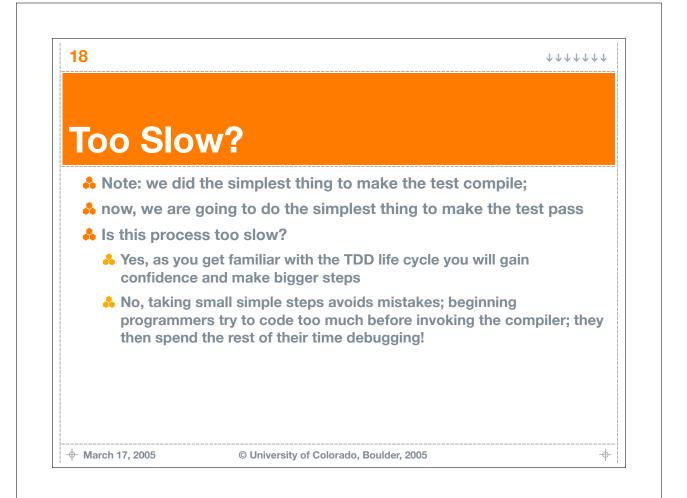
© University of Colorado, Boulder, 2005

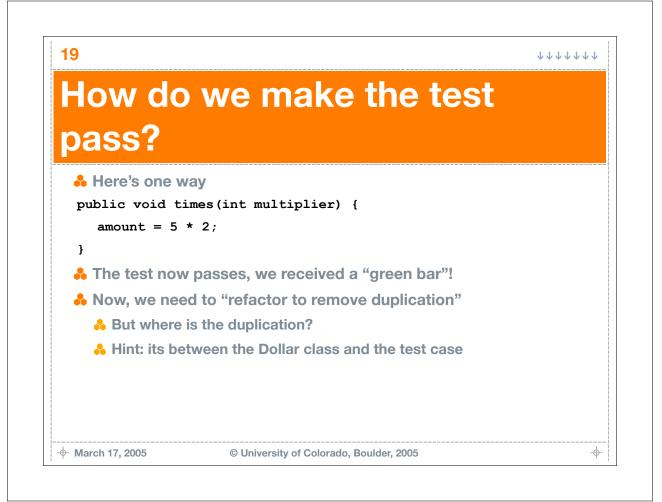
-

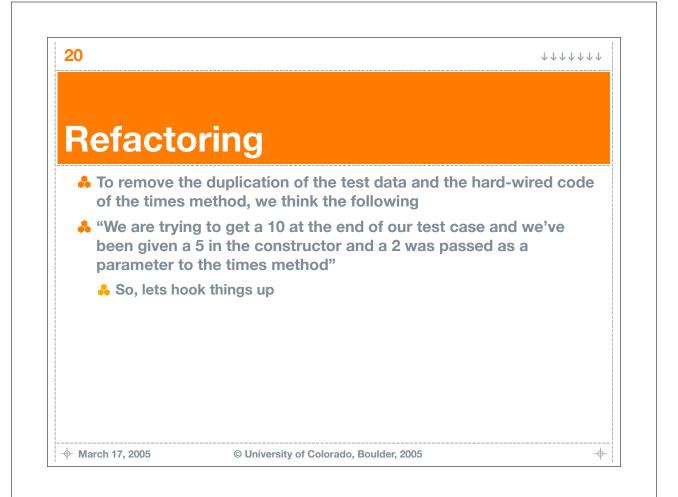
Vr	nat's Next?
🔒 W	e need to update our test list
*	The test case revealed some things about Dollar that we will want to clean up
	We are representing the amount as an integer, which will make it difficult to represent values like 1.5 USD; how will we handle rounding of factional amounts?
	Dollar.amount is public; violates encapsulation
	What about side effects?; we first declared our variable as "five" but after we performed the multiplication it now equals "ten"

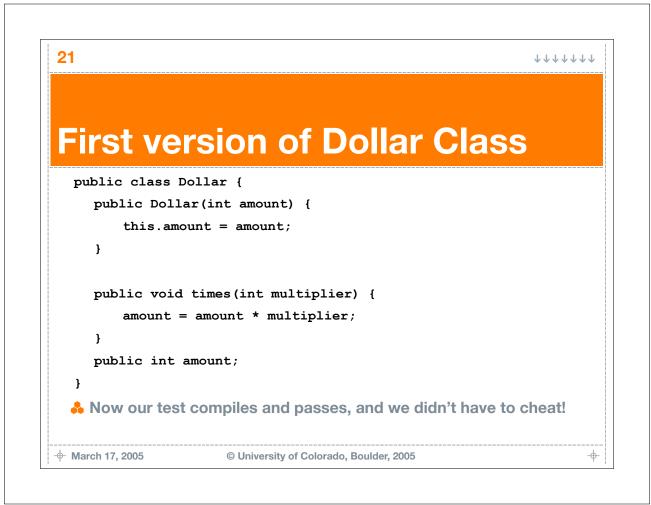


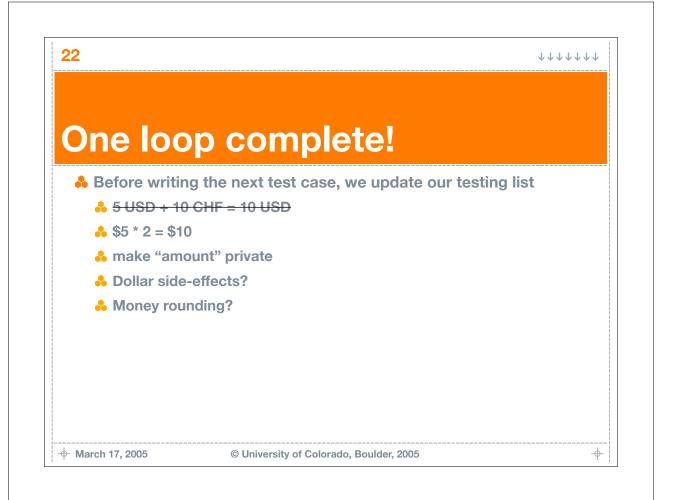




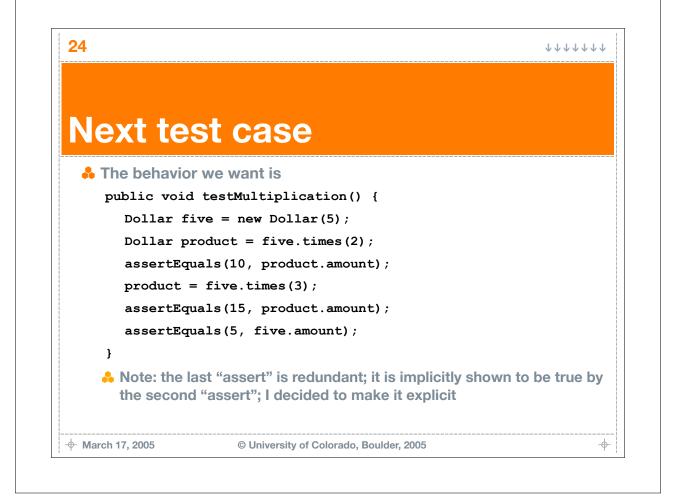




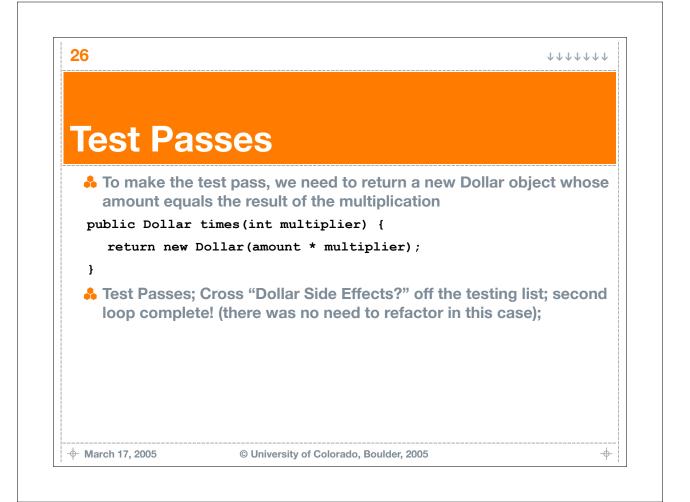




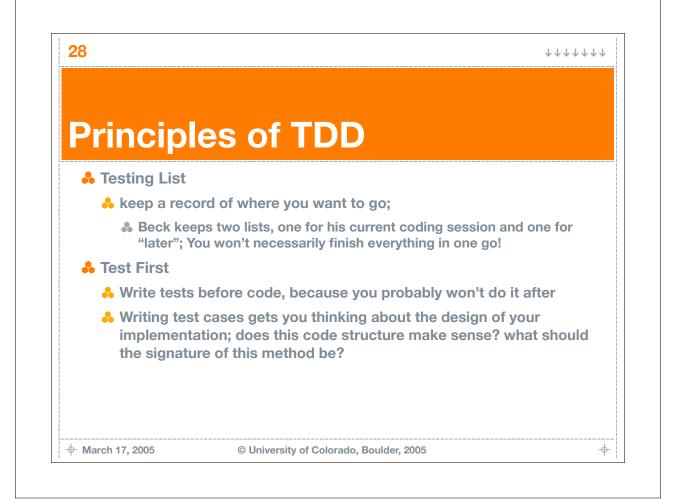
ne mo	ore example
Lets addres general less	s the "Dollar Side-Effects" item and then move on to sons
So, lets writ	e the next test case
	called the times operation our variable "five" was pointing a whose amount equaled "ten"; not good
	es operation had a side effect which was to change the value of a sly created "value object"
bill into	bout it, as much as you might like to, you can't change a 5 dollar a 500 dollar bill; the 5 dollar bill remains the same throughout financial transactions



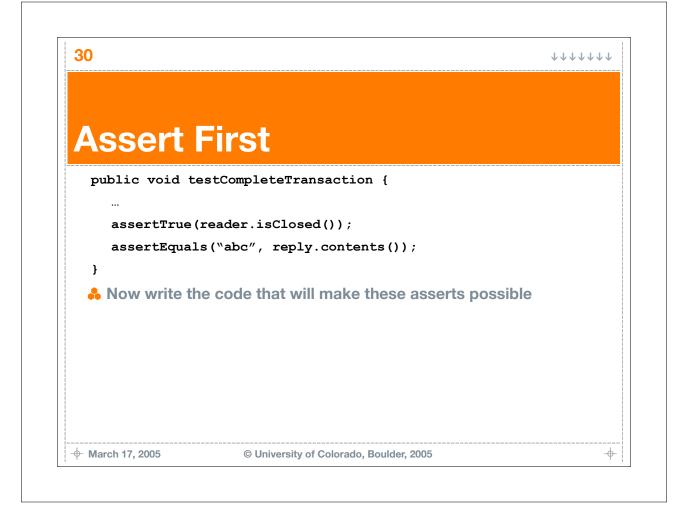
es	st fails
	e test fails because it won't compile;
	e need to change the signature of the times method; previously it urned void and now it needs to return Dollar
pu	blic Dollar times(int multiplier) {
	amount = amount * multiplier;
	return null;
}	
	e test compiles but still fails; as Kent Beck likes to say rogress!"



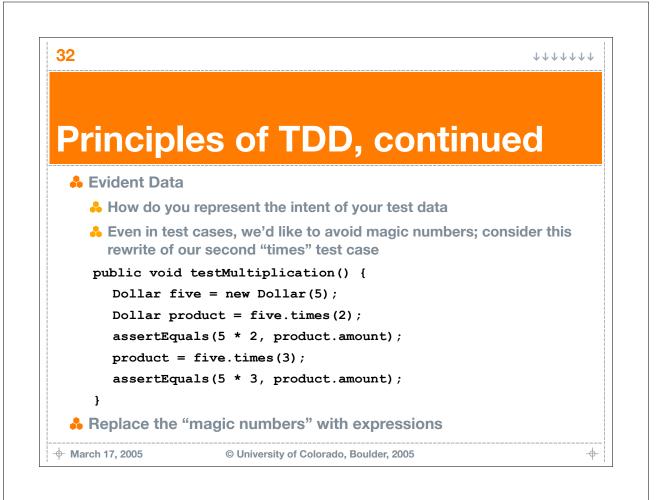
JISCUSS	sion of the	Example
There is still	a long way to go	
🙏 only scratc	hed the surface	
🔒 But		
🔒 we saw the	e life cycle performed twice	<u>h</u>
🔒 we saw the	e advantage of writing tests	s first
🔒 we saw the	e advantage of keeping thir	ıgs simple
we saw the progress	e advantage of keeping a te	esting list to keep track of our
because our	-	now if we are breaking things we do; if the old tests stay ce



rincipl	es of TDD, continued
Assert First	
🔒 How do yo	u write a test case?
🌲 By writin	g its assertions first!
	ou are writing a client/server system and you want to test ion between the server and the client
from the	that for each transaction, some string has to have been read server and that the socket used to talk to the server should be fter the transaction
🙏 Lets write	the test case







Summa	ry
	Design is a "mini" software development life cycle that Inize coding sessions and make them more productive
🔒 Write a faili	ng test case
🔒 Make the s	implest change to make it pass
🔒 Refactor to	remove duplication
🔒 Repeat!	

