

Answer Key

#	Answer
1	\$500.0. The <code>double balance</code> declared inside the <code>if</code> is a new local variable that hides the field. It gets \$600 and then disappears when the block ends. The field never changes.
2	It returns <code>225.0</code> (<code>100 → 150 → 225</code>). <code>a.getBalance()</code> is still <code>500.0</code> . The local balance starts at 100 and hides the field.
3	Change the first line to <code>double balance = getBalance();</code> . The local starts from the real balance, but it's still only a copy, so the field isn't changed. (Giving the local a different name, like <code>projected</code> , would make this clearer.)
4	OK. An <code>int</code> can be passed where a <code>double</code> is expected (100 becomes 100.0).
5	CE: the arguments are in the wrong order. A <code>double</code> can't be passed as the <code>String</code> .
6	CE: <code>"100"</code> is a <code>String</code> , not a number.
7	CE: no constructor takes three arguments.
8	CE: <code>new</code> is missing.
9	Only (ii) is a CE. Its signature (<code>String, double</code>) matches the existing constructor, and parameter names don't count. (i) and (iii) have new type lists. (iv) is legal too, because <code>int</code> is a different type from <code>double</code> . (Bonus: then <code>new BankAccount("Ada", 100)</code> would call (iv), because it's the exact match.)
10	Account #1 (null): \$0.00
11	0.0
12	RE: <code>NullPointerException</code> . <code>getOwner()</code> returns <code>null</code> .
13	Assigning a field to a parameter is legal. The constructor just copies the fields' default values (<code>null</code> and <code>0.0</code>) into its parameters, which then disappear.
14	CE: missing return statement. A non-void method must return a value.
15	Works.
16	Compiles but wrong. It returns the literal text <code>owner: \$balance</code> for every account.
17	CE in the client code: a <code>void</code> result can't be stored in a <code>String</code> . (By itself the method compiles and prints.)
18	Ada. It changes the parameter, not the field.
19	Ada. It returns a value that the client ignores, and nothing is assigned.
20	CE: <code>rename</code> is <code>private</code> , so the client can't call it.
21	Ava
22	Account #1 (Ada): \$0.00
23	<code>25.0</code> . Java passes a copy of <code>cash</code> , so setting <code>amount = 0.0</code> changes only the copy.
24	2. The "Nobody" account was really created, even though nothing kept it.
25	Ada Bo. Pointing a at a new object doesn't move b.
26	1 3
27	<code>acct</code> is a copy of the reference, so it points at Ada's real account. Calling <code>withdraw</code> through it changes that account. Assigning <code>acct = new ...</code> only re-points the local copy. The caller's a still points at Ada.
28	2 (only Ada and Cy were counted)
29	1. Ben gets the same number as Ada.
30	2. Dee gets the same number as Cy.
31	<code>true true</code> . Different accounts are "equal" because they share a number.
32	When code is duplicated, a fix or change to one copy can be missed in the other. Here, the account count is wrong, account numbers repeat, and <code>equals</code> says different accounts are the same.