

BankAccount, Part 2: Bugs and Traps

Name: _____

Date: _____

Each part below changes or uses the BankAccount class from homework. Unless a part says otherwise, assume the class is exactly as you know it and that no accounts exist before each part starts.

Part A: Which balance?

Suppose deposit were written like this instead:

```
public void deposit(double amount)
{
    if (amount > 0)
    {
        double balance = getBalance() + amount;
    }
}
```

1. Account a has \$500. After `a.deposit(100.0)`, what is `a.getBalance()`? Explain what happened to the \$100.

Now suppose this method is added to BankAccount, and `BankAccount.setInterestRate(0.5)` has been called.

```
public double projectedBalance(int years)
{
    double balance = 100.0;
    for (int i = 0; i < years; i++)
    {
        balance += balance * interestRate;
    }
    return balance;
}
```

2. Account a still has \$500. What does `a.projectedBalance(2)` return? What is `a.getBalance()` afterward?
3. Change one line so that `projectedBalance` projects from the account's real balance without changing it.

Part B: Will it construct?

BankAccount has two constructors: `BankAccount(String owner, double startingBalance)` and `BankAccount(String owner)`. For each statement, write **OK** or **CE** with a reason.

#	Statement	OK or CE (why?)
4	<code>BankAccount x = new BankAccount("Ada", 100);</code>	
5	<code>BankAccount x = new BankAccount(100.0, "Ada");</code>	
6	<code>BankAccount x = new BankAccount("Ada", "100");</code>	
7	<code>BankAccount x = new BankAccount("Ada", 100.0, 5);</code>	
8	<code>BankAccount x = BankAccount("Ada");</code>	

Each constructor below is added to BankAccount by itself. Which ones cause a compile error? Circle them and explain.

- (i) public BankAccount(double startingBalance)
- (ii) public BankAccount(String nickname, double money)
- (iii) public BankAccount(double startingBalance, String owner)
- (iv) public BankAccount(String owner, int startingBalance)

9. Your answer and explanation for (i)–(iv):

Part C: Backwards

A student wrote the two-parameter constructor like this. (The rest of the class is unchanged.)

```
public BankAccount(String owner, double startingBalance)
{
    owner = this.owner;
    startingBalance = balance;
    numAccounts++;
    accountNumber = numAccounts;
}
```

What does this code print? If a line causes a runtime error, write RE and the exception name.

```
BankAccount a = new BankAccount("Ada", 50.0);
System.out.println(a); // line 1
System.out.println(a.getBalance()); // line 2
System.out.println(a.getOwner().length()); // line 3
```

10. Line 1:

11. Line 2:

12. Line 3:

13. Why didn't this constructor cause a compile error, even though it never sets owner or balance?

Part D: Return, print, or neither?

A summary method should let a client write String s = a.summary(); and get back something like Ada: \$80.0. For each version, write **works**, **compiles but wrong** (and what goes wrong), or **CE** (and why).

#	Version	Your answer
14	public String summary() { System.out.println(owner + ": \$" + balance); }	
15	public String summary() { return owner + ": \$" + balance; }	
16	public String summary() { return "owner: \$balance"; }	

#	Version	Your answer
17	public void summary() { System.out.println(owner + ": \$" + balance); }	

Each rename below is added to BankAccount. A client runs this code. What is printed? Write CE if it doesn't compile.

```
BankAccount a = new BankAccount("Ada", 80.0);
a.rename("Ava");
System.out.println(a.getOwner());
```

#	Version	Printed
18	public void rename(String newOwner) { newOwner = owner; }	
19	public String rename(String newOwner) { return newOwner; }	
20	private void rename(String newOwner) { owner = newOwner; }	
21	public void rename(String newOwner) { owner = newOwner; }	

Part E: What does a method really get?

This method is in a class called Bank. The code below it is in Bank's main.

```
public static void reset(BankAccount acct, double amount)
{
    amount = 0.0;
    acct.withdraw(acct.getBalance());
    acct = new BankAccount("Nobody", 999.0);
}

BankAccount a = new BankAccount("Ada", 80.0);
double cash = 25.0;
reset(a, cash);
System.out.println(a); // line 1
System.out.println(cash); // line 2
System.out.println(BankAccount.getNumAccounts()); // line 3
BankAccount b = a;
a = new BankAccount("Bo", 10.0);
System.out.println(b.getOwner() + " " + a.getOwner()); // line 4
System.out.println(b.getAccountNumber() + " " + a.getAccountNumber()); // line 5
```

22. Line 1:

23. Line 2:

24. Line 3:

25. Line 4:

26. Line 5:

27. reset emptied Ada's money, but the line `acct = new BankAccount(...)` didn't replace Ada's account. Explain the difference.

Part F: The forgotten line

Someone deletes `numAccounts++`; from the **one-parameter** constructor only. Everything else stays the same, including `accountNumber = numAccounts`;

```
BankAccount a = new BankAccount("Ada", 10.0);  
BankAccount b = new BankAccount("Ben");  
BankAccount c = new BankAccount("Cy", 5.0);  
BankAccount d = new BankAccount("Dee");  
System.out.println(BankAccount.getNumAccounts());           // line 1  
System.out.println(b.getAccountNumber());                   // line 2  
System.out.println(d.getAccountNumber());                   // line 3  
System.out.println(a.equals(b) + " " + c.equals(d));         // line 4
```

28. Line 1:

29. Line 2:

30. Line 3:

31. Line 4:

32. Why is having two copies of the same constructor code risky? (Hint: compare lines 2–4 to what a bank would want.)