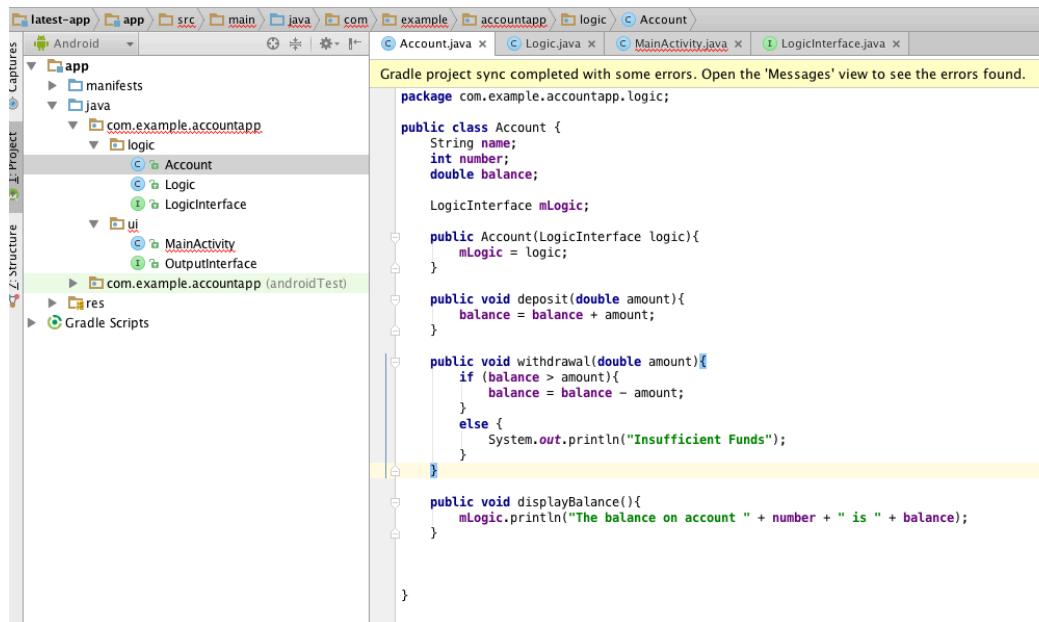


# Open Android Studio and follow along



# an Account object

Account  
data:

Lois Bidder  
1567083  
784.56

Account methods:

Balance?  
Deposit  
Withdrawal

# the Account class

Field name	Description
name	Account holder's first and last name
number	A unique identification number
balance	Amount currently in account

Method name	Description
displayBalance ()	returns the current balance
deposit (x)	add x to the current balance
withdrawal (x)	subtract x from the current balance

## The class or “blueprint”

state		behavior
String	name	displayBalance()
int	number	deposit(x)
double	balance	withdrawal(x)

↓

state		behavior	state	behavior	state	behavior
name	Lois Bidder	displayBalance()	number	deposit(x)	balance	withdrawal(x)
number	1567083	deposit(x)	number	2890471	balance	Turner Luce
balance	784.56	withdrawal(x)	balance	190.00	balance	displayBalance()
					number	8734921
					balance	deposit(x)
					balance	withdrawal(x)

↓

The objects or instances

# an Account object

Account  
data:

Lois Bidder  
1567083  
784.56

Account methods:

Balance?  
Deposit  
Withdrawal

# class data– the state of an object

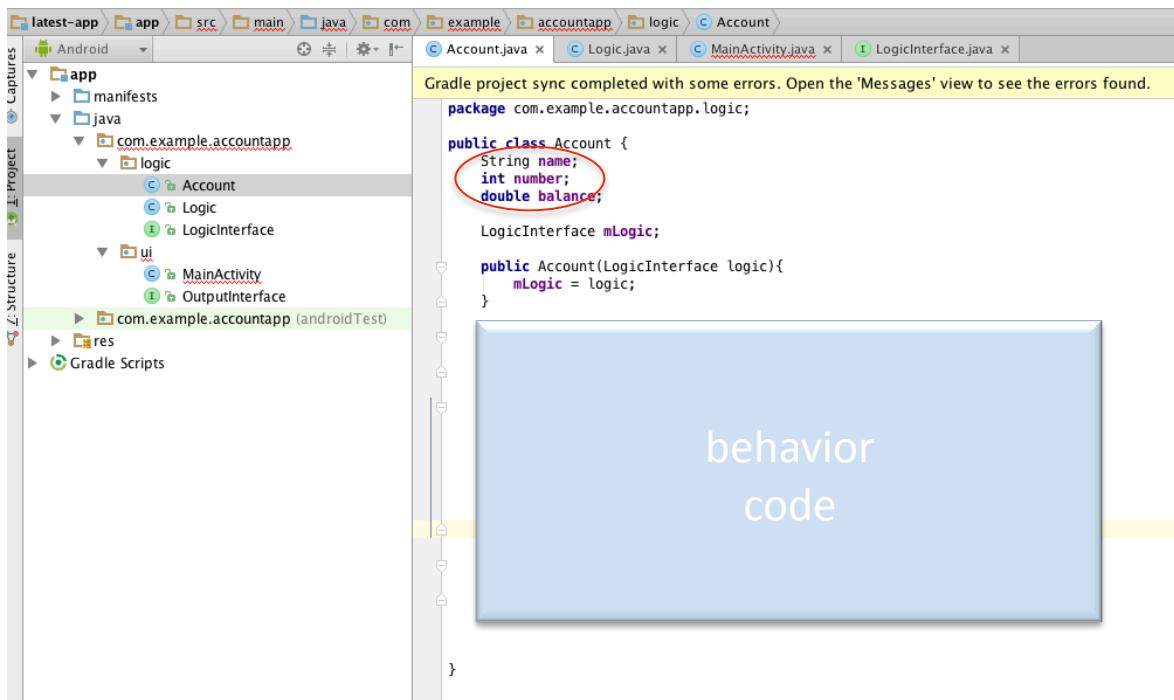
- The data portion of a class (the blueprint) or an object (the instance of a class) is sometimes called the state
- Each piece of data is called a field or *instance variable*
- Collectively, these fields store the state of an object

# Build the Account class

1. Open Account.java
2. Create the fields or instance variables for this class

```
String name;  
int number;  
double balance;
```

# Build the Account class



# an Account object

Account  
data:

Lois Bidder  
1567083  
784.56

Account methods:

Balance?  
Deposit  
Withdrawal

# Build the Account class

Create the behaviors or methods

- `displayBalance()`
  - prints the Account number and the current balance
- `deposit(double amt)`
  - input parameter double
  - increases balance by the amount deposited

# Build the Account class

The screenshot shows a Java code editor with the following code:

```
package com.example.accountapp.logic;

public class Account {
    String name;
    int number;
    double balance;

    LogicInterface mLogic;

    public Account(LogicInterface logic){
        mLogic = logic;
    }

    public void deposit(double amount){
        balance = balance + amount;
    }

    public void withdrawal(){
        // Implementation of withdrawal
    }

    public void displayBalance(){
        mLogic.println("The balance on account " + number + " is " + balance);
    }
}
```

Annotations in the code:

- A blue arrow points to the first line of the class definition: `public class Account {`.
- A red arrow points to the `deposit` method: `public void deposit(double amount){`.
- A light blue rounded rectangle highlights the `withdrawal()` method.
- A red arrow points to the `displayBalance` method: `public void displayBalance(){`.

# Build the Account class

- `withdrawal(double amt)`
  - input parameter double
  - decreases balance by the amount withdrawn
  - what checks should be made?
  - what feedback should we give the user?

# Think about Client classes

- As the client classes

ABSTRACTION

- Keep program users



Often you are both the author of the class and the author of the Client class(or user of the class)

You should consider both roles separately.



# Build the Account class

What kind of information would other programmers need to know after a withdrawal has been attempted?



# Build the Account class

```
package com.example.accountapp.logic;

public class Account {
    String name;
    int number;
    double balance;

    LogicInterface mLogic;

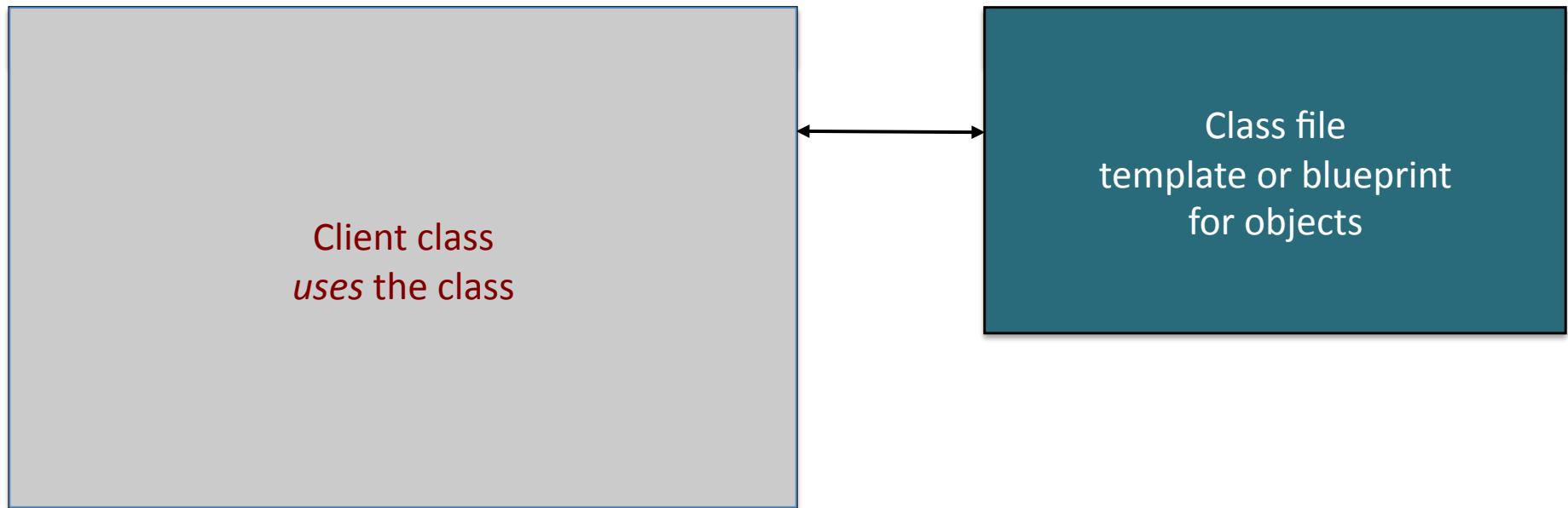
    public Account(LogicInterface logic){
        mLogic = logic;
    }

    public void deposit(double amount){
        balance = balance + amount;
    }

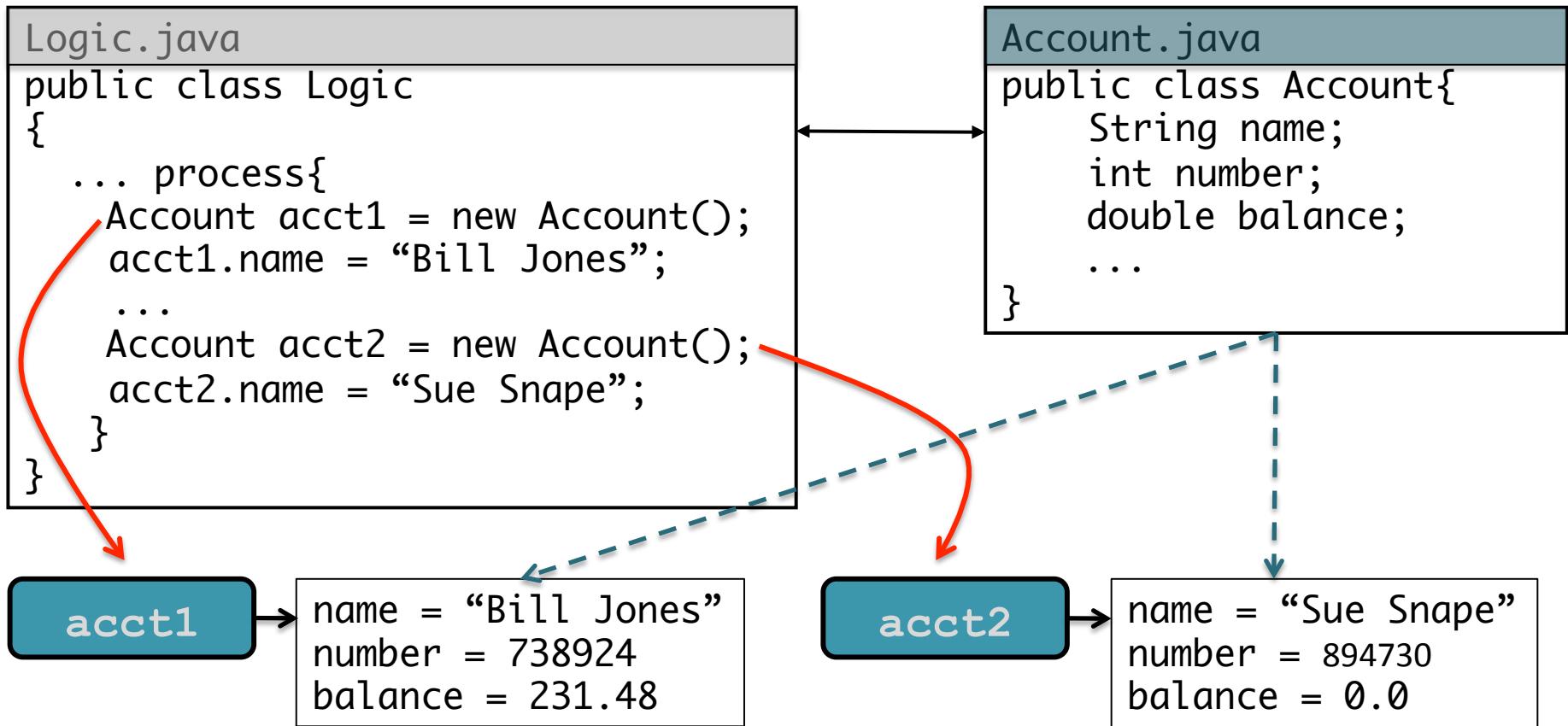
    public boolean withdrawal(double amount){
        if (balance > amount){
            balance = balance - amount;
            return (true);
        }
        else
            return(false);
    }

    public void displayBalance(){
        mLogic.println("The balance on account " + number + " is " + balance);
    }
}
```

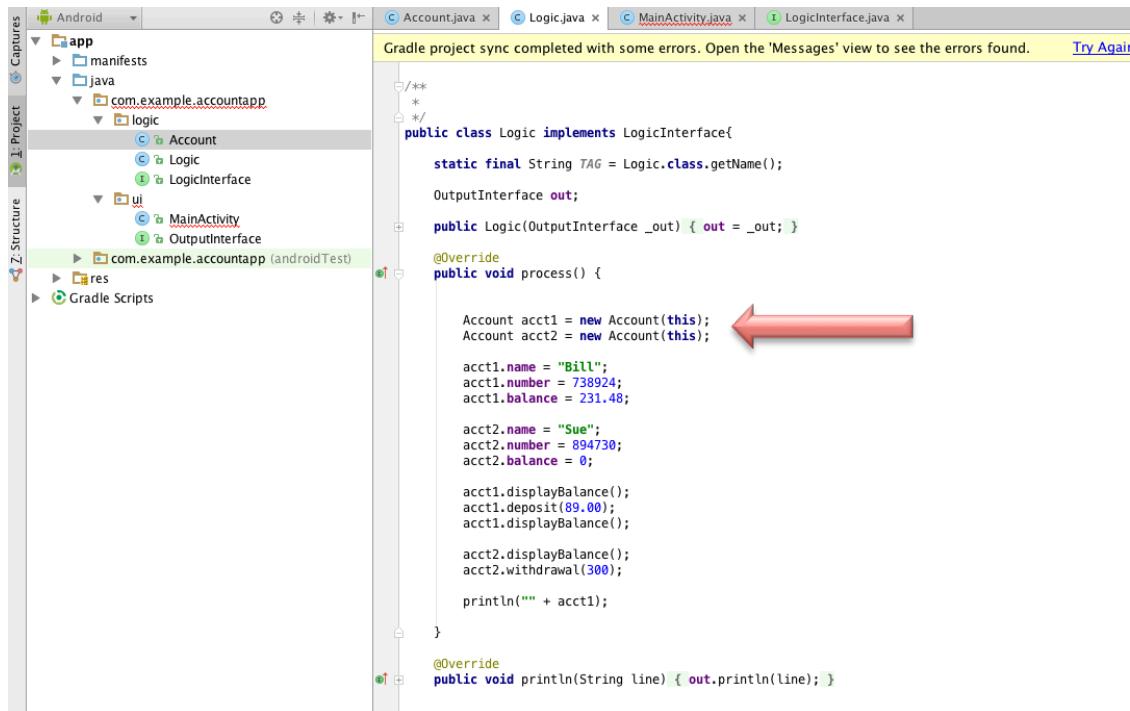
# The Account class



# The Account class



# An Account Client class: Logic.java



The screenshot shows the Android Studio interface with the Logic.java file open in the editor. The project structure on the left includes modules like app, manifests, java, and com.example.accountapp. Inside com.example.accountapp, there are logic, ui, and res directories containing Account, Logic, LogicInterface, MainActivity, and OutputInterface classes. The Logic.java file contains Java code for a Logic class that implements LogicInterface. A red arrow points to the line where two Account objects are created: `Account acct1 = new Account(this);` and `Account acct2 = new Account(this);`.

```
/*
 */
public class Logic implements LogicInterface{
    static final String TAG = Logic.class.getName();

    OutputInterface out;

    public Logic(OutputInterface _out) { out = _out; }

    @Override
    public void process() {

        Account acct1 = new Account(this);
        Account acct2 = new Account(this); ← Red arrow here

        acct1.name = "Bill";
        acct1.number = 738924;
        acct1.balance = 231.48;

        acct2.name = "Sue";
        acct2.number = 894730;
        acct2.balance = 0;

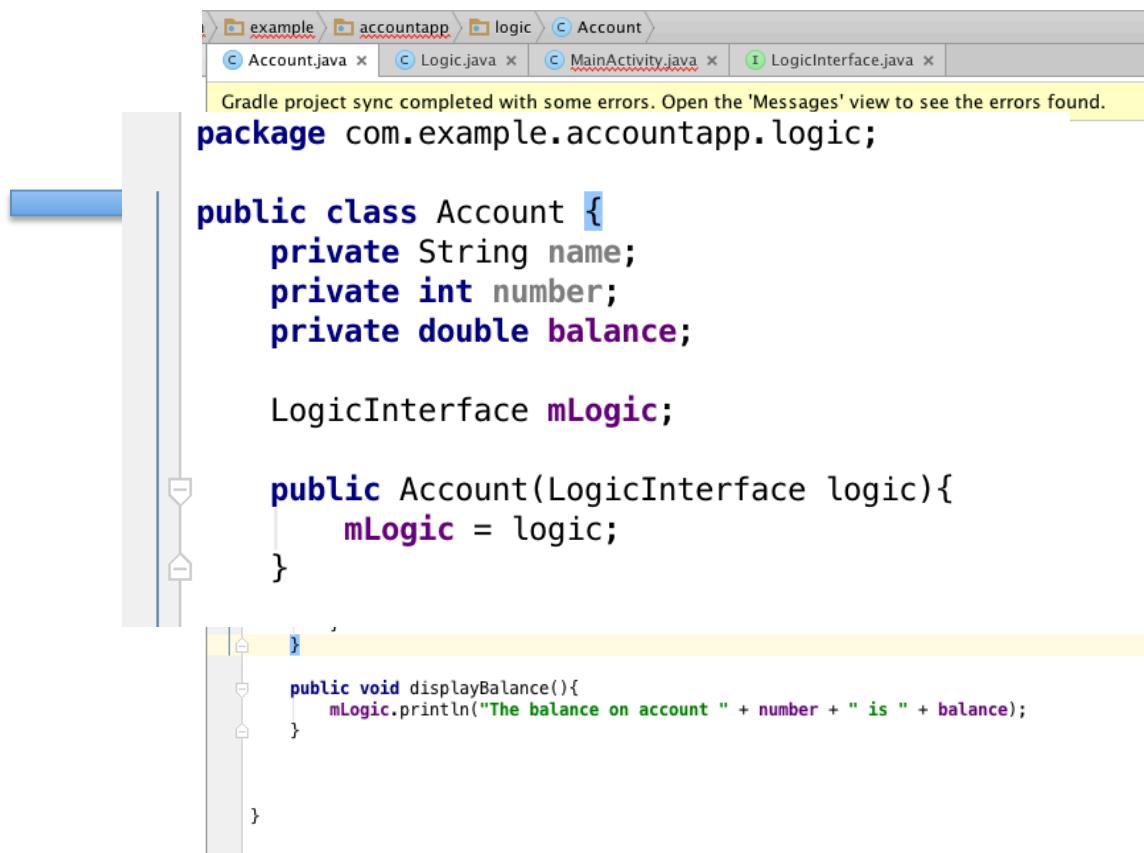
        acct1.displayBalance();
        acct1.deposit(89.00);
        acct1.displayBalance();

        acct2.displayBalance();
        acct2.withdrawal(300);

        println("+" + acct1);
    }

    @Override
    public void println(String line) { out.println(line); }
}
```

# The Account class



A screenshot of an IDE interface showing the `Account.java` file. The file contains Java code for an `Account` class. The code includes private fields for name, number, and balance, a constructor that initializes `mLogic`, and a `displayBalance` method that prints the account number and balance. A yellow status bar at the bottom indicates that a Gradle project sync completed with some errors, and it suggests opening the 'Messages' view to see the errors found.

```
package com.example.accountapp.logic;

public class Account {
    private String name;
    private int number;
    private double balance;

    LogicInterface mLogic;

    public Account(LogicInterface logic){
        mLogic = logic;
    }

    public void displayBalance(){
        mLogic.println("The balance on account " + number + " is " + balance);
    }
}
```

## **When Client programs have direct access to data**

- You can access data directly from client program

ABSTRACTION

?

# Protecting your data



# **Build the Account class**

- Instance variables should be private
- Methods not used to access or mutate data should be private
  - helper methods for example

# Modify the Account class

```
package com.example.accountapp.logic;

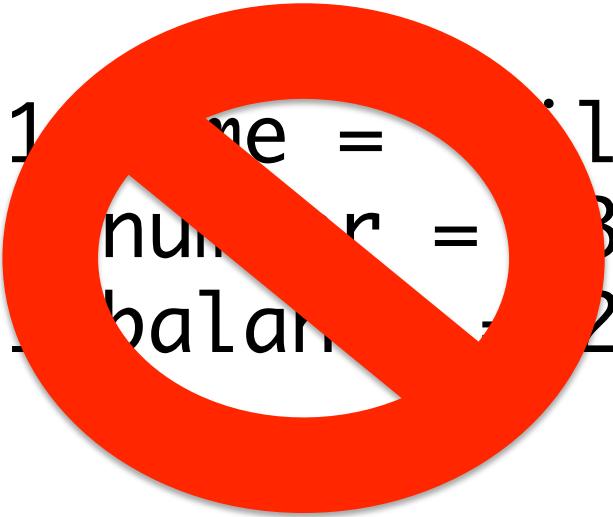
public class Account {
    private String name;
    private int number;
    private double balance;

    LogicInterface mLogic;

    public Account(LogicInterface logic){
        mLogic = logic;
    }
}
```

Attempt to run the app

# Trouble!



```
acct1.name = "Bill";
acct1.number = 38924;
acct1.balance = 231.48;
```