Protecting your data

Two types of methods often present in a class file

Accessor

- Used to access the value of a field from outside of the object
- Simply returns a value

Mutator

- Used to change the value of a field from outside of the object
- Could be a simple method that assigns a value; often called a “set” method
- Could be more complex such as our withdrawal method which requires some checks

Two types of methods often present in a class file

Accessor

```
value = acct1.getName();
```

```
public String getName(){
    return (name);
}
```

Mutator

```
acct1.setName("Rick Rudd");
acct1.deposit(18.91);
```

```
public void setName(String
    nm){
    name = nm;
}
```

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

Modify your files

the Account Class

- ✓ Add `get` methods for each of the field variables
- ✓ Add `set` methods for the name and number fields only

the Account Client logic

- ✓ Change the code by using the `set` and `deposit` methods to assign values for `acct1` and `acct2`
- ✓ Run the program

The Account Client logic

```
Account acct1 = new Account();
Account acct2 = new Account();

acct1.setName( "Bill");
acct1.setNumber (738924);
acct1.deposit(231.48);

acct1.setName( "Sue");
acct1.setNumber (894730);
acct1.deposit(0);

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

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

create objects

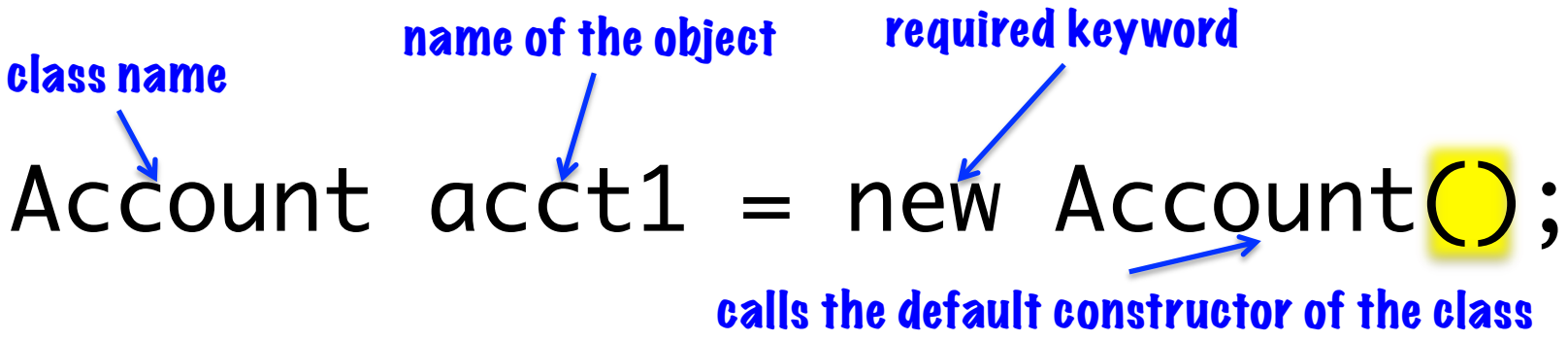
assign values

assign values

code can be found in the Logic.java process method

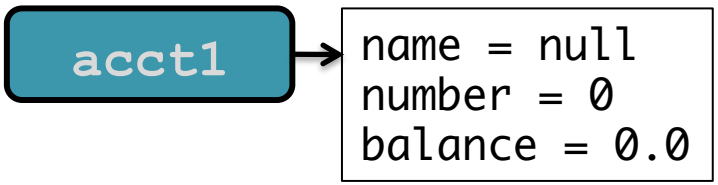
Constructors

- Used to instantiate objects
- Called using the keyword new



Two phase instantiation

```
Account acct1;
acct1 = new Account();
```



The default constructor

- Is automatically created when a class file is written
- Creates an empty instance of an object
  - int = 0
  - double = 0.0
  - boolean = false
  - String = null

We can build our own constructors

- Can take input parameters to initialize any or all of the instance variables
- Can write several constructors
  - each with their own unique input parameter list
  - this will *overload* the constructor
- Caution: once you begin writing constructors
  - the default constructor is no longer automatically created
  - must be written by the author if you want/need it

Add some constructors to the Account class

```
public Account(String newName, int newNumber){
    Account acctA = new Account("Sue Vlaki", 289476);
}


public Account(String newName, int newNumber,
               double initialBalance){
    name = newName;
Account acctB = new Account("Joseph Schmoe", 392784, 187.13);
    balance = initialBalance;
}
```

Overloading the constructor of the Account class

```
Account acctA = new Account("Sue Vlaki", 289476);

Account acctB = new Account("Joseph Schmoe", 392784, 187.13);
```

Modify your files

the Account Class

- ✓ Add a constructor method that sets the Name and number
- ✓ Add a constructor method that sets the Name, number and balance

the Account Client logic

- ✓ Look at the client program, but don't make any changes to it
- ✓ Can you figure out what's going on?

The Account Class

```
public Account(String newName, int newNumber){
    name = newName;
    number = newNumber;
}

public Account(String newName, int newNumber, double initialBalance){
    this(newName, newNumber);
    balance = initialBalance;
}
```

Loss of the default constructor

- The default constructor came "free" with the class
- As long as no other constructors were written
- Once you begin writing constructors, you must write the default constructor if you want it

```
public Account(){
  any code you want, or none at all
}
```

Modify your files

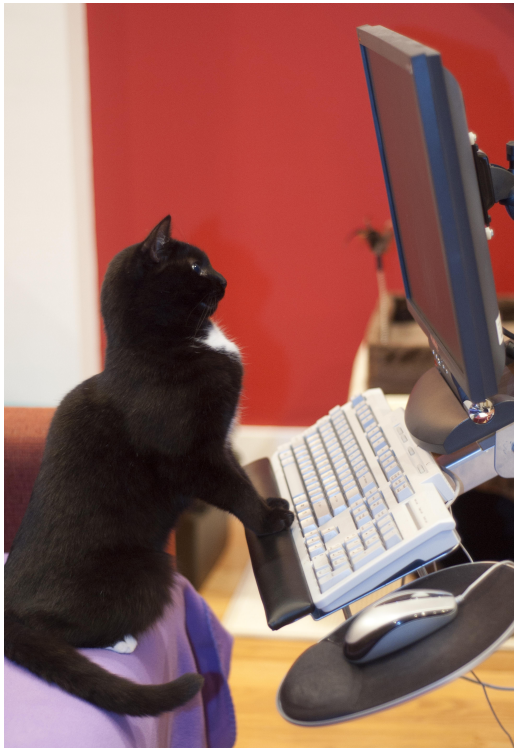
the Account Class

- ✓ Do not rewrite the default constructor

the Account Client logic

- ✓ Use the new constructors to instantiate `acct1` and `acct2`
- ✓ Errors are removed, run the program

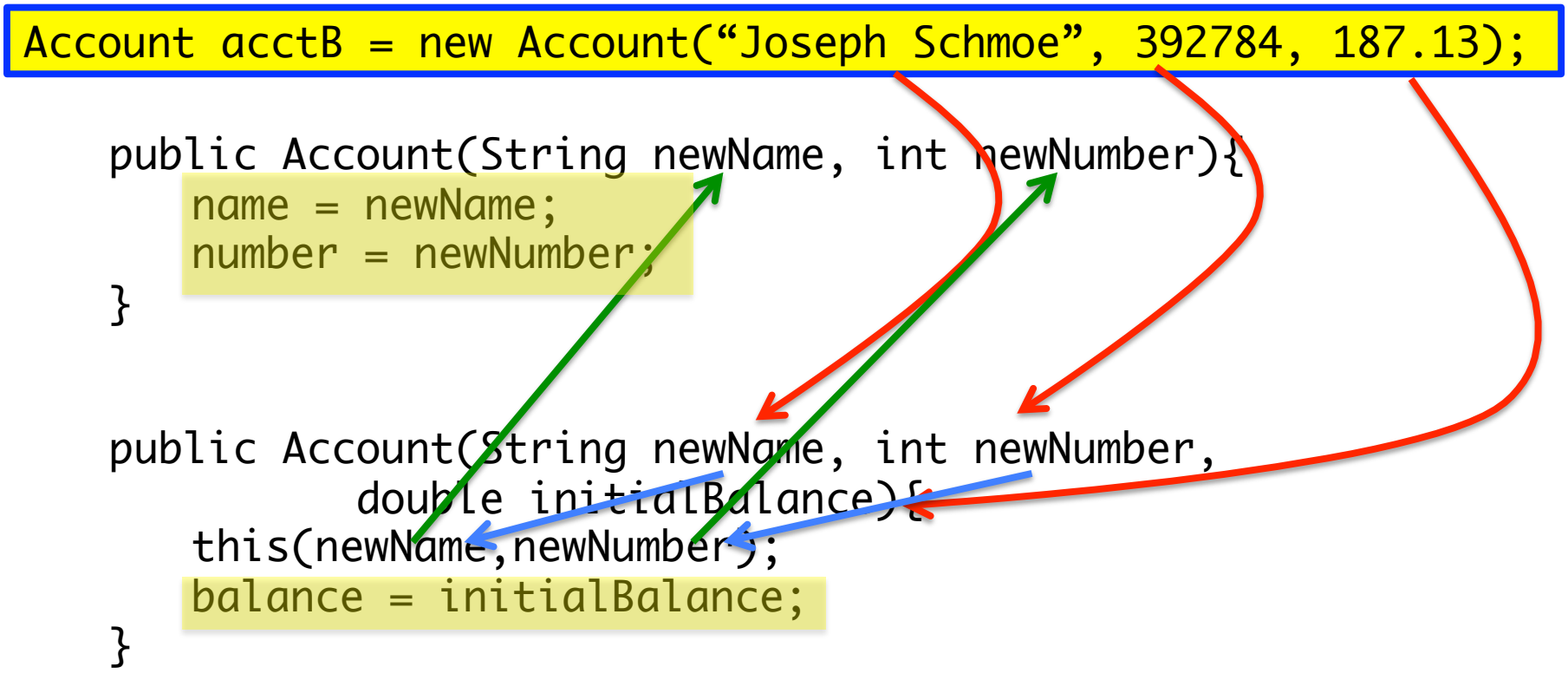
Dealing with errors!



Improve the constructors in the Account class

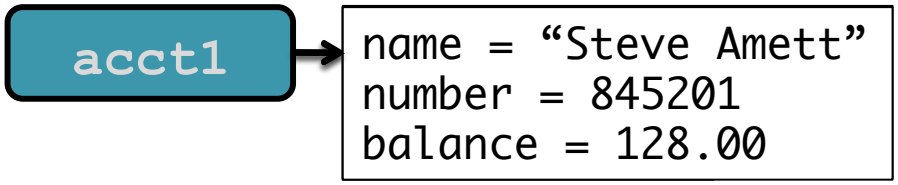
```
public Account(String newName, int newNumber){
    name = newName;
    number = newNumber;
}


public Account(String newName, int newNumber,
               double initialBalance){
    this(newName, newNumber);
    balance = initialBalance;
}
```

What's going on?



Get the order right!

```
public Account(String newName, int newNumber){
    name = newName;
    number = newNumber;
}
```




```
public Account(String newName, int newNumber,
               double initialBalance){
    this(newName,newNumber);
    balance = initialBalance;
}
```

Get the order right!

```
public Account(String newName, int newNumber){
    name = newName;
    number = newNumber;
}
```

```
name = null
number = 0
balance = 128.00
```

```
public Account(String newName, int newNumber,
               double initialBalance){
    balance = initialBalance;
    this(newName,newNumber);

}
```