LECTURE 5: INHERITANCE

CS 2110
Fall 2021

Agenda

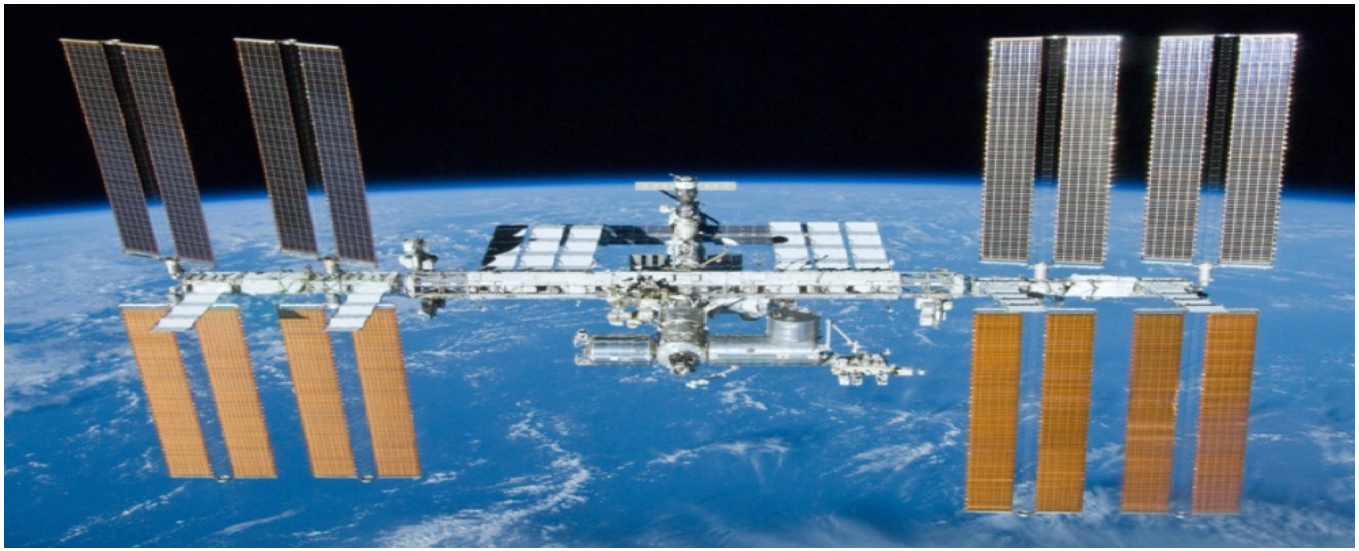
Previously in 2110:

- Primitive types
- Objects and classes
- Encapsulation

Today:

- Subclassing
- Inheritance
- Overriding
- Constructors, revisited
- Exception handling, revisited

Recall: Building Bigger



Clockwise: Knap of Howar, St Peter's Basilica, Burj Khalifa, ISS; all images in public domain

Reusable Building Blocks

Image: https://pixabay.com/photos/lego-building-blocks-colorful-2285065/

Customizable Components

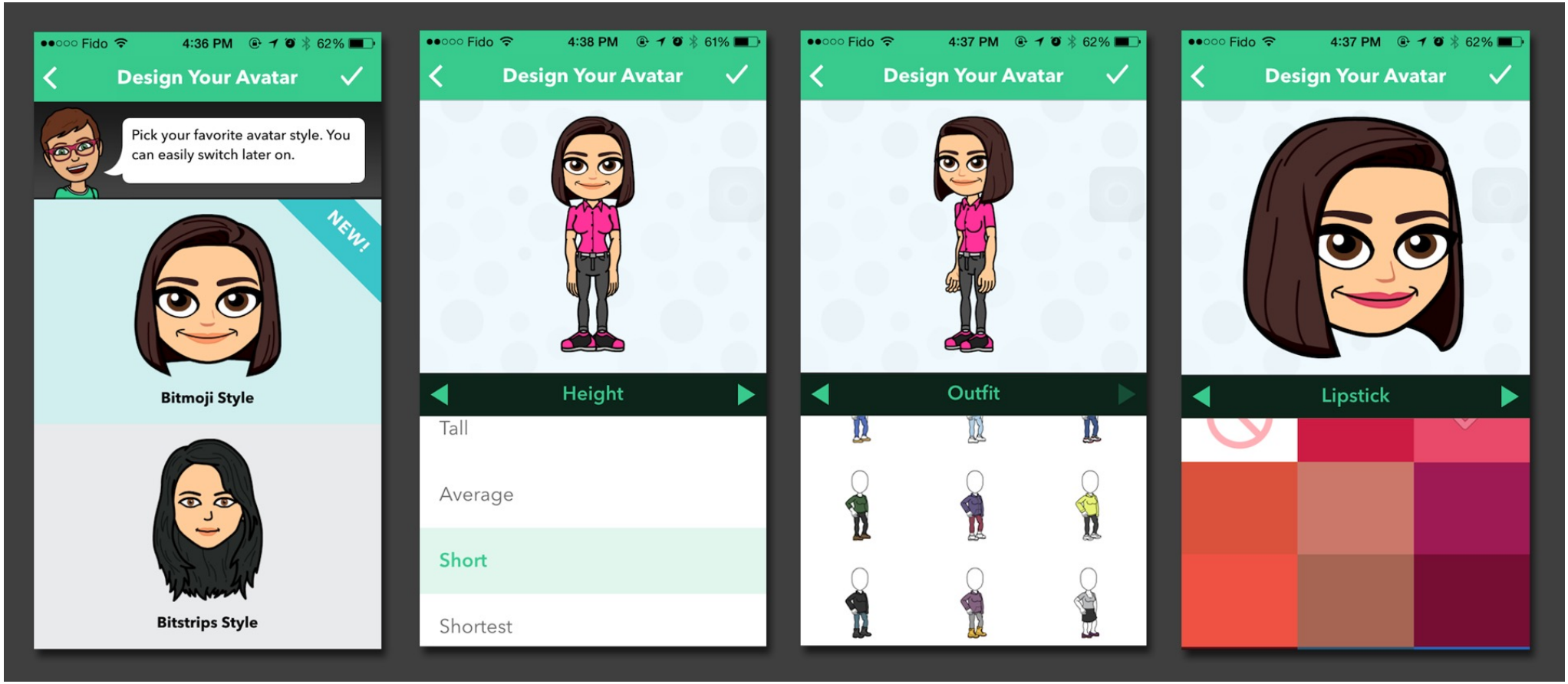


Image: https://tinaciousdesign.com/portfolio/bitmoji/

Inheritance

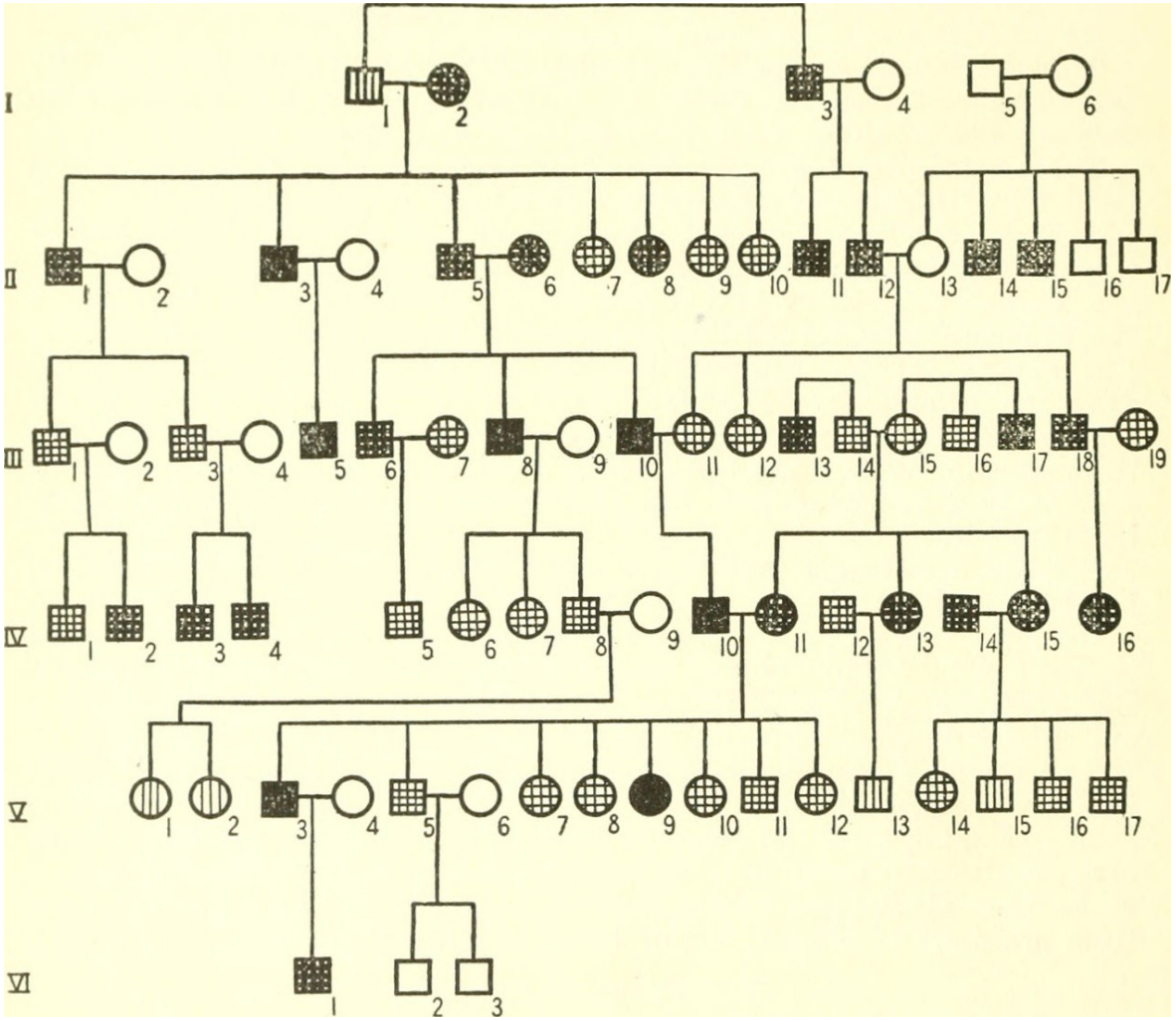


Image: https://www.flickr.com/photos/internetarchivebookimages/19765946813/

LECTURE 5: INHERITANCE

PART 2: DEMO, BANK ACCOUNTS

CS 2110
Fall 2021

Account


```
public class Account {
    private double balance;
    public void credit(double amount)
        { balance+= amount; }
}
```

Code Copy: Interest-Bearing Account


```
public class InterestAccount {
   private double balance;
   private double interestRate;
   public InterestAccount(double rate)
      { interestRate= rate; }
   public void credit(double amount)
      { balance+= amount; }
}
```

Problem: Adding Account Numbers


```
public class Account {
    private double balance;
    private String number;
    public Account(String num) { number= num; }
    … }
public class InterestAccount {
    private double balance;
    private String number;
    private double interestRate;
    public InterestAccount(String num, double
        { number= num; interestRate= rate; }
    … }
```

Duplicated code

Image: https://www.flickr.com/photos/donnieray/9436653177

Solution: Inheritance


```
public class Account {
    private double balance;
    private String number;
    public Account(String num) { number= num;
    ... }

public class InterestAccount extends Account {
    private double interestRate;
    public InterestAccount(String num, double rate)
        { super(num); interestRate= rate; }
    ... }
```

Reuse

Customize

LECTURE 5: INHERITANCE

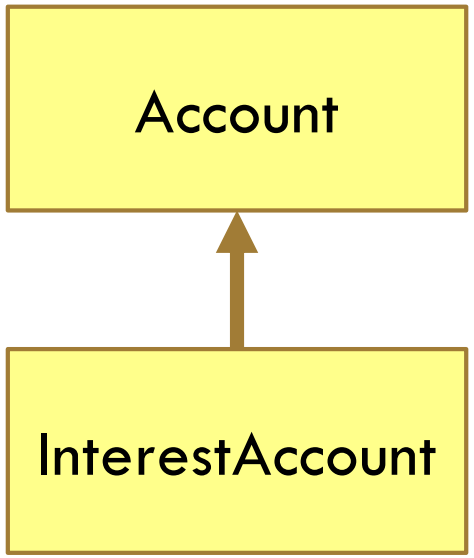
PART 3: SUBCLASSES

CS 2110
Fall 2021

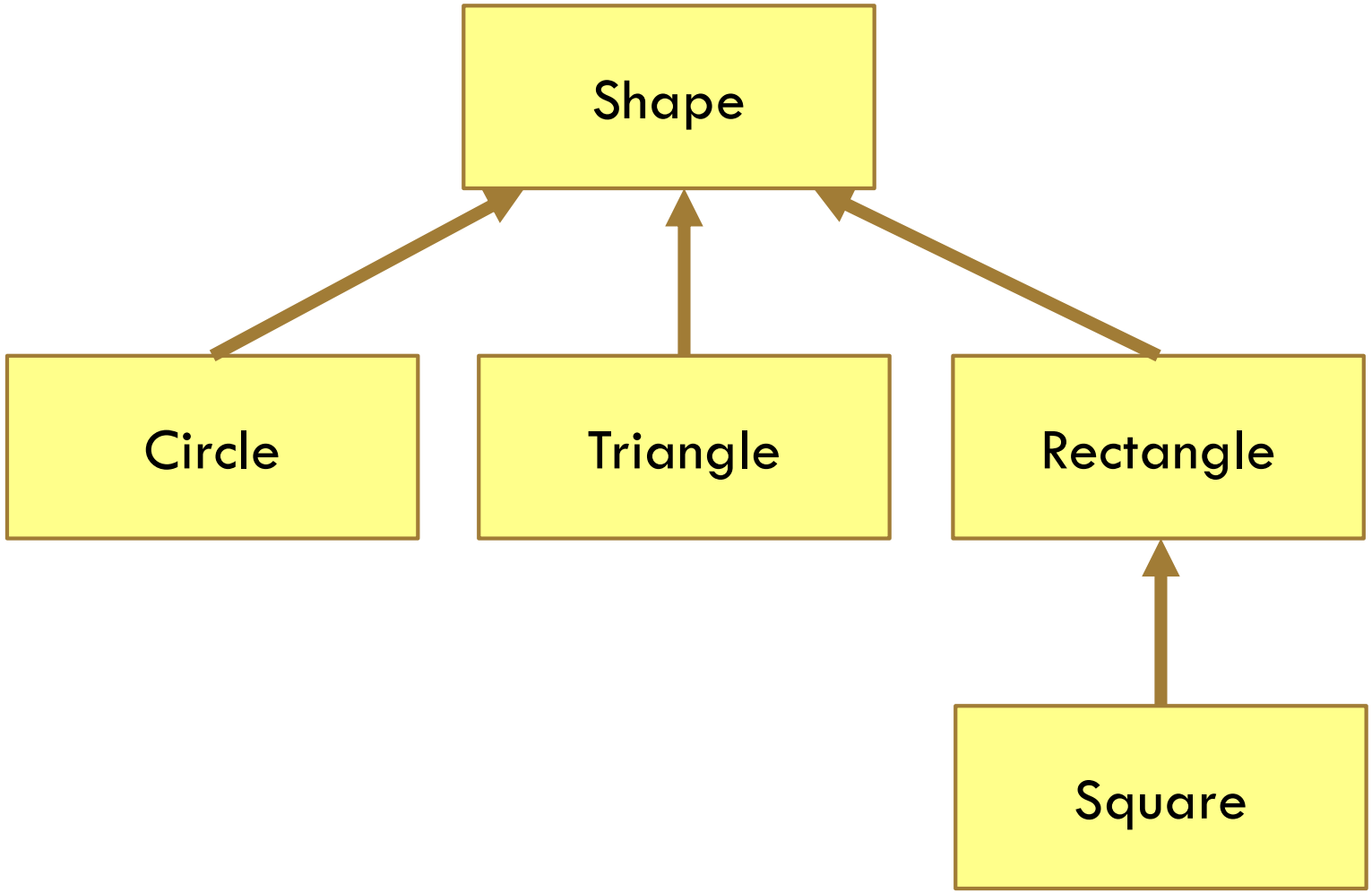
Subclassing

public class InterestAccount extends Account

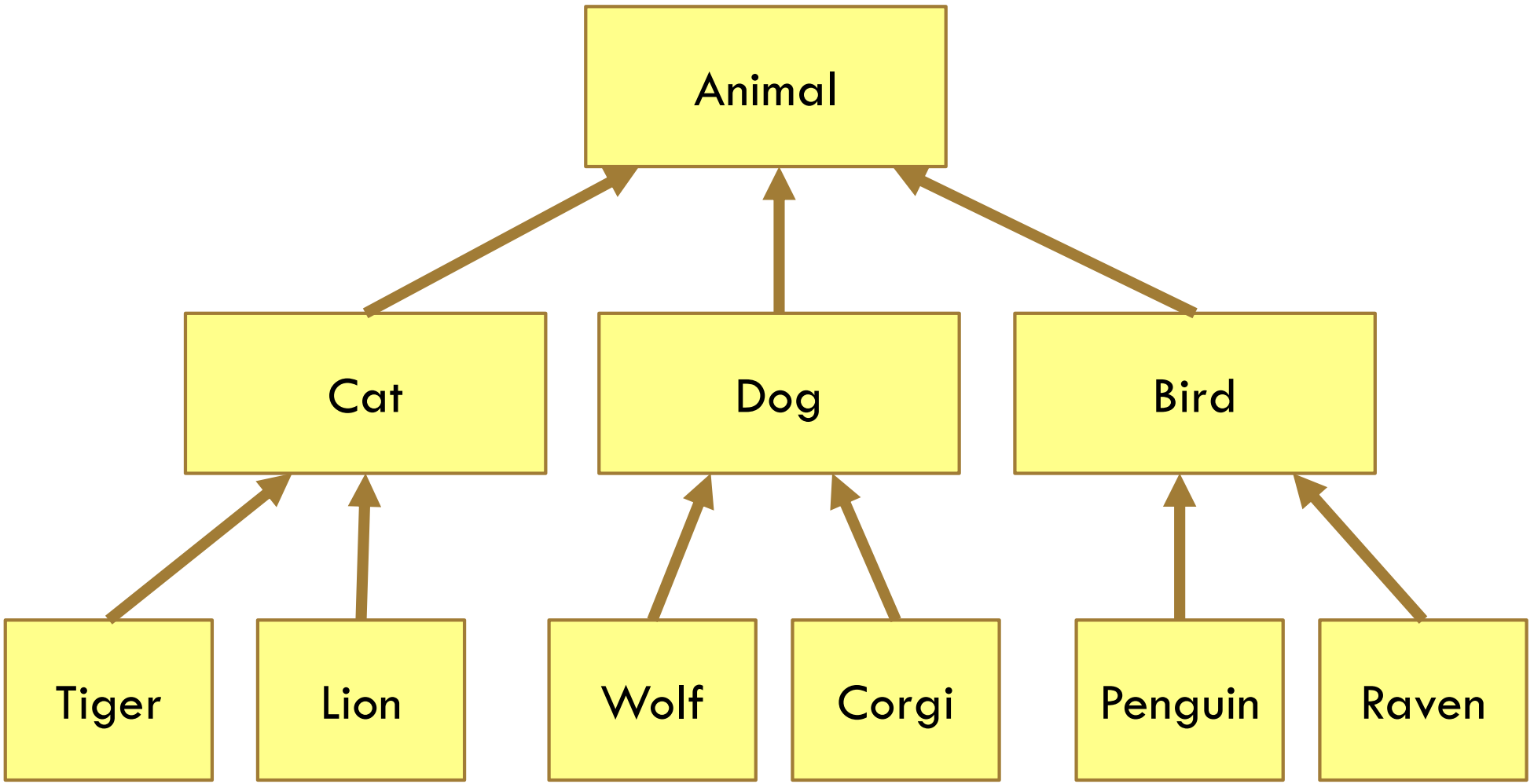
- InterestAccount is *a* subclass of Account
- Account is *the* superclass of InterestAccount
- A class has at most one direct superclass
- Class hierarchy diagram:



Class Hierarchy



Class Hierarchy



Extends: "Is A"

- Extension should model real world
- A should extend B if and only if **A "is a" B**
  - An elephant is an animal, so Elephant **extends** Animal
  - A car is a vehicle, so Car **extends** Vehicle

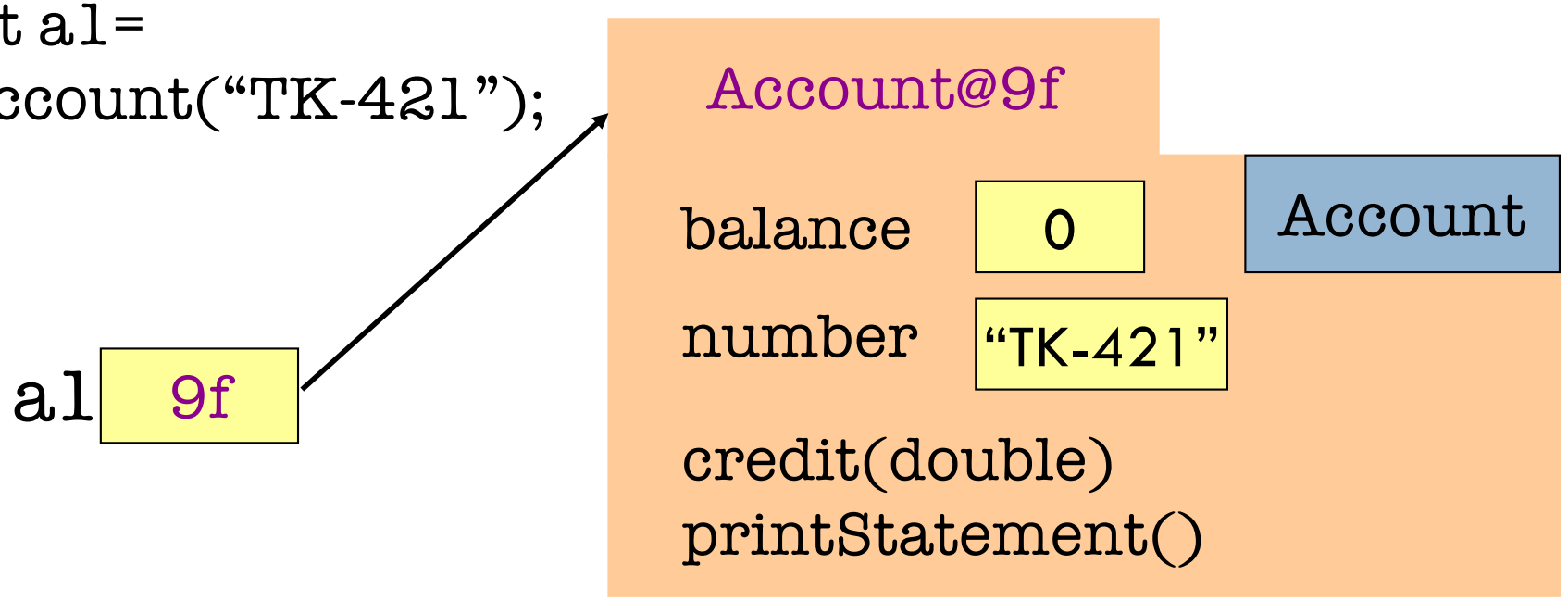
LECTURE 5: INHERITANCE

PART 4: INHERITANCE

CS 2110
Fall 2021

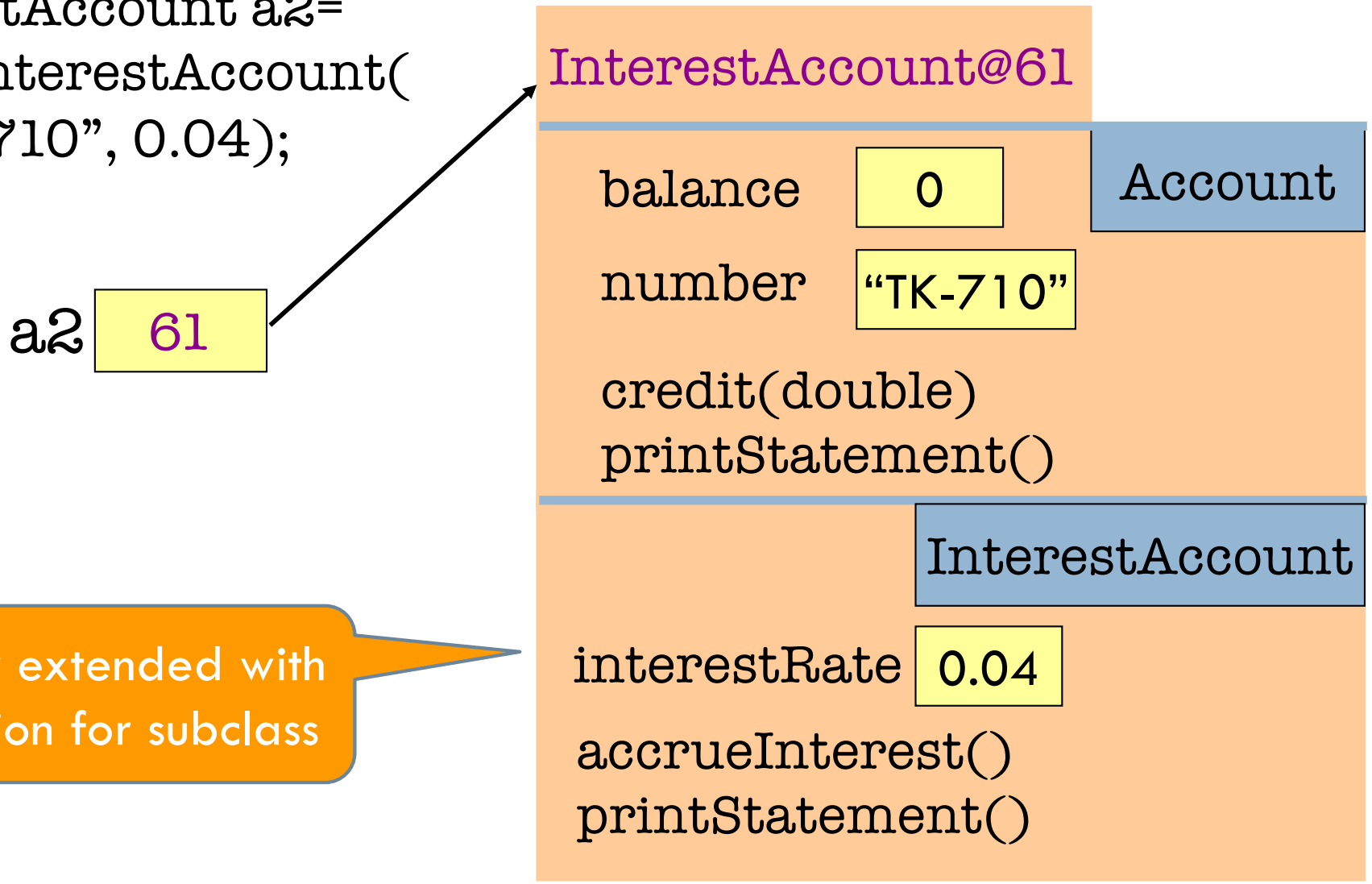
Subclasses Add Partitions to Folder


```
Account al=
  new Account("TK-421");
```

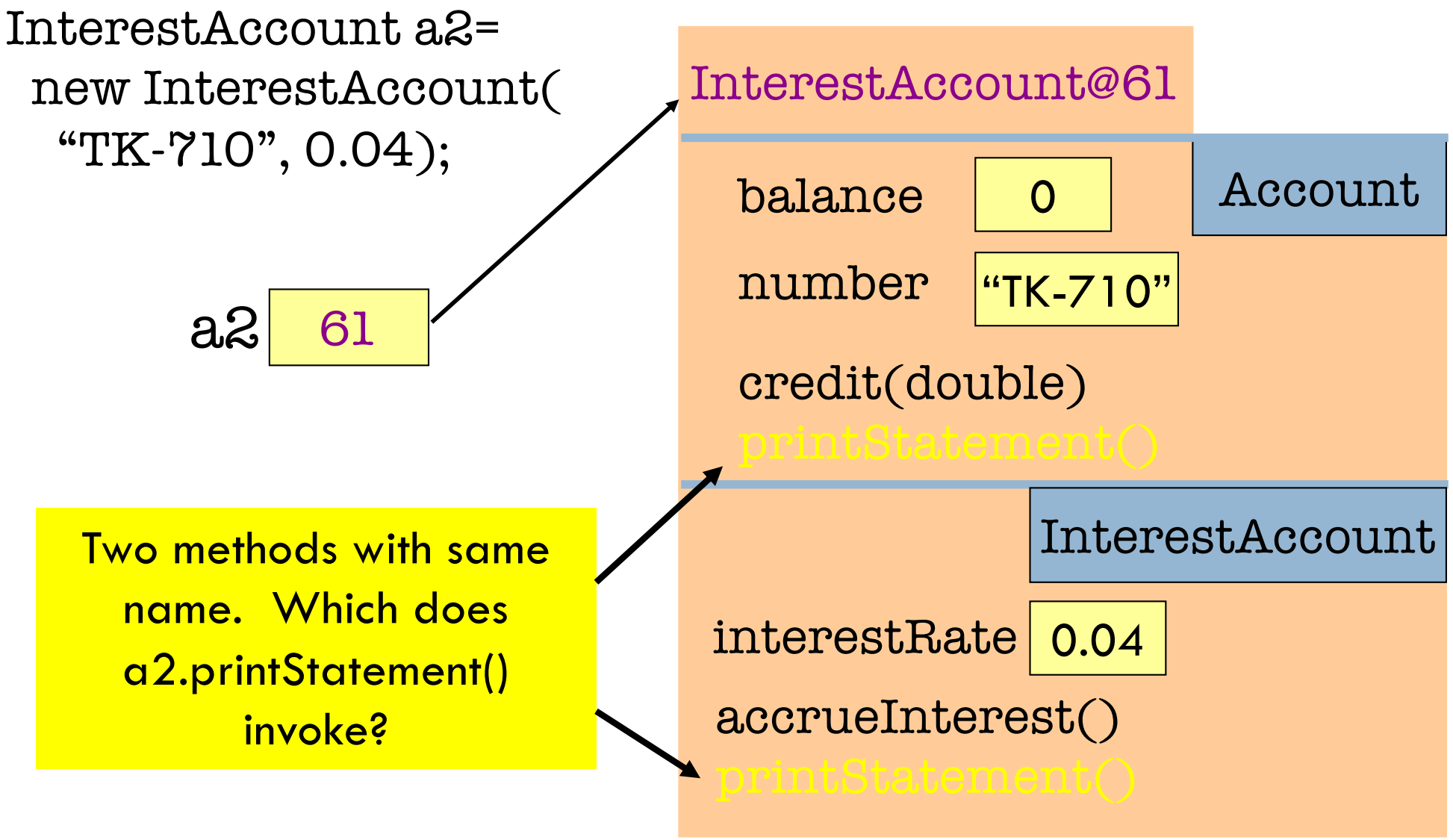


Subclasses Add Partitions to Folder


```
InterestAccount a2=
  new InterestAccount(
    "TK-710", 0.04);
```



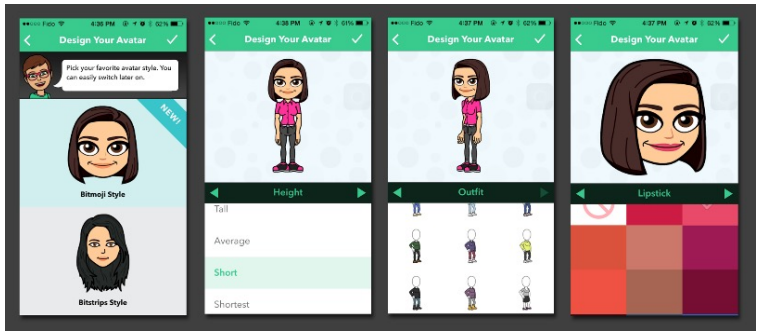
Subclasses Add Partitions to Folder



Inheritance

- A subclass inherits the fields and methods of its superclass
  - Like genetic traits that are passed on
  - Enables reuse

- A subclass can override methods of its superclass
  - Like engineered genetic variation
  - Enables customization



Overriding

Annotation placed on method to indicate intent to override (not just accidental)

```
@Override
public void printStatement() {
    super.printStatement();
    System.out.println("Rate: " + interestRate);
}
```

Invoke method using bottom-up rule, but start looking in partition above

Overriding vs. Overloading

Override

```
class C { void m() { ... } }
class D extends C { void m() { ... } }
```

Overload

```
class E {
  void m() { ... }
  void m(int i) { ... }
}
```

LECTURE 5: INHERITANCE

PART 5: OVERRIDING TOSTRING()

CS 2110
Fall 2021

toString()

Every object has method toString(), which represents object as a string

- Default implementation: object's name as a string
- Override to customize

```
class Account { ...
   @Override
   public String toString() {
      return "Account " + number;
   }
}
```

...why does this work?

Class Object

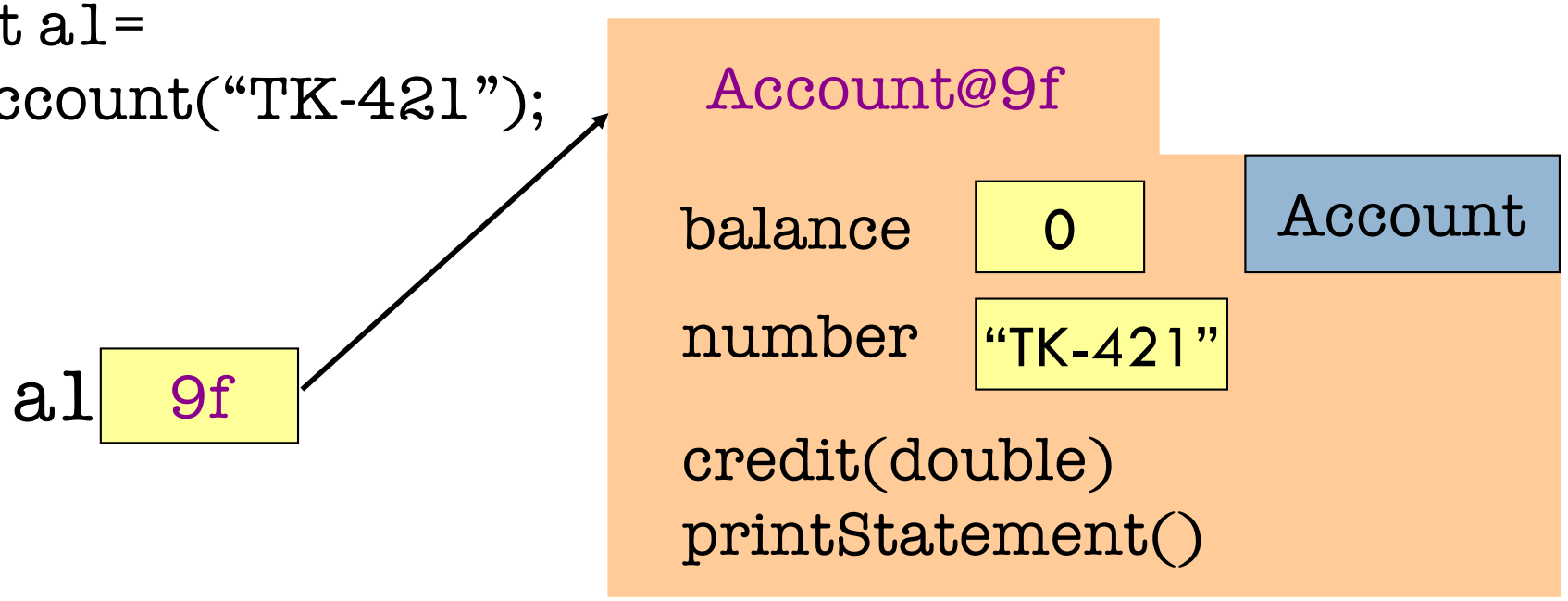

```
class Object { ...
   public String toString() {
      return ... /* object's name */; } }

class Account extends Object { ... }
```

Any class that does not explicitly extend another class automatically extends class Object

Object is the **superest** class of them all

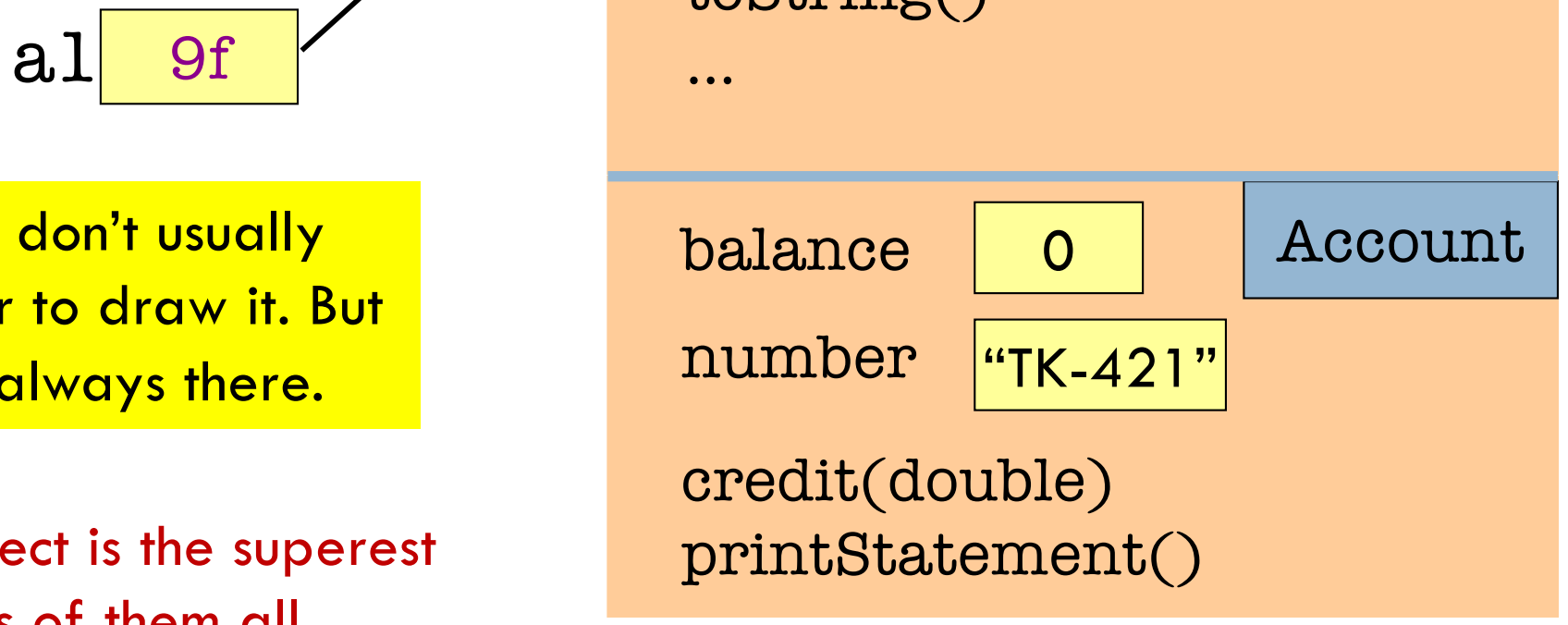
Every object has an Object partition


```
Account al=
  new Account("TK-421");
```



Every object has an Object partition


```
Account a1=
  new Account("TK-421");
```



We don't usually bother to draw it. But it's always there.

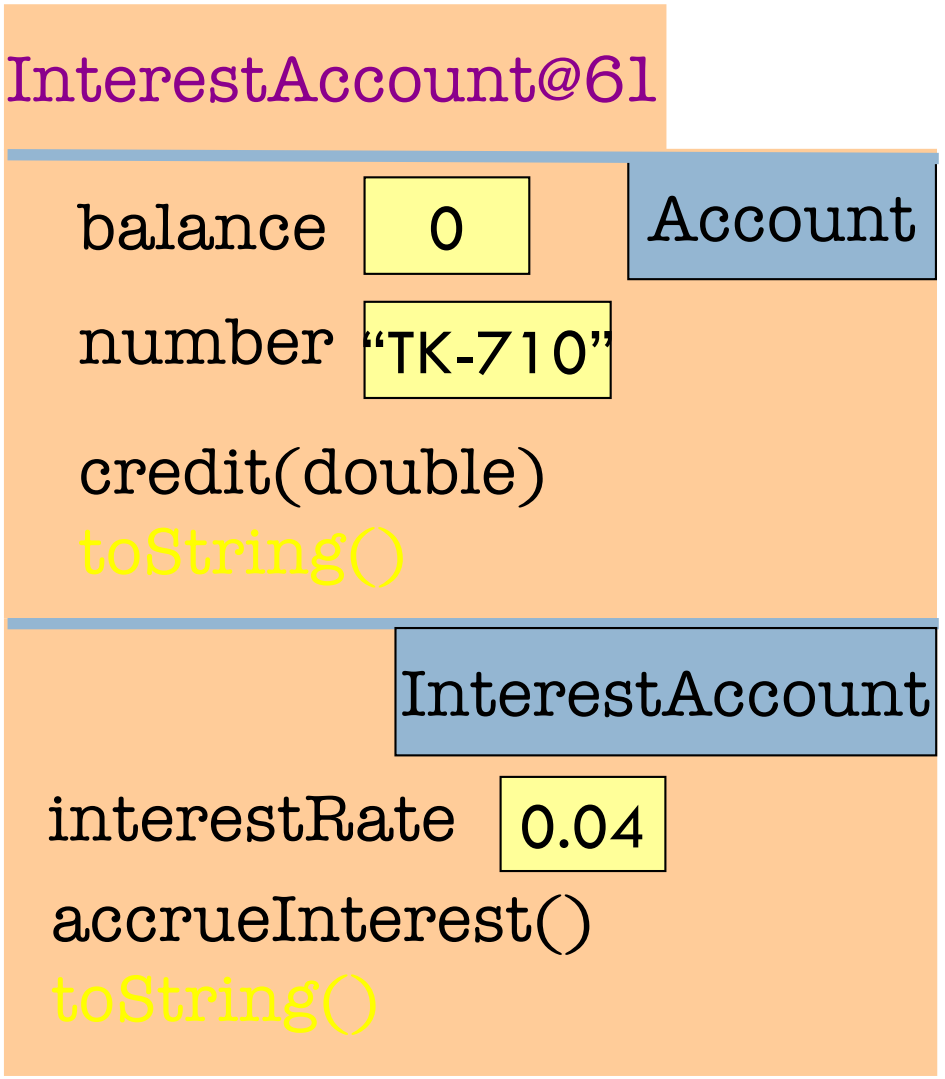
Object is the superest class of them all

Subclasses Add Partitions to Folder


```
public String toString() {
    return " acct " + number +
            " balance " + balance;
}
```

super.m(…): Bottom-up rule for m(…), but start in partition above

```
public String toString() {
    return   super. toString() +
        " int rate " + interestRate;
}
```



LECTURE 5: INHERITANCE

PART 6: SUPER CONSTRUCTORS

CS 2110
Fall 2021

Delegation

```
class C {
   C(...) {
      // first statement of constructor
      ...
   }
}
```

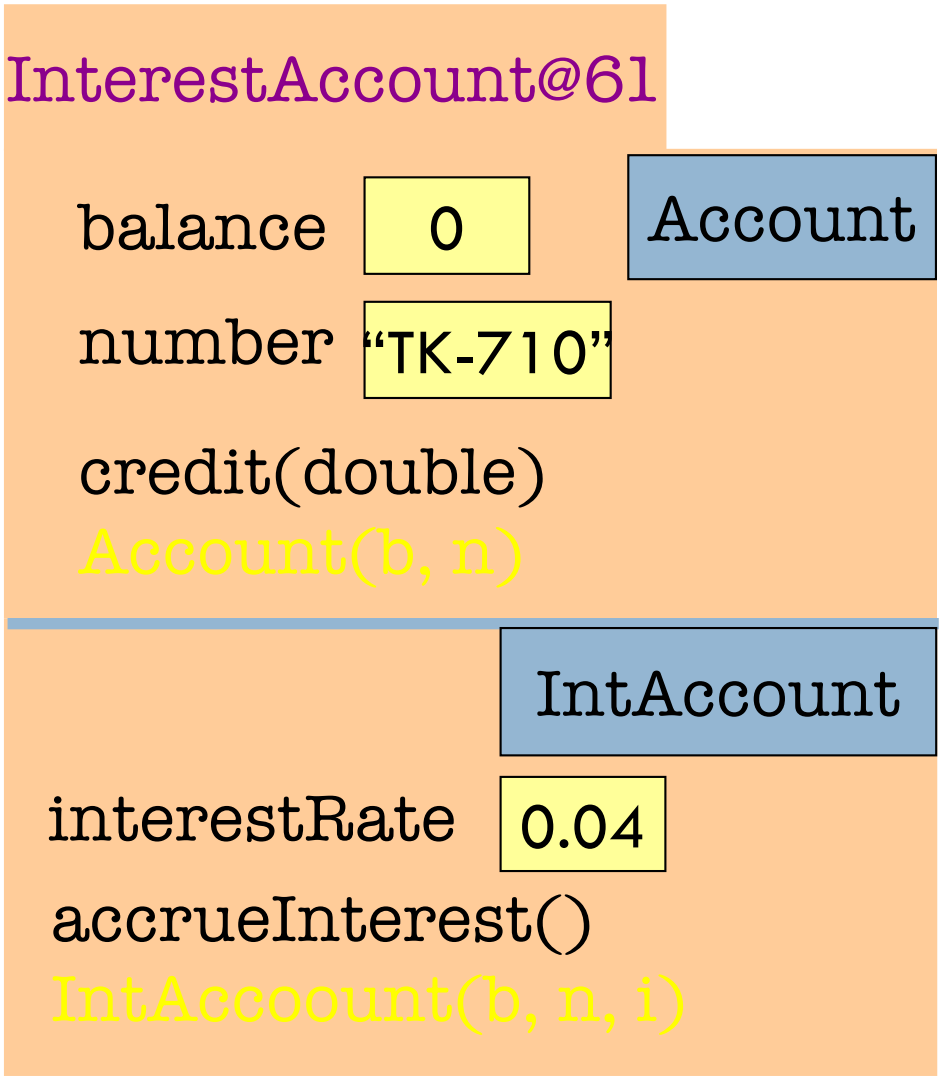
- Can delegate within class: `this(args)`
- Can delegate to superclass: `super(args)`
- If omitted, delegates to superclass: `super()`

Delegation


```
public Account(b, n){
    balance= b;
    number= n;
}
```

Every constructor must start with a call on another constructor. If not, Java inserts super();

```
public IntAccount(b, n, i){
    super(b, n);
    interestRate= i;
}
```



Default Constructor, revisited

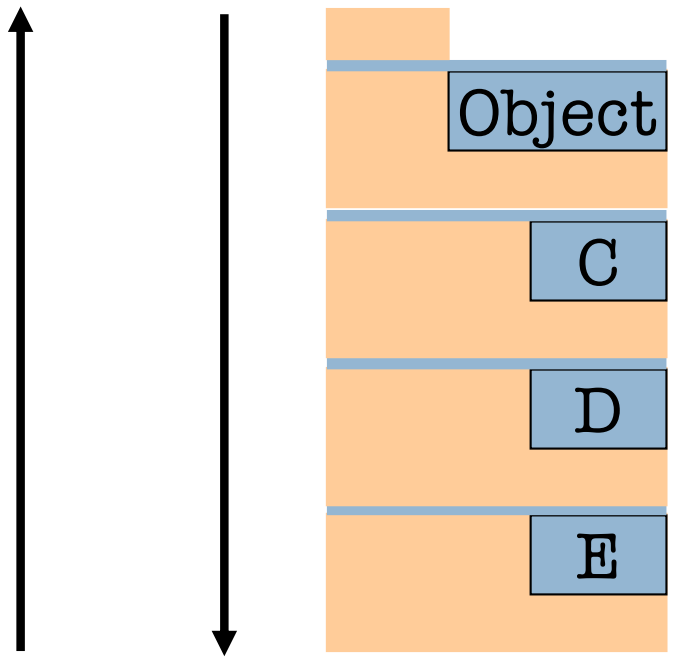
Java inserts default constructor if class C does not define any constructors:

```
public C() {}
```

Automatically calls `super()`

Initialization Order

- Consequence of last two slides: every partition invokes super constructor before doing its own work
- Initialization is therefore top down



Summary: Inheritance

- Extend class to reuse code in maintainable way
- Override methods to customize behavior

LECTURE 5: INHERITANCE

PART 7: MORE ON EXCEPTIONS

CS 2110
Fall 2021

Class Throwable

From the Javadoc for class Throwable

Class Throwable is the superclass of all errors and exceptions in Java. Only objects that are instances of this class (or one of its subclasses) are thrown by the Java Virtual Machine or can be thrown by the Java throw statement.

By convention, class Throwable and its subclasses have two constructors: one takes no arguments and one takes a String argument that can be used to produce a detail message.

The Throwable Hierachy

Throwable

- Exception
  - RuntimeException
    - ArithmeticException
    - IllegalArgumentException
    - IndexOutOfBoundsException
    - NullPointerException
  - Error — don't try to catch these
    - AssertionError
    - IOError

There are many more subclasses of Exception and Error!

Many catch clauses


```
try {
    code that might throw an exception
} catch (NullPointerException e) {
        code that handles it
} catch (RuntimeException e) {
        code that handles it
} catch (Throwable e) {
        code that handles it
}
```

Catches all RuntimeExceptions except NullPointerExceptions

Catches all Throwable objects except RuntimeExceptions

Your Turn: Read in JavaHyperText

- Inheritance, extends, subclass, superclass
- super (uses: super.m(…), super(…))
- this (uses: this.m(…), this(…))
- Bottom-up rule, override
- Constructor
- Exception