

Input/Output, Inheritance

Textbook:

18.3 (I/O)

22.2-22.4 (Inheritance)

I/O – Input/Output

- Input to a computer can be from multiple sources:
 - Keyboard
 - Mouse
 - Touchscreen (basically like a mouse if not multi-touch)
 - Microphone

 - The network
 - Files on the hard drive (our next goal)
- Output from a computer can be to multiple destinations:
 - Screen
 - Speakers

 - The network
 - Files on the hard drive (our next goal)

How to get String(s) from a File

Processing Style

Function description is:

```
String[] loadStrings(String filename);
```

Used as:

```
String[] arrayNameHere =  
loadStrings("C:\\a\\full\\path\\goes  
\\here.txt");
```

- If just give a name, it expects the file with the given filename to be in the sketch's data/ directory.
- Can also use full pathname to point to any location on hard drive
- Notice it returns an array – each entry in the array contains one line of text from the file
- *arrayNameHere* can be the name of any array

How to save String(s) to a File

Processing Style

Function description is:

```
void saveStrings(String filename,  
String[] arrayToSave);
```

Used as:

```
saveStrings("C:\\a\\full\\path\\goes\\  
here.txt", arrayToSave);
```

Saves all of the String entries in the *arrayToSave* array to the file specified by filename, with one String per line (*arrayToSave* can be String array's name)

How to get String(s) from a File – Java-ish Style

Need to make use of a class called `BufferedReader` (reads a line at a time from a file)

Create a reader for a given file:

```
BufferedReader theReader = createReader("C:\\a\\full\\path\\goes  
\\here.txt");
```

Read one line:

```
String aLineFromFile = theReader.readLine(); // want to put this in a  
loop to read multiple lines
```

Make sure it gave you something useful:

```
if (aLineFromFile != null) { ... do something with the line ... };
```

When done reading, close the reader:

```
theReader.close();
```

Be prepared to handle file errors (misspelled name, file can't be found, ...):

```
try  
{  
    // do all of the above file work in here  
}  
catch (Exception exc)  
{  
    println(exc);  
}
```

See
[JavaStyleFiles.pde](#)

Processing hides much of
this ugliness from us!

How to save String(s) to a File - Java-ish Style

Need to make use of a class called `PrintWriter` (prints a line at a time from a file)

Create a writer for a given file:

```
PrintWriter theWriter = createWriter("C:\\a\\full\\path\\goes  
  \\here.txt");
```

Print one line:

```
theWriter.println(someString); // want to put this in a loop to write  
multiple lines
```

When done writing, flush and close the writer:

```
theWriter.flush();  
theWriter.close();
```

Be prepared to handle file errors (misspelled name, file can't be found, ...):

```
try  
{  
  // do all of the above file work in here  
}  
catch (Exception exc)  
{  
  println(exc);  
}
```

See
[JavaStyleFiles.pde](#)

Processing hides much of
this ugliness from us!

Buffer what, flushing huh?

- Reading and writing to the hard-drive is an expensive operation – the computer tries to minimize doing this.
- When writing text to a drive, it doesn't write one character at a time or even one line at a time. It will wait and “buffer” information until it has a large chunk. When it gets a large enough chunk, that whole chunk is physically written to the drive.
- Using “flush” says – “ go ahead and write what you are holding onto out to the hard drive – don't wait!”

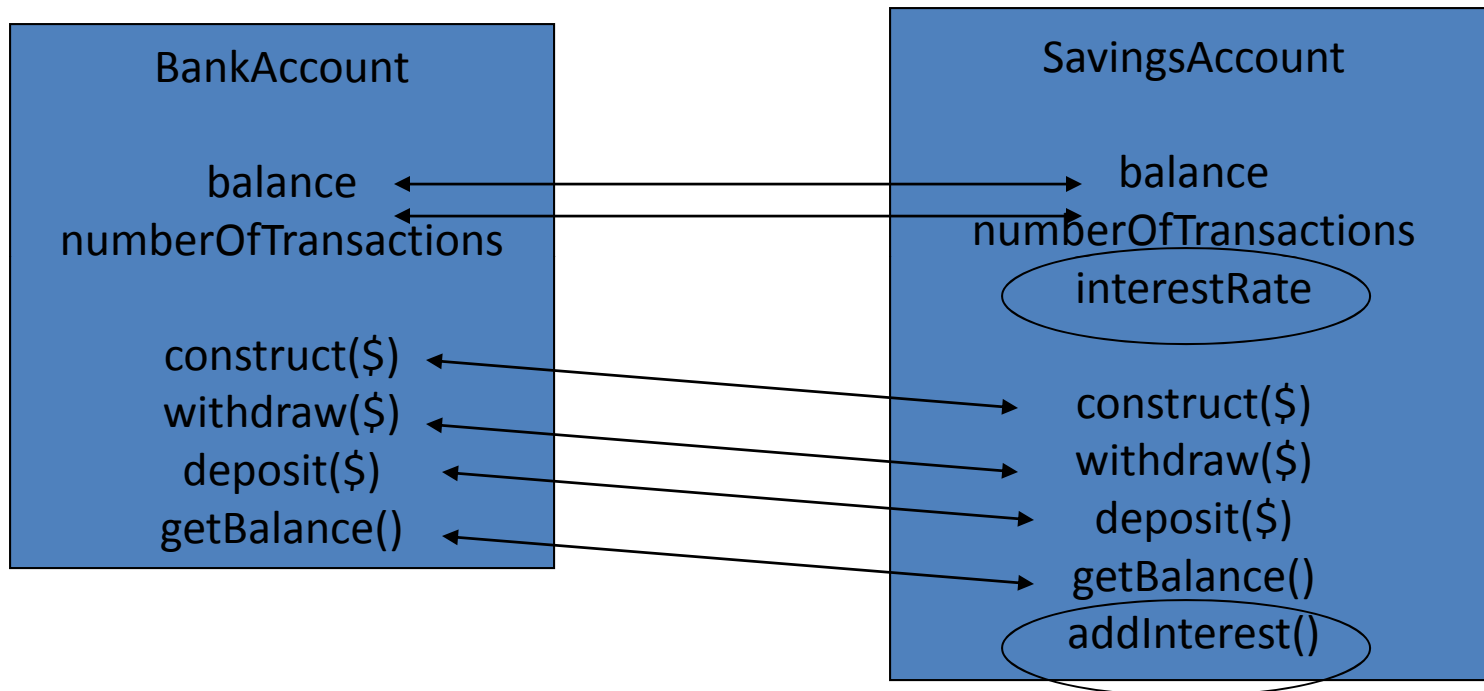
A New Topic.....

Inheritance

Object Oriented Programming: Inheritance

- Think about the following scenario:
 - Imagine I have a *BankAccount* class that has as variables *int id*, *float balance*, *int numberOfTransactions*, and functions for *construction*, *handleDeposit*, *handleWithdrawal*, *getBalance*, etc
 - I want a *SavingsAccount* class, which is essentially a *BankAccount*, but has an additional variable called *float interestRate*, and a new method called *addInterest*.

Inheritance



Similar items marked with arrows

New items/information circled

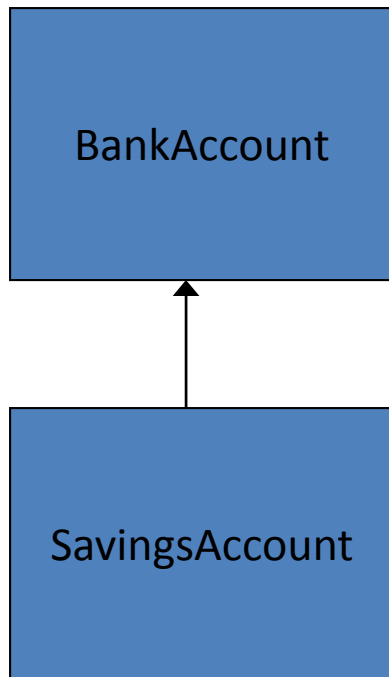
Inheritance

- I don't want to change BankAccount.pde, because it's pretty general and useful - not all accounts need an interest rate.
- I want to exploit code I already have
 - Don't want to copy all the code from BankAccount.pde into SavingsAccount.pde, because
 - I'm somewhat lazy
 - Any changes to BankAccount activities should affect SavingsAccounts, as SavingsAccounts are just a specialized type of BankAccount
 - If the BankAccount code was just copied into SavingsAccount, then I would have to update both files to ensure that both types of accounts correctly represent the core of how accounts work.

Inheritance

- If possible, I would like a simple way to indicate:
 - A particular class (SavingsAccount) “inherits” all variables and methods from another class (BankAccount), and can add some of its own variables and methods
 - Processing allows us to do this – but first let’s look at more examples...

Inheritance Examples

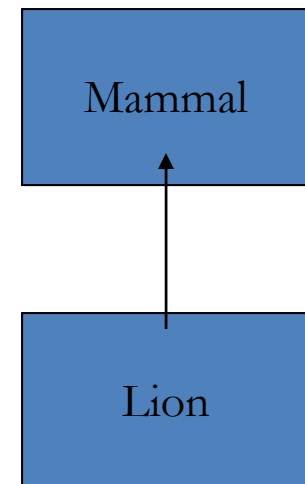


BankAccount is the **superclass** and is **general**

SavingsAccount is the **subclass**, inheriting attributes and behaviors from BankAccount, and is more **specific/specialized**, as it adds its own attributes and behaviors

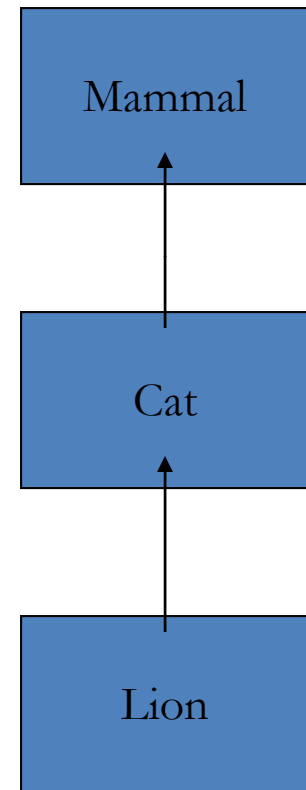
Inheritance Examples

- Lion is a subclass of Mammal:
 - A Lion fulfills Mammal properties
 - Live birth
 - Warm-blooded
 - Hair
 - Also has some lion-specific properties
 - Ability to roar
 - Tail



Inheritance Examples

- Can have multiple layers of inheritance:
A Lion is a subclass of Cat,
which is a subclass of Mammal
 - A Cat fulfills Mammal properties
 - Live birth
 - Warm-blooded
 - Hair
 - A Lion fulfills Cat properties
 - Tail CanPurr
 - Whiskers
 - A Lion has its own properties
 - Large size CanRoar



Inheritance Examples

Automobile



Variables:

year

make

model

transmissionType

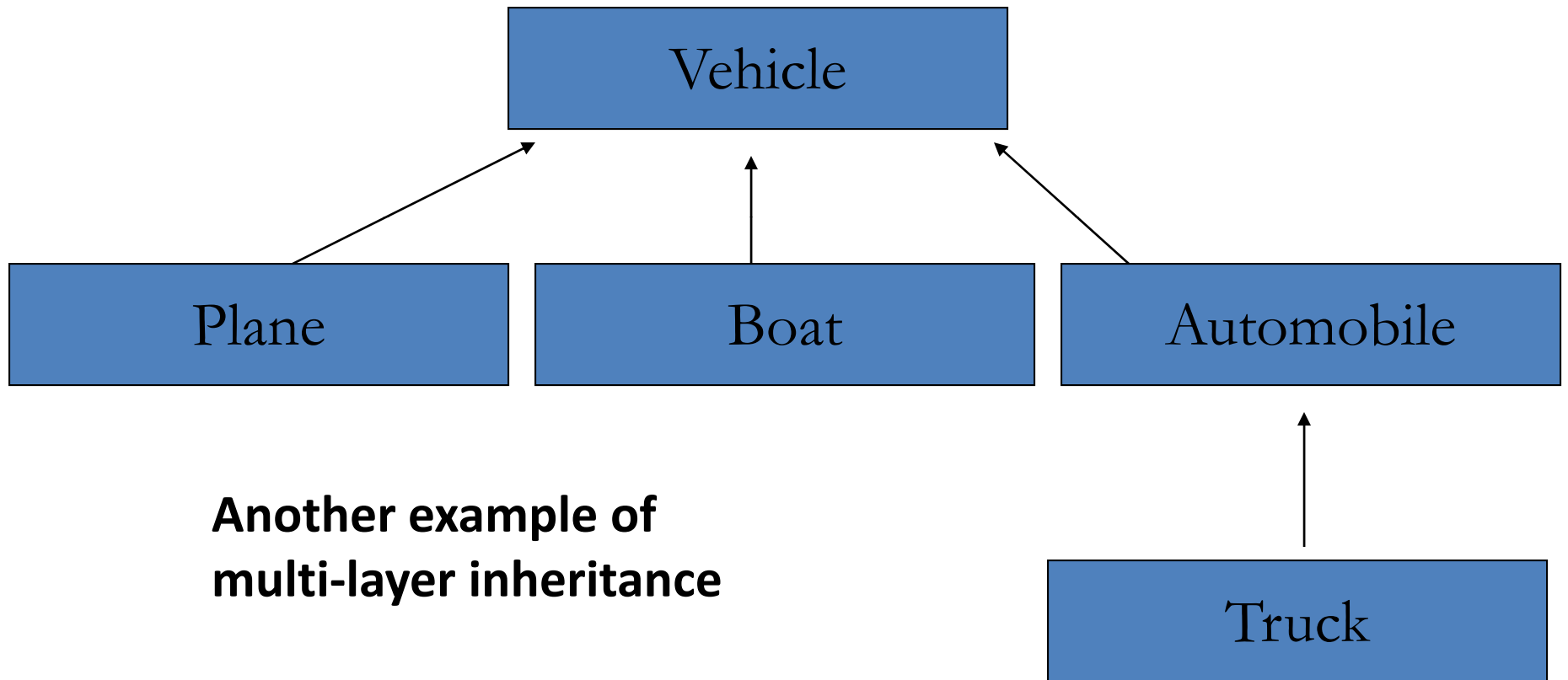
passengerCapacity

Truck

Truck-specific
variables:

bedLength

Inheritance



Inheritance

- It is within our *definition* of a class (datatype) where need to implement inheritance
 - Use normal syntax for indicating new variables and methods
 - Use new syntax for indicating who inheriting from
 - New syntax for calling parent's (superclass) methods and for overwriting parent's methods

Inheritance

- Syntax: Indicate that your current class extends its superclass
 - At top of file, instead of:

class ClassName

use

*class ClassName **extends** AnotherClassName*

```
class BankAccount
{
    ...
}
```

```
class SavingsAccount extends BankAccount
{
    ...
}
```

BankAccount - superclass

SavingsAccount - subclass