Inheritance

* You can write a class that inherits all the public features from a “parent”
* CheckingAccount – inherits features from BankAccount
* In some sense, CheckingAccount “contains” a BankAccount, in addition to providing other services

What would you do to implement CheckingAccount without inheritance?

* You could create a composition that held a BankAccount

Note: in inheritance, we don’t inherit constructors – only public fields and methods

Inheritance may be though of as a special kind of composition

e.g.

public class CheckingAccount

extends BankAccount

* CheckingAccount contains a BankAccount … in other words, any “state” contained by BankAccount is also contained by CheckingAccount
* Reminder: “state” means data
* BankAccount contains a balance as state (data), therefore, CheckingAccount contains a balance

As we saw from the example, it is possible to implement a class doing something similar to inheritance, using only composition

But … inheritance gives you two important features that are more powerful than composition:

1. The child inherits the public features from the parent (except constructors) – If the child and the parent have many public features in common, there is no need to rewrite any code, or to pass results from the contained object
2. Polymorphism – it is possible to handle CheckingAccount as a BankAccount (i.e. it is legal to cast a child to its parent). However, if we call methods within the BankAccount API, and these are overridden in CheckingAccount (withdraw and deposit), Java will call the CheckingAccount methods.

However, similar to a composition, we generally need to go through the BankAccount API in order to access features of the BankAccount.