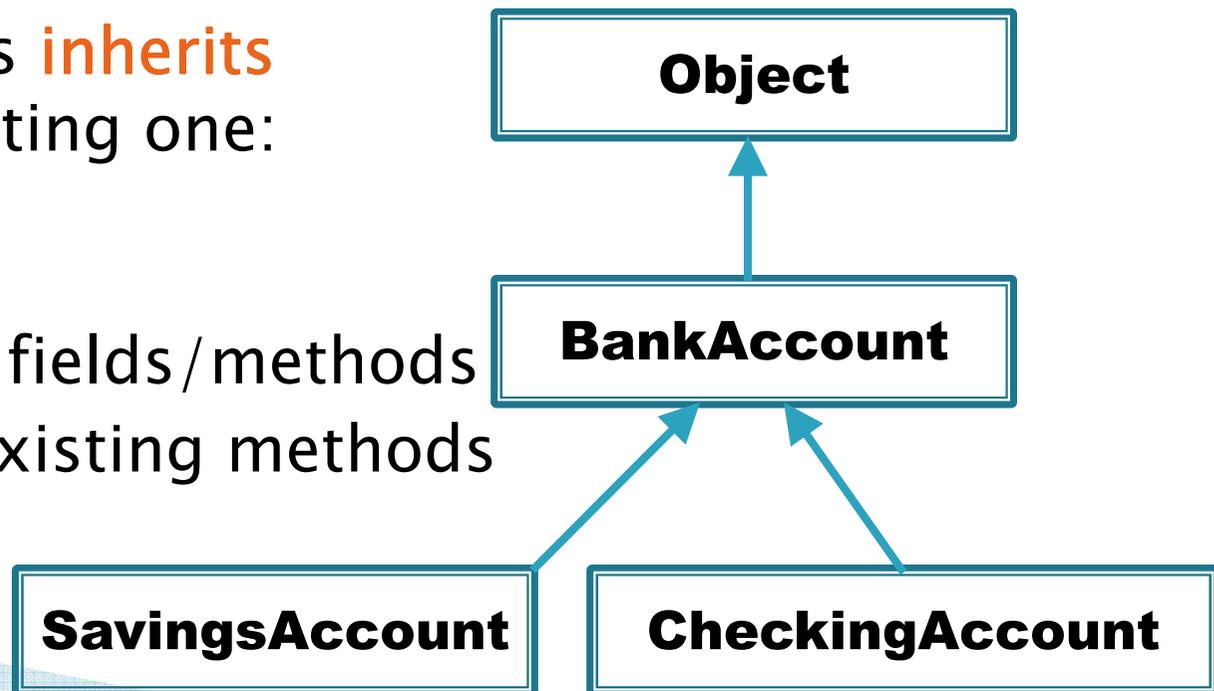


Inheritance – what is it?

- ▶ Sometimes a new class is **a special case** of the concept represented by another
 - A SavingsAccount *is-a* BankAccount
 - An Employee *is-a* Person
- ▶ Can **extend** existing class, changing just what we need
- ▶ The new class **inherits** from the existing one:
 - all methods
 - all fields
- ▶ Can add new fields/methods
- ▶ Or override existing methods



```
public class BankAccount {  
    private double balance;
```

Subclasses will inherit this field even though they cannot directly access it

- Subclasses inherit *all* fields

```
    public BankAccount() {  
        this(0.00);  
    }  
}
```

Calls the one-parameter constructor

```
    public BankAccount(double initialBalance) {  
        this.balance = initialBalance;  
    }  
}
```

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

```
    public void withdraw(double amount) {  
        this.balance -= amount;  
    }  
}
```

```
    protected final double getBalance() {  
        return this.balance;  
    }  
}
```

final means that subclasses are not permitted to override this method

- We want to count on it working just like this

protected means that *subclasses* and classes in the same *package* can access it.

- *public* makes more sense here, but I have made it protected just so that you can see an example

```
public class SavingsAccount extends BankAccount {
```

```
    private double interestRate;
```

```
    public SavingsAccount(double rate) {  
        this.interestRate = rate;  
    }
```

Fields:

- Inherits *balance* field
- DON'T put your own *balance* field here!
- Adds *interestRate* field

Implicit `super();` that calls superclass' no-parameter constructor

```
    public SavingsAccount(double rate, double initBalance) {  
        super(initBalance);  
        this.interestRate = rate;  
    }
```

Calls superclass' constructor

- Must be first statement in constructor

```
    public void addInterest() {  
        double interest;  
        interest = this.getBalance()  
            * this.interestRate / 100;  
        this.deposit(interest);  
    }
```

Adds this method to those inherited

Calls inherited *getBalance* and *deposit* methods

```
public class CheckingAccount extends BankAccount {
    private int transactionCount;

    public CheckingAccount(double initialBalance) {
        super(initialBalance);
        this.transactionCount = 0;
    }

    @Override
    public void withdraw() {
        super.withdraw();
        ++ this.transactionCount;
    }

    public void runThisOnFirstDayOfMonth() {
        if (this.transactionCount > 100) {
            super.withdraw(10.00);
        }
        this.transactionCount = 0;
    }
}
```

Overrides inherited *withdraw* method and also calls inherited *withdraw* method

- The class would have, but I have not shown, a similar *deposit* method.

This (rather silly) checking account charges a \$10 fee if you do more than 100 transactions in a month

- Note call to superclass' *withdraw*