CSSE 220 Day 18

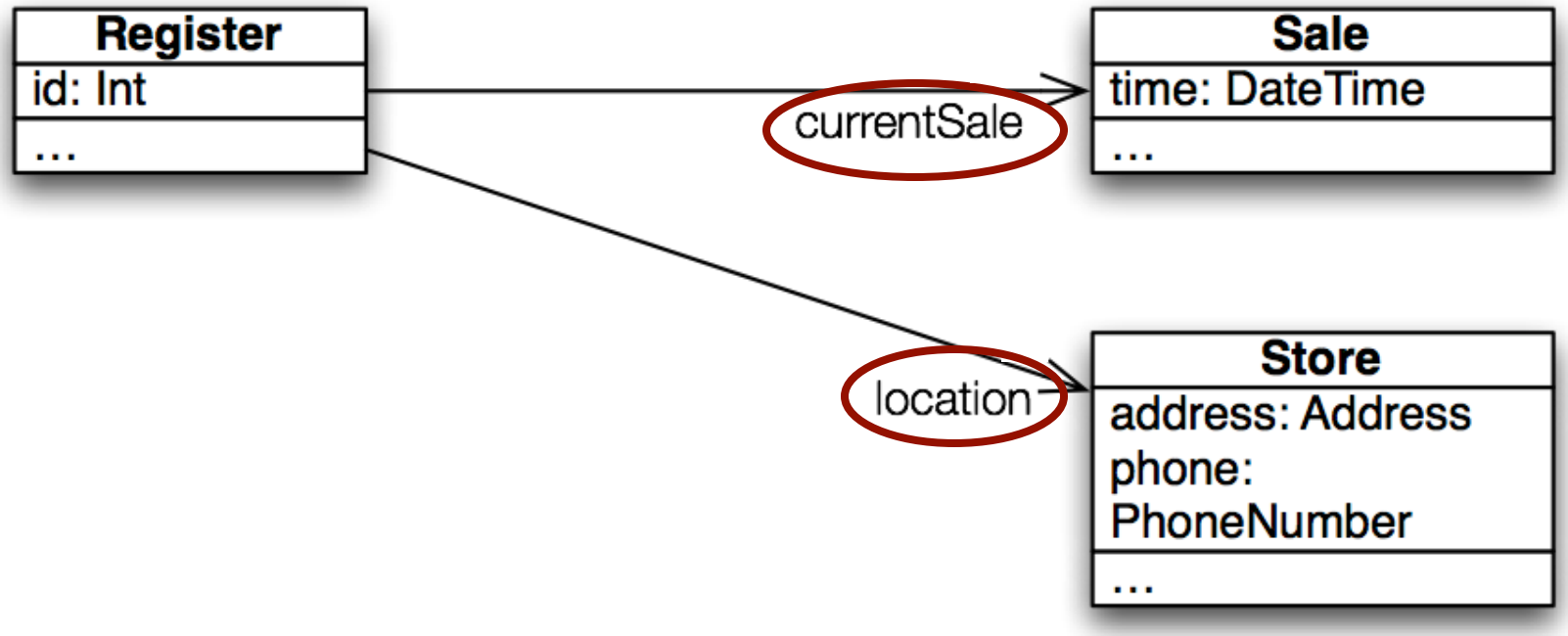
UML and Inheritance recap
`Object`

Check out *Inheritance2* from SVN

UML Class Diagram Preliminaryies

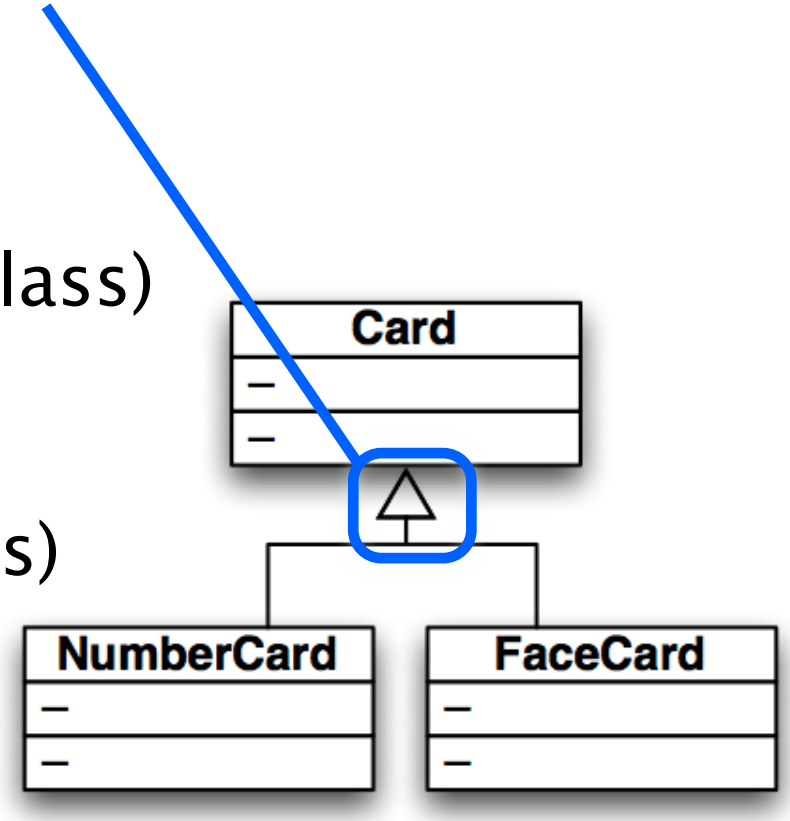
- Inheritance, Associations, and Dependencies

Recall UML: Associations



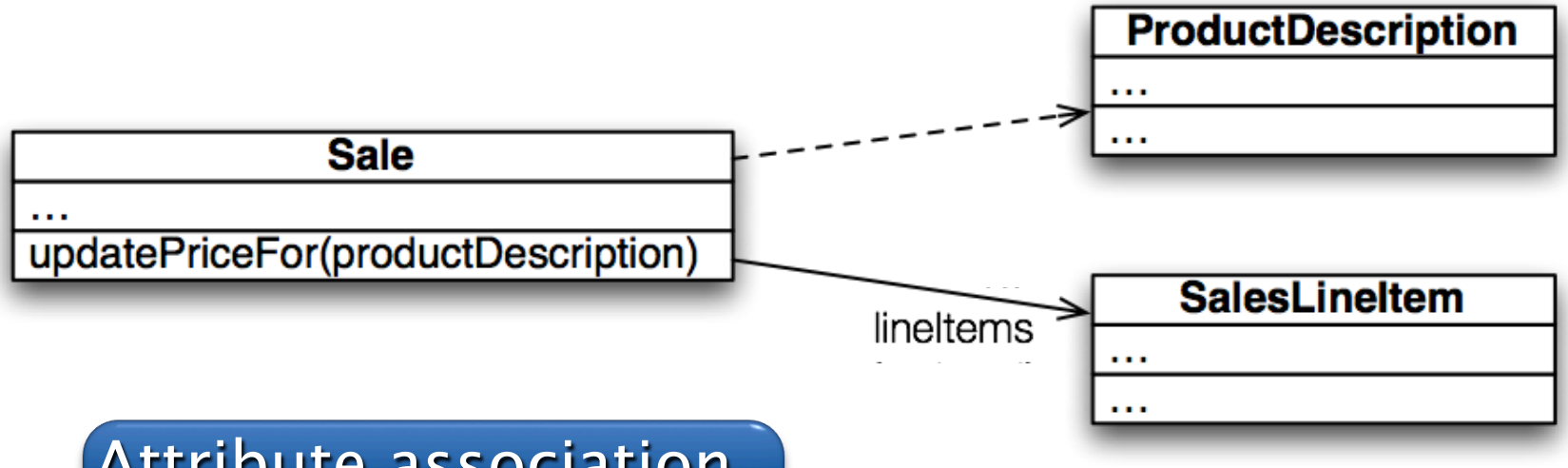
Recall UML: Inheritance

- Generalization (superclass)
- Specialization (subclass)



Recall UML: Dependencies

Dependency lines are dashed



Attribute association lines are solid

Use dependency lines when a more specific line type doesn't apply.

I, Object

- The superest class in Java

Object

Every class in Java inherits from ***`Object`***

- Directly and **explicitly**:
  - `public class String extends Object {…}`
- Directly and **implicitly**:
  - `class BankAccount {…}`
- **Indirectly**:
  - `class SavingsAccount extends BankAccount {…}`

Q1

`Object` Provides Several Methods

- `String toString()`
- `boolean equals(Object otherObject)`
- `Class getClass()`
- `Object clone()`
- …

Often overridden

Sometimes useful

Often dangerous!

Q2

Overriding `toString()`

- Return a concise, human-readable summary of the object state

- Very useful because it's called automatically:
  - During string concatenation
  - For printing
  - In the debugger

- **`getClass().getName()`** comes in handy here…

Q3

Overriding `equals(Object o)`

- `equals(Object foo)` – should return true when comparing two objects of same type with same "meaning"

- How?
  - Must check types—use **`instanceof`**
  - Must compare state—use **cast**

Recall casting a variable: Taking an Object of one particular type and "turning it into" another Object type

Q4

Polymorphism

- Review and Practice

Recall: Polymorphism and Subclasses

- A subclass instance **is a** superclass instance
  - Polymorphism still works!

```
BankAccount ba = new SavingsAccount();
ba.deposit(100);
```

- But not the other way around!

```
SavingsAccount sa = new BankAccount();
sa.addInterest();
```

- Why not?

Another Example

- Can use:

```
public void transfer(double amt, BankAccount o) {
    this.withdraw(amount);
    o.deposit(amount);
}
```

in `BankAccount`

- To transfer between different accounts:

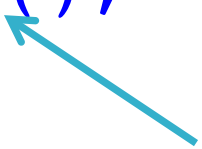
```
SavingsAccount sa = …;

CheckingAccount ca = …;

sa.transfer(100, ca);
```

Summary

- If B extends or implements A, we can write

```
A x = new B();
```

Declared type tells which methods x can access. Compile-time error if try to use method not in A.



The actual type tells which class' version of the method to use.

- Can cast to recover methods from B:

```
((B)x).foo()
```

Now we can access all of B's methods too.

If x isn't an instance of B, it gives a run-time error (class cast exception)

Exercise

- Do questions 5 through 7 from Quiz.
- Please had them in to TA when done and then start reading the BallWorlds specification on your schedule page.

Q5-7, hand in when done, then start reading BallWorlds spec

BallWorlds

- Meet your partner (see link in part 3 of spec)
- Carefully read the requirements and provided code
- Ask questions (instructor and TAs).

BallWorlds Teams same as GameOfLife pairs.

Look over the BallWorlds UML Class Diagram and start Questions.

Check out *BallWorlds* from SVN

BallWorlds Worktime

- Pulsar, Mover, etc.