

Inheritance and Polymorphism

Slides derived from the work of Dr. Amy McGovern and Dr. Deborah Trytten


Sharing Data Between Classes

Aggregation (*Has-A*) is one way to share data between classes

- Can only use public parts of the class
- Limitation or advantage?

Sharing Data Between Classes

Another way to share data is inheritance

- New class keyword: **extends**
 - Defines an inheritance relationship
 - UML: Arrow with open head 
- New method/data visibility keyword: **protected**
 - States that this data item/method is available both inside the class and to classes that extend this class
 - # in UML (as opposed to + or -)

Example

Online Ordering for Amazon

- Consider the following products and create a hierarchy
 - Products
 - Downloadable software
 - Software with media
 - Books

What is the UML?

Where Do These Properties Belong in the Hierarchy?

- Price
- URL for downloading software
- Name of item
- Author
- ISBN
- Delivery method
- Shipping costs

Terminology

- Subclass
 - Child class
- Superclass
 - Parent class
 - Base class

Terminology

- Subclasses get direct access to all of the public and protected data and methods from superclass
 - May have to implement methods again if we need more specific behavior

Consider equals()

Have you noticed that equals() works in a class, even if you didn't put it there?

```
public class Equalizer
{
    private int data;

    public Equalizer(int data)
    {
        this.data = data;
    }
}
```


Consider equals()

How does the program find an equals method in the Equalizer class?

Consider equals()

How does the program find an equals method in the Equalizer class?

- **public boolean equals(Object o)**

Consider equals()

Exercise:

- Demonstrate that this method is not working properly
 - Why?
- Fix it and demonstrate it
- Draw UML of Equalizer, both before and after

How about toString()

- What does toString() do? Or hashCode()?

Modeling Relationships

- The relationship represented by aggregation (with the diamond in UML) is “has-a”
- The relationship represented by inheritance (with the open headed arrow in UML) is “is-a”
 - More specialized classes are lower in the hierarchy

Modeling Relationships

Exercises:

- Example: Shape, Circle, Square, Ellipse, Rectangle, Quadrilateral
- Example: Student, Name, Address, City, State, Country, First Name, Last Name, Middle Name

Inheritance Can be Bad if Done Incorrectly

- Inheritance is widely used in Java
 - And all OOP languages
- Works fabulously in GUI components, and collections
- Inheritance breaks encapsulation if we use the *protected* keyword
- Aggregation/composition do not break encapsulation

Private or Protected Data?

Choosing private or protected can be a tough call

- If everything is private
 - Inheritance doesn't provide the subclass itself with anything it can't get through composition
 - However: the “user” of a class does get to see a consistent interface between the super and child classes

Private or Protected Data?

Choosing private or protected can be a tough call

- If everything is protected
 - Classes become closely coupled
 - Changes in one are likely to causes changes in the other
 - Bad for maintenance (\$\$\$)
- These effects can be mitigated somewhat through the use of multiple packages

Private or Protected Data?

Choosing private or protected can be a tough call

- My take: stick with private

Implementing Inheritance: Instance Methods and Variables

- `super.methodName()` to explicitly call public or protected methods in the superclass
 - For a given class, remember that there is exactly one superclass because Java does not allow multiple inheritance
- `super.instanceVariableName` to refer to public or protected instance variables from the superclass

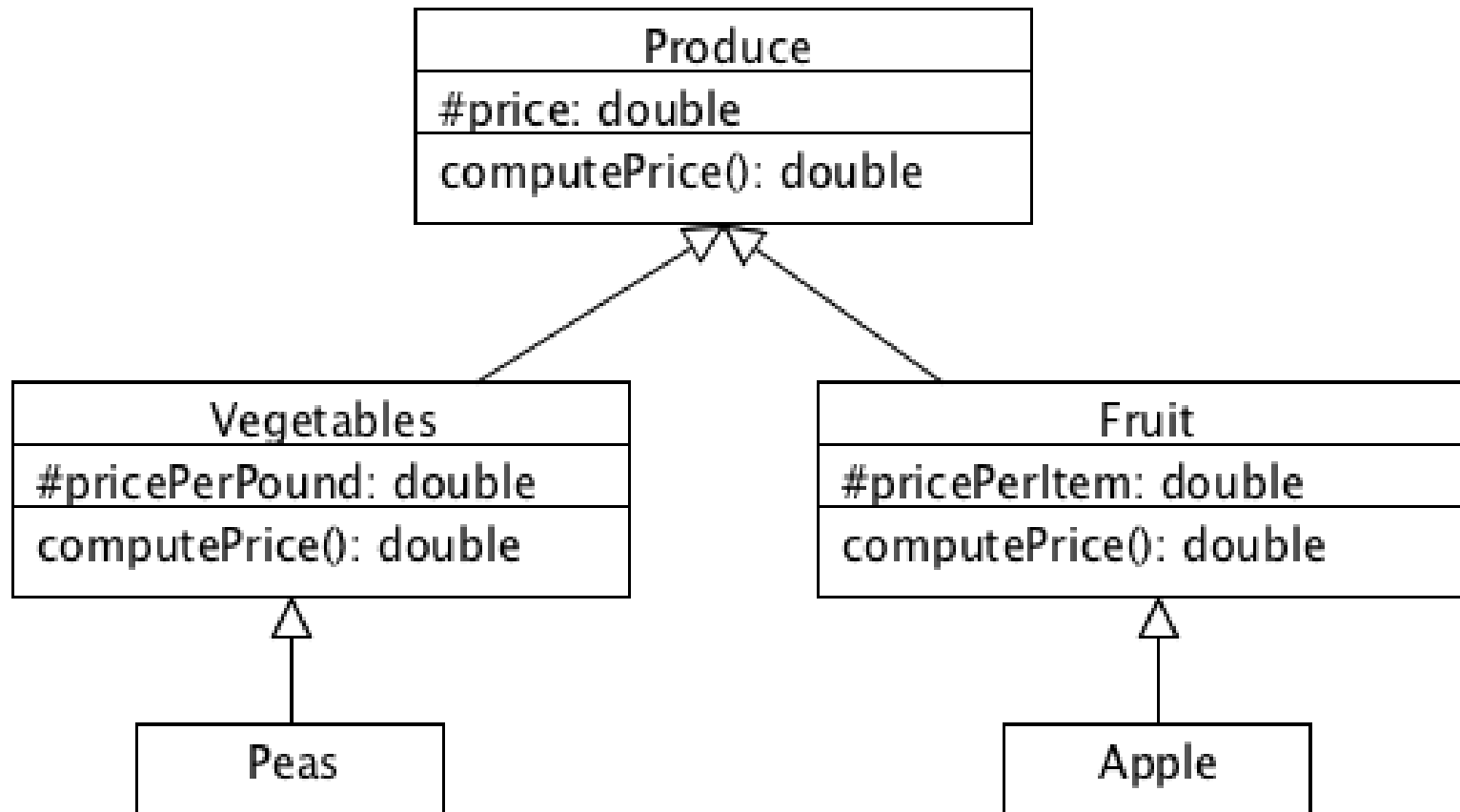
Implementing Inheritance: Constructor

- Constructors are not inherited
- But: can use `super()` to call the superclass constructor
 - If used, it must be first statement in subclass constructors
 - Can call any of the constructors associated with the superclass
- Most constructors call other constructors...

Compiler

- If you don't use `super()`, compiler adds implicitly for you
 - Why?
- All classes that allow inheritance must provide a no argument constructor
 - If you don't write one, the compiler adds a default

Inheritance example



Polymorphism

A variable of a super type can really be an instantiation of the sub type

```
Produce pr = new Apple();
```

This is called “Upcasting”

Polymorphism

- Calling methods: Java Virtual Machine will select data/method based on **object type at run time** (not the type of the reference)
 - Search order: constructed class if available, then parent, then grandparent, etc.

```
Produce pr = new Apple();  
pr.toString();           // Calls Apple.toString()
```

- Exercise: show example with Product hierarchy

Overriding Methods

When a subclass implements a method that is identical to one in the superclass it is **overridden**

- Method must be public or protected
- Same name
- Same parameters
- Return values: new method must return a subclass of the original method's return type
- Static methods cannot be overridden

Down-Casting

The other way can be made to work, but we need to be explicit:

```
Apple a = pr;    // Compiler disallows
```

```
Apple a = (Apple) pr;    // Allowed
```

- Forces java to treat the object as if it is the subclass
- Lets you access subclasses methods
- If you improperly cast an object, you will receive Exceptions when you try to access the object

Casting and isinstance

isinstance will tell you what class an instance is:

```
if (pr isinstance Apple) {  
    Apple a = (Apple) pr;  
    // Use a....  
  
}
```

ArrayList example

Exercise: make an ArrayList of Produce and Fruit

- What can go in each?
- Printing out the lists

Immutable Classes and Inheritance

- It is possible to make a class so that it cannot be inherited from

```
public final class ClassName
```

- This must be done with all immutable classes
 - Why?
- Again, if unsure, make class final
 - Can always remove it later
 - Once you let people extend a class, you can't make changes or risk breaking their code

