Designing with Aggregation and Inheritance

Software Design

Step 1
Identify the objects.

Step 2
Find/design a class for each object.

One of the Challenges
deciding when to use aggregation and when to use inheritance

<table>
<thead>
<tr>
<th>SupplierClass</th>
<th>SuperClass</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ClientClass</th>
<th>SubClass</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

What is AGGREGATION?

```java
public class SomeAggregate {
    private ClassA partA;
    private ClassB partB;
    private ClassC partC1;
    private ClassC partC2;
    // methods omitted
}
```

```
SomeAggregate

ClassA  ClassB  ClassC
```

© 2006 Pearson Addison-Wesley. All rights reserved
```java
public class SomeAggregate {
    private ClassA partA;
    private ClassB partB;
    private ClassC partC1;
    private ClassC partC2;
    // methods omitted
}
```

### Class Diagram

![Class Diagram](imageURL)

### Object Diagram

![Object Diagram](imageURL)

### When to aggregate?

One class **contains_a** other.

- An equilateral triangle **contains_a** vertex.
- A dog **has_a** tail.
- A sandal **includes_a** sole.
- A company’s work roster **contains_an** employee.
- Cherry pie **contains_a** cherry pit.

**contains_a**
When to inherit?

One class is_a another class

An equilateral triangle is_a triangle.
A dog is_an animal.
A sandal is_a special kind of footwear.
An hourly employee is_an employee.
Cherry pie is_a dessert.

is_a

Generalizations can be discovered from common attributes.

Create the best relationship from...

camcorder
record button
tape recorder
lens
on/off switch
viewfinder

record button
tape recorder
on/off switch
lens
viewfinder

camcorder
lens
viewfinder
```java
public class Camcorder {
    private TapeRecorder rec;
    
    //somewhere in the code
    rec.startRecording();
}
```

```java
public class Camcorder extends TapeRecorder {
    
    //somewhere in the code
    startRecording();
}
```

---

**java.awt.Container**

Container is an awt class well suited to inheritance.

A Container object is a transparent rectangle. It is useful for grouping other graphical objects into a single unit.

An Container subclass can reuse...
- `add` and `remove`
- `setBounds` and `setLocation`
- `getX`, `getY`, `getWidth`, `getHeight`, `getBackground`
- `repaint`

A Container subclass may want to override...
- `setSize` and `setBounds`
- `setBackground`