Software Engineering

Lesson Design Pattern 11
Flyweight
v1.0

Uwe Gühl

Fall 2007/ 2008

Used sources:

- [GHJ04] Gamma, Helm, Johnson, Vlissides: Design Pattern, Addison Wesley, 2004
- [Hus08] Vince Huston: Design Pattern, www.vincehuston.org/dp/, 2008
- [Wik08] http://en.wikipedia.org/wiki/Flyweight_pattern, 2008

Flyweight

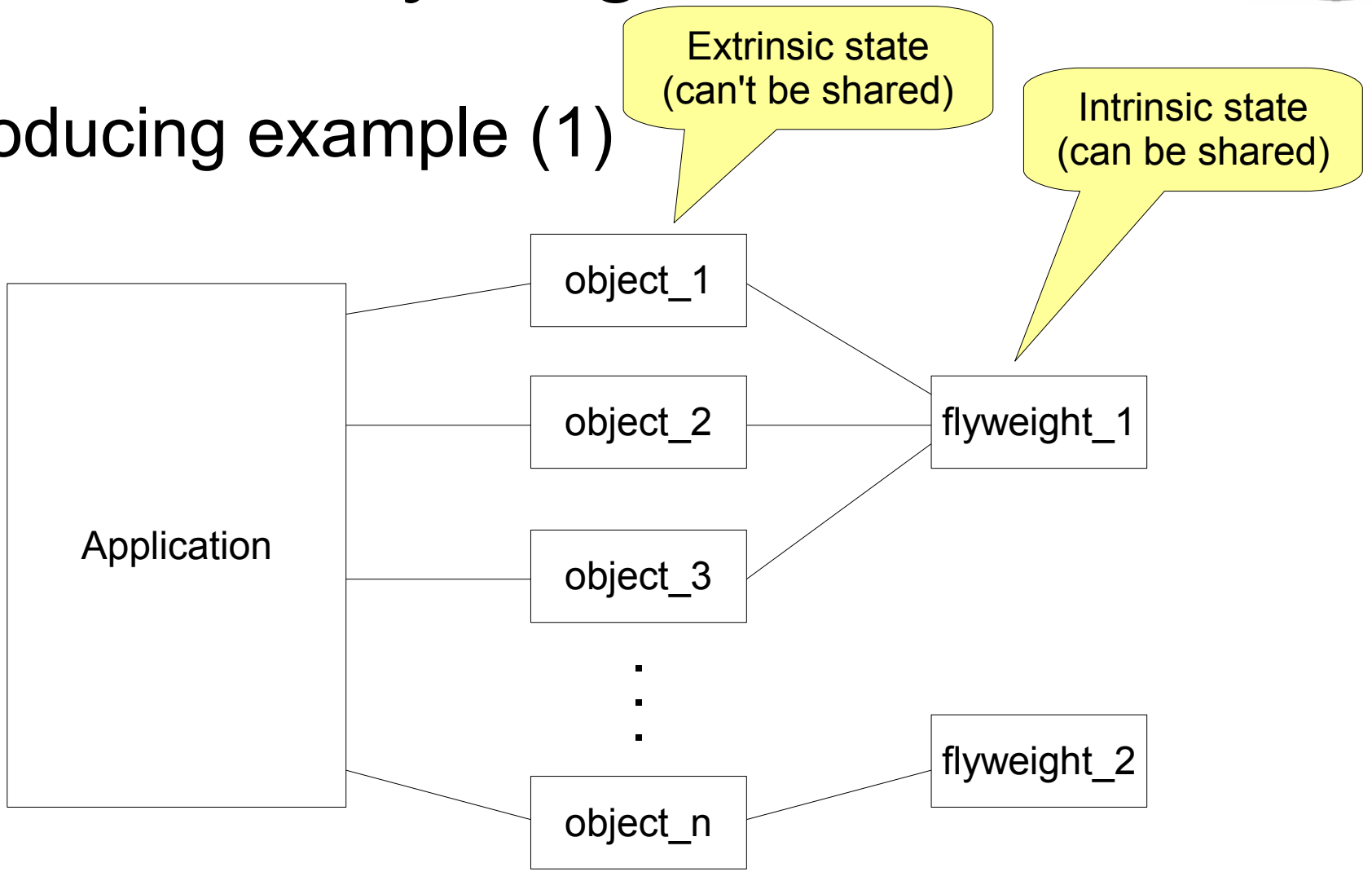
- Intent:
  - Use sharing to support large numbers of fine-grained objects efficiently
  - minimizes memory occupation by sharing as much as possible with other similar objects
  - ... is a Structural Pattern

Flyweight

- Motivation
  - Problem: If you use really many objects of a specific class a naive implementation would be really expensive
  - Example: Object oriented document editors
    - Tables and figures as objects
    - characters as objects? ... usually not
      - would be nice because you get flexibility even in the finest level
      - new characters could be supported easily
      - drawback: costs in memory and run-time overhead – maybe hundreds of thousands of character objects needed
  - Idea: Using the ***Flyweight*** pattern

Flyweight

- Introducing example (1)



Flyweight

- Motivation
  - A flyweight is a shared object that can be used in multiple contexts simultaneously.
  - A flyweight acts as an independent object in each context

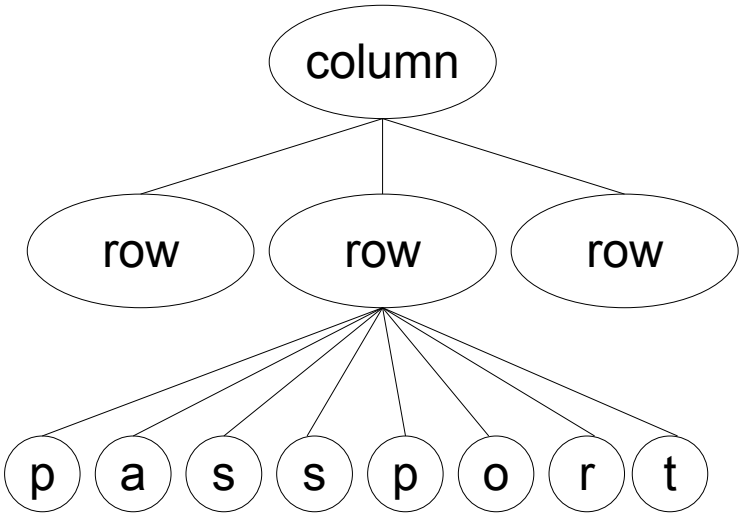
Flyweight

- Motivation
  - Key concept:
    - Intrinsic state (context independent)
      - stored in the flyweight
      - consists information that is independent of the flyweight's context
      - sharable
    - Extrinsic state (context dependent)
      - depends on and varies with the flyweight's context
      - can't be shared
      - Client objects are responsible for passing extrinsic state to flyweights when requested

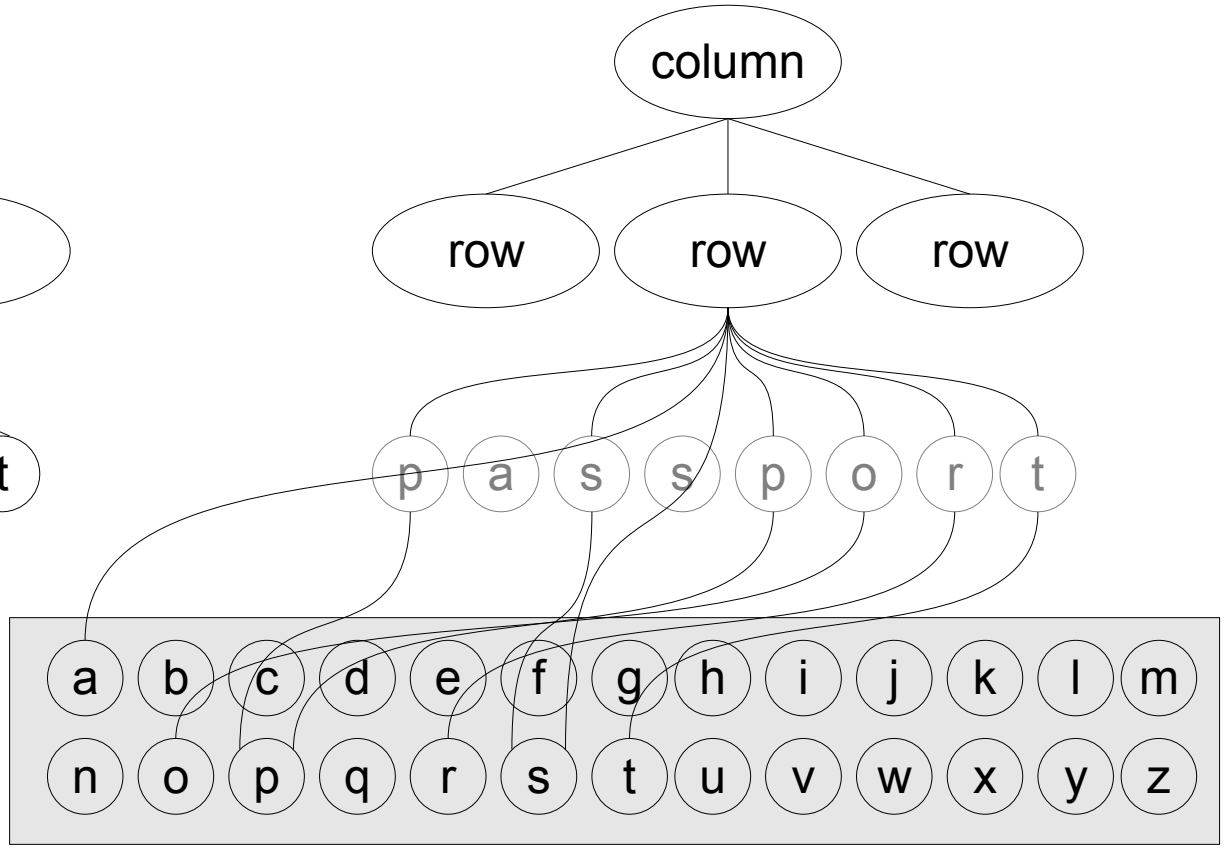
Flyweight

- Introducing example (2)

Logically:



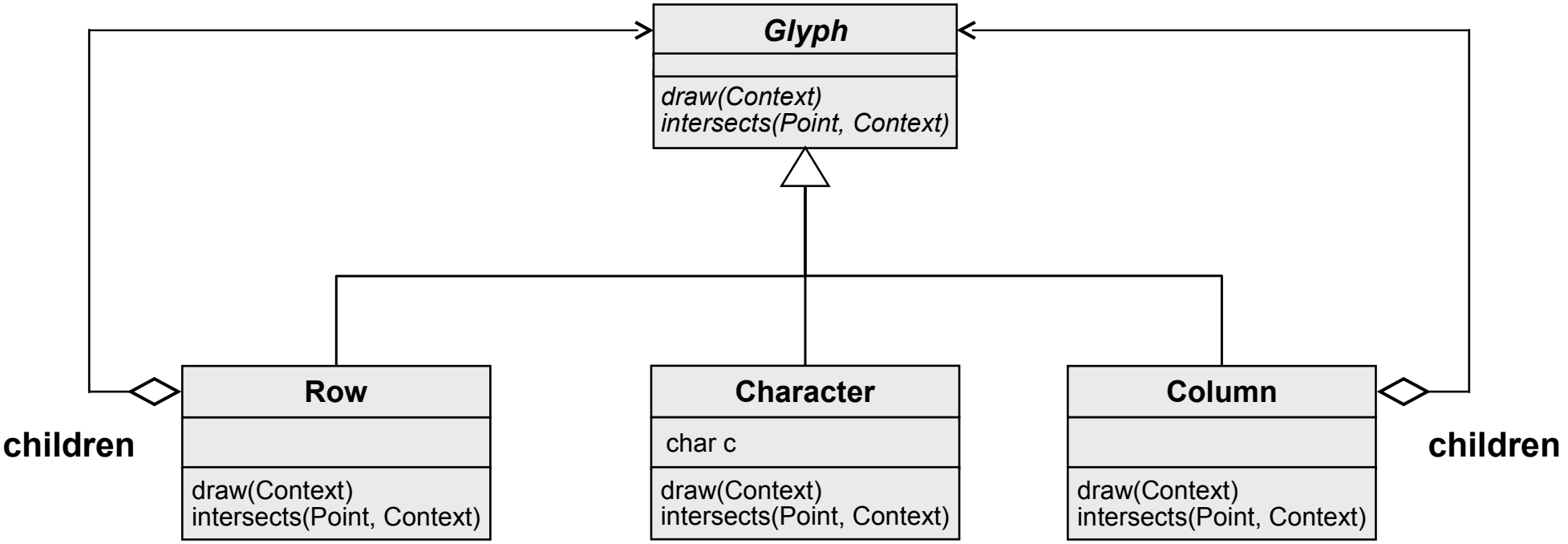
Physically:



flyweight pool

Flyweight

- Introducing example (2)
  Class structure

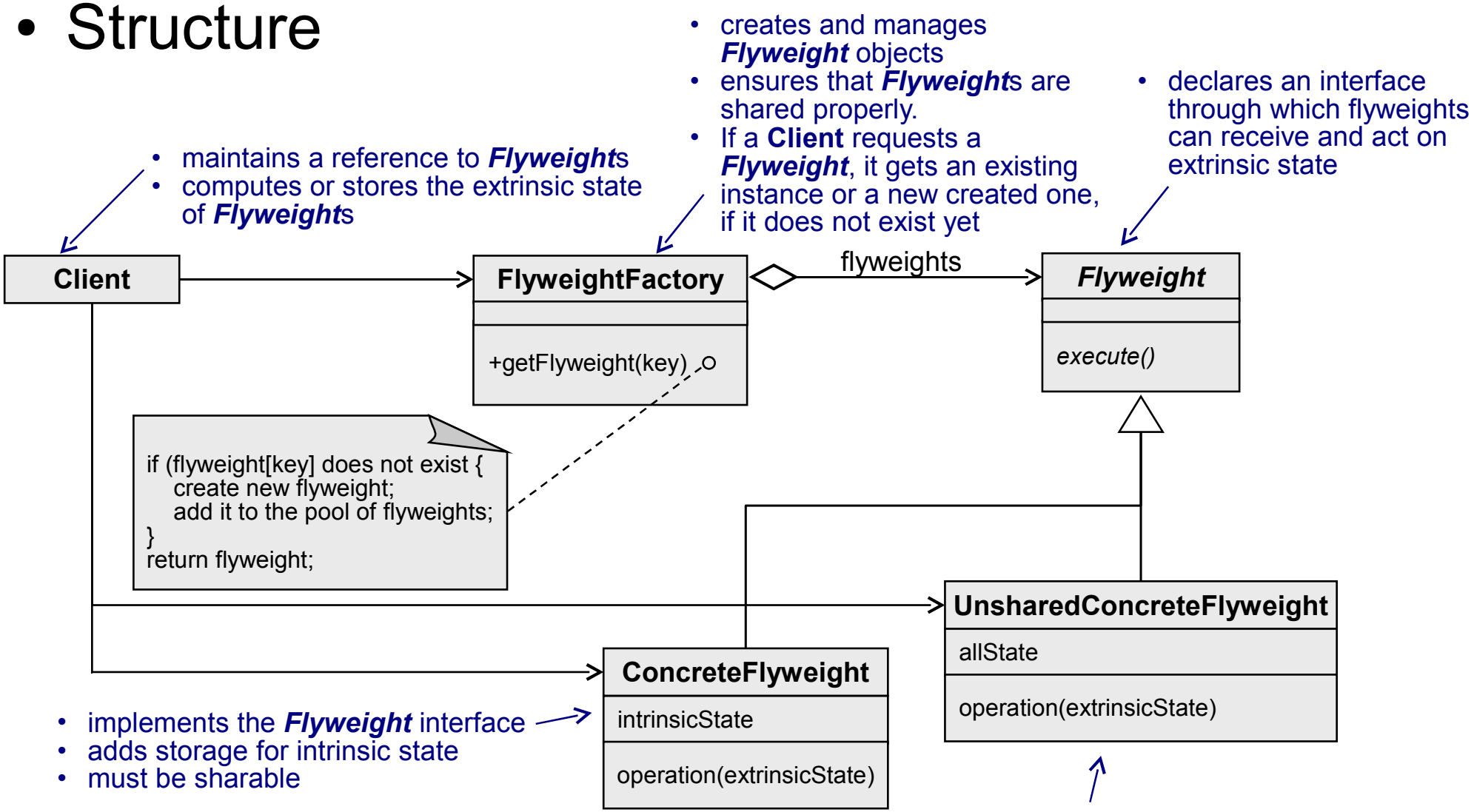


Flyweight

- Introducing example (2)
  - For every character exists a reference to a glyph object shared by every instance of the same character in the document
  - The position of each character (in the document and/or the page) needs to be stored externally

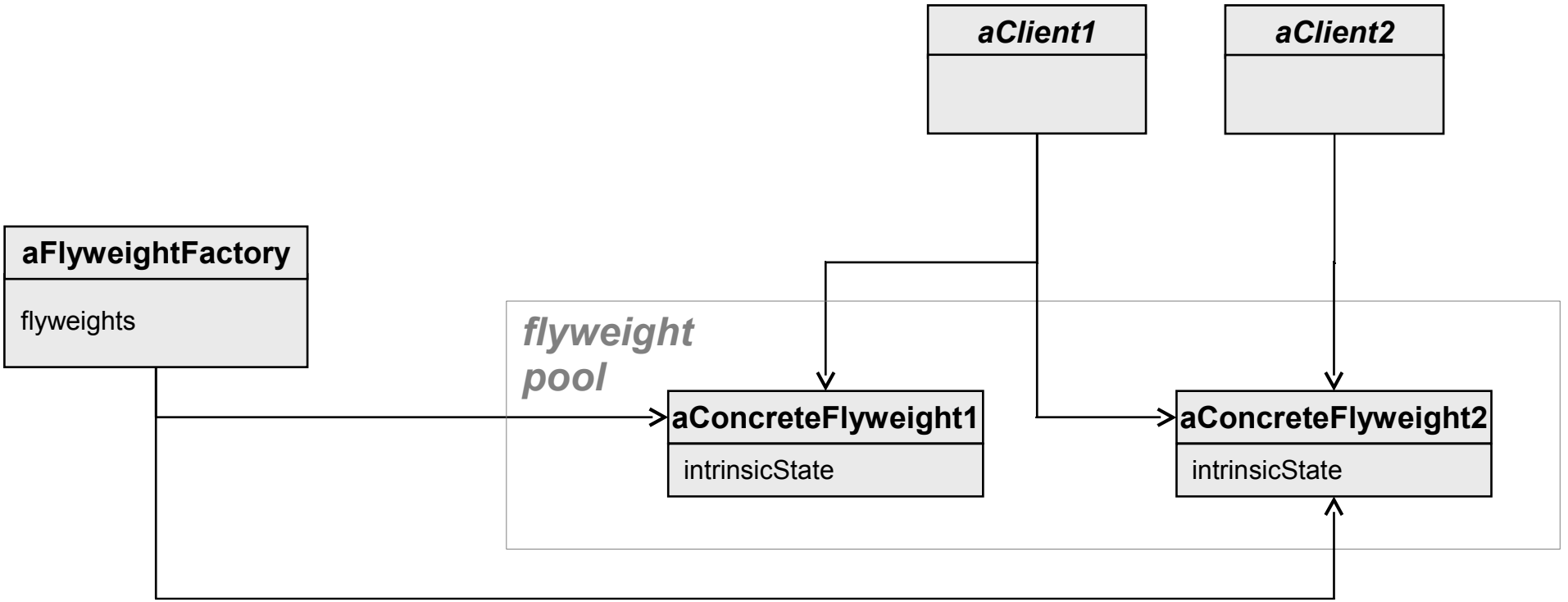
Flyweight

- Structure



Flyweight

- Structure
  ... how flyweights are shared



Flyweight

- Collaboration
  - To make a **Flyweight** work the basic states have to be characterized
    - either as intrinsic state
      - stored in the **ConcreteFlyweight** object
    - or as extrinsic state
      - stored or computed by Client objects
      - Clients pass this state to the flyweight when they invoke its operations
  - Clients should not instantiate **ConcreteFlyweight** objects directly – they should use the **FlyweightFactory** to ensure they are shared properly

Flyweight

- Applicability
  Use the Flyweight Pattern if all of following statements are true
  - A large number of objects is involved
  - Storage costs for these objects are high
  - Most object state information could be made extrinsic
  - Many groups of objects may be replaced by few shared objects
  - Object identity is not important

Flyweight

- Consequences

+ **Flyweight** enable space savings – enhanced by
  - more **Flyweight**s being shared
  - possible reduction in the total number of instances
  - increasing the amount of intrinsic state per object
  - computing rather storing the extrinsic state?
+ The more shared **Flyweight**s, the more savings
– **Flyweight**s may introduce run-time costs associated with
  - transferring, finding, and / or computing extrinsic state

Flyweight

- Implementation
  - Removing extrinsic state
    - Ideally extrinsic state can be computed, so we have far smaller storage requirements
  - Managing shared objects
    - Clients should not instantiate them directly, because objects are shared
    - Creation of flyweights on demand
    - Suggestion: Storage of a mapping in a (hash) table

Flyweight

- Implementation – Check list [Hus08]
  1. Ensure: Object overhead is an issue and the client of the class could take realignment responsibility
  2. Divide the target class's state into: shareable (intrinsic) state, and non-shareable (extrinsic) state.
  3. Remove the non-shareable state from the class
  4. Create a Factory that manages class instances
  5. Client uses the Factory to request objects.
  6. The client must look-up or compute the extrinsic state, and supply that state to class methods.

Flyweight

- Known Uses (see [GHJ+95])
  - InterViews 3.0: The concept of flyweight objects was first described
  - ET++ system: support look-and-feel independence
  - XHTML uses a a common attributes parameter entity that is a Flyweight
  - The XML & SGML Cookbook page 1-126 mentions the Flyweight pattern

Flyweight

- Related Patterns [Hus08] [p. 200, 206 GHJ+95]
  - Flyweight is often combined with Composite to represent a hierarchical structure as a graph with shared leaf notes
    - Using a flyweight disables the possibility to store a pointer to the parents

Flyweight

- Related Patterns [Hus08] [p. 138, 206, 255 GHJ +95]
  - State and Strategy pattern could be implemented as flyweights
  - Terminal symbols within Interpreter's abstract syntax tree can be shared with Flyweight
  - Whereas Flyweight shows how to make lots of little objects, Facade shows how to make a single object represent an entire subsystem