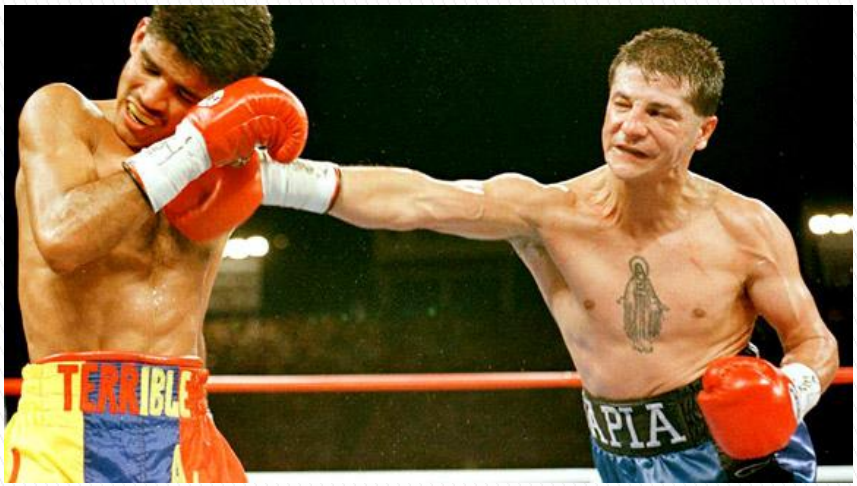
Flyweight

Many objects with some shared properties

Michael Abramowitz

Presentation Layout

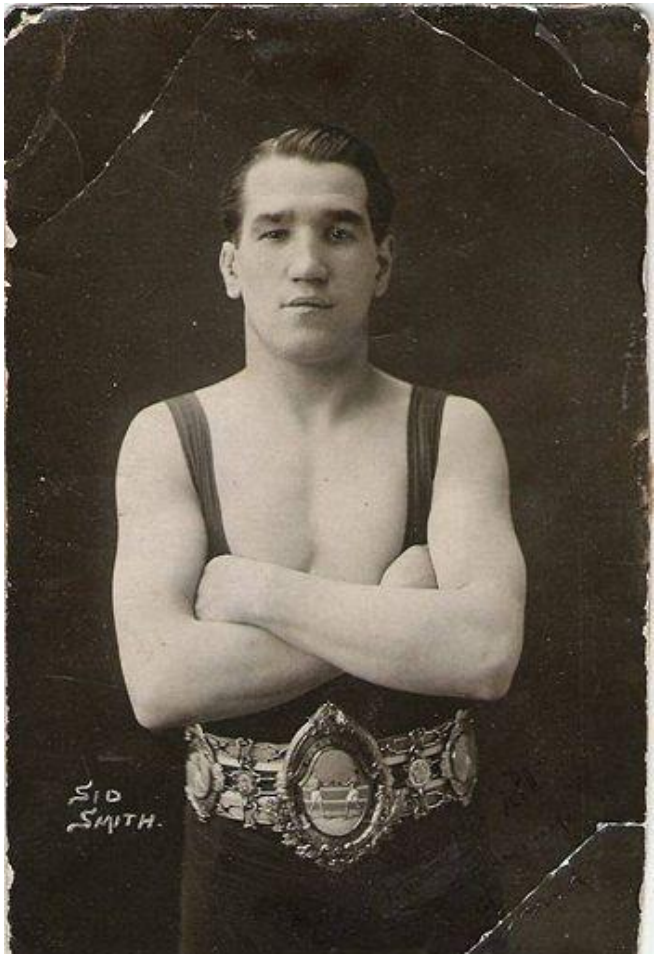
- Elements of the Flyweight
  - Classification / Name
  - Problem
  - Solution
  - Elements of Pattern
- Forest Example



Elements of the Flyweight – Classification / Name

- Structural Object Pattern
  - Deals with the composition of objects
  - Use pattern as a way to assemble objects
  - Ease design by identifying a simple way to realize relationships between entities
- Flyweight
  - Organize large number of fine grained objects efficiently
  - Name is derived from boxing weight class
  - Basically makes objects lighter by storing shared info about objects in one place (the flyweight)

Sid Smith (First Flyweight Champion)



Elements of Flyweight – Problem

- Program requires a large number of objects
- Storage costs high as a result
- Examples:
  - Word Processor (common usage)
    - Each character as an object
      - A-z : flyweights
  - Seeing the Forest for the Trees (my example)
    - Each tree as an object
    - Tree Type : flyweight

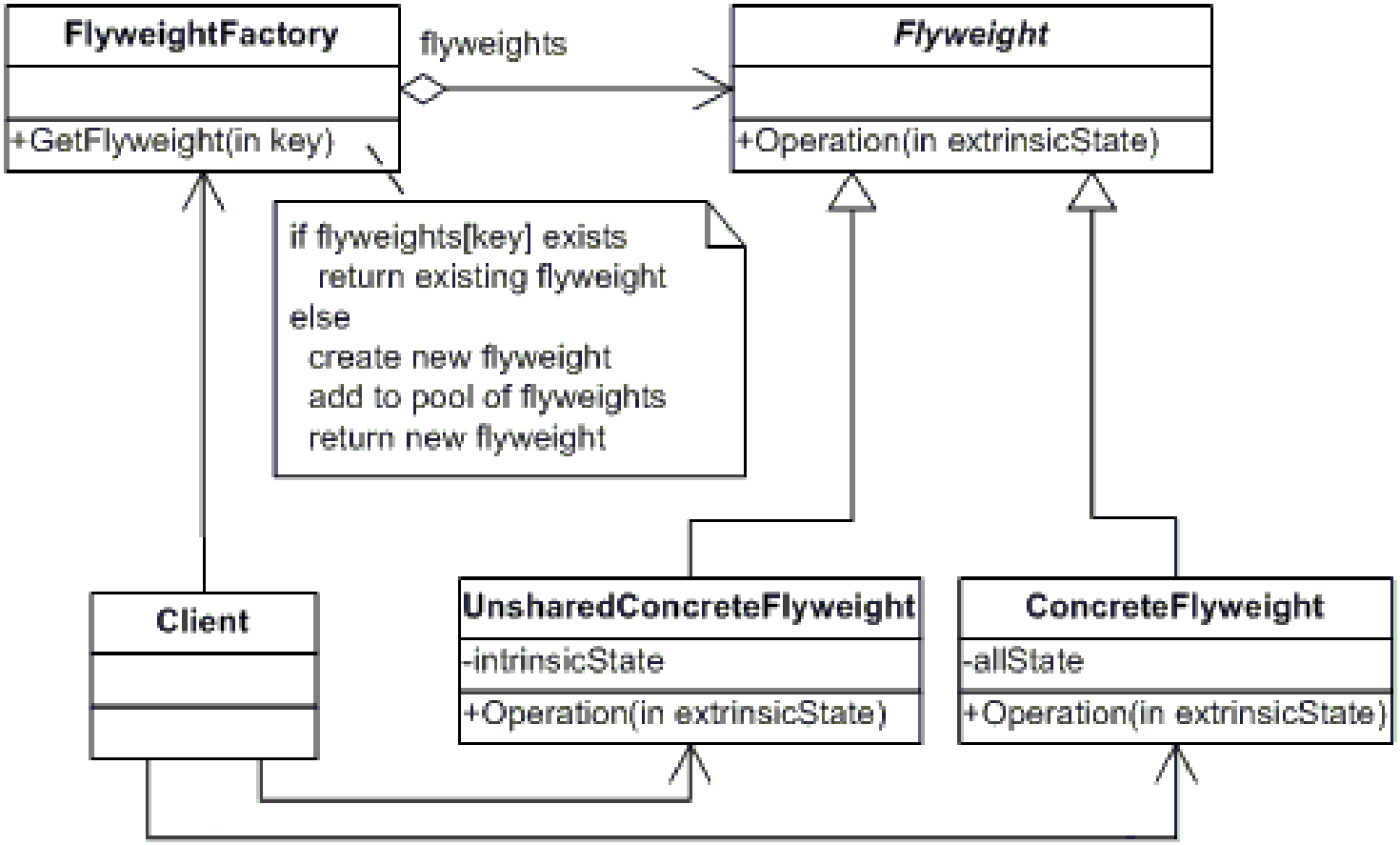
Elements of Flyweight – Solution

- Separate object properties by following
  - INTRINSIC
    - Information that is constant
    - Can be removed from each individual granular object
    - STORED IN FLYWEIGHT OBJECT
    - Referenced by granular objects with the properties
  - EXTRINSIC
    - Information must is determined by context
    - Values that can be calculated on the spot
    - STORED IN CLIENT ALONG WITH REFERENCE TO FLYWEIGHT
- Objects are organized by intrinsic information similarities
- Allows creation of many objects differing only in context

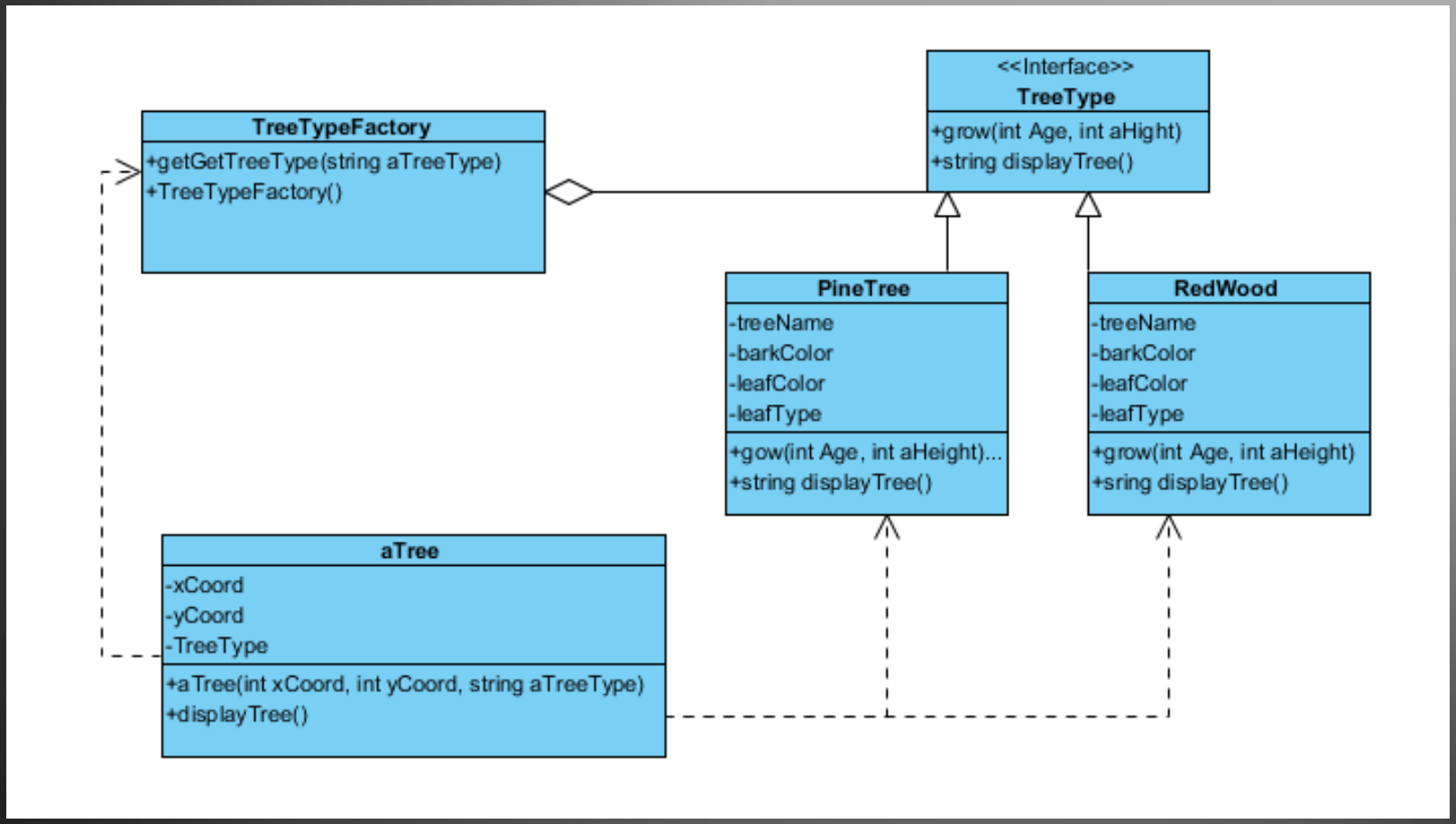
Elements of Flyweight Pattern

- Flyweight Interface
  - Abstract – set of methods all flyweights will have in common
  - EXAMPLE: TreeType Interface
- Flyweight
  - Inherits of Flyweight Interface
    - Contains all intrinsic info, as well as how to calculate extrinsic info
  - EXAMPLE: PineTree and RedWood
- Flyweight Factory
  - Dispense flyweights when requested)
  - EXAMPLE: TreeTypeFactor
- Client
  - In creating a new object, assigns flyweight to it. Using Flyweight Factory method
  - Combines intrinsic reference and extrinsic state
  - EXAMPLE: aTree

From Book


FlyweightFactory
+GetFlyweight(in key)
flyweights
Flyweight
+Operation(in extrinsicState)
if flyweights[key] exists
return existing flyweight
else
create new flyweight
add to pool of flyweights
return new flyweight
Client
UnsharedConcreteFlyweight
-intrinsicState
+Operation(in extrinsicState)
ConcreteFlyweight
-allState
+Operation(in extrinsicState)

Forest Example



Forest Example

- TreeType
  - (FlyWeight interface)
  - (interface)
- RedWood
  - FlyWeight
  - Implements TreeType
- PineTree
  - FlyWeight
  - Implements TreeType
- TreeTypeFactory
  - FlyWeight Factroy
  - Returns RedWood or PineTree object
  - Same instance each time
- aTree
  - Client
  - Has TreeType fields that uses factory
  - xCoord, yCoord are extrinsic
- ForestDriver
  - Creates a Forest Catalog

EXAMPLE CODE AND OUTPUT

- ForestDriver.java
  TreeType.java
  PineTree.java
  RedWood.java
  TreeTypeFactory.java
  aTree.java