Team 302
Design Patterns

Design Patterns

- Solutions to software design problems you find again and again in real-world application development
- Patterns are about reusable designs and interactions of objects.
- Contains “Best Practices”
- Gang of Four (GoF) released the first ones

Design Patterns

- Gang of Four
  - Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides
  - 23 Design Patterns released in 1994 Book
    - Program for interface vs. implementation
    - Composition vs. Inheritance
      - More flexible
      - “Has a” is better than “Is a”
  - Three categories of Patterns
    - Creational
    - Structural
    - Behavioral

Design Patterns - References

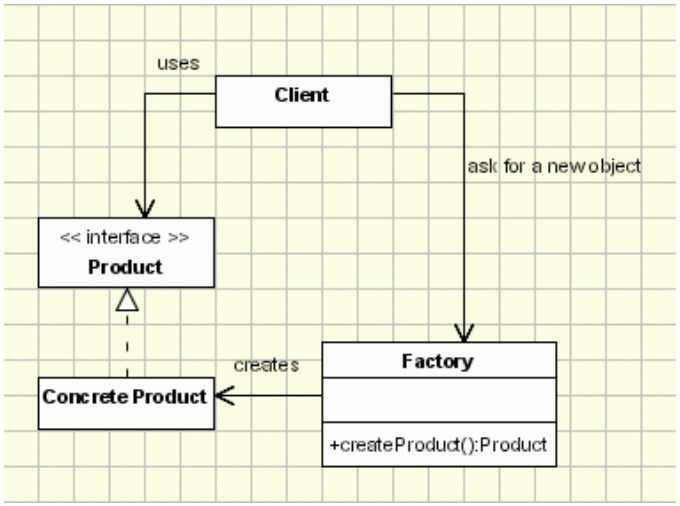
- http://gameprogrammingpatterns.com/contents.html
- http://www.dofactory.com/net/design-patterns
- http://www.oodesign.com/
- https://sourcemaking.com/design_patterns

Design Patterns - Creational

- Creates objects while hiding the creation logic
- Patterns:
  - Abstract Factory
  - Builder
  - **Factory**
  - Prototype
  - **Singleton**

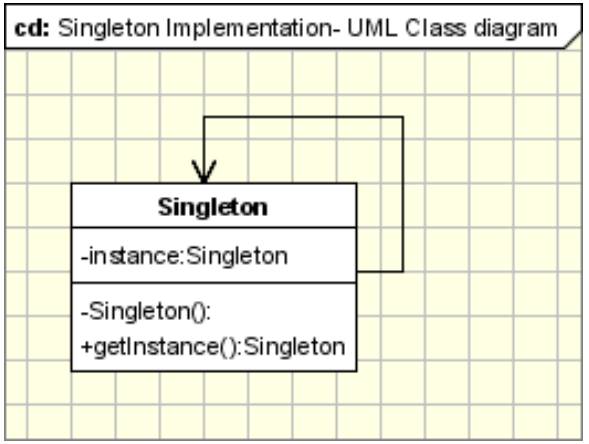
Factory Pattern

- Interface for creating object
- Factory calls creator
- Interaction with the interface of created object
- Example



Singleton Pattern

- One instance of the class
- Careful not to abuse (limits multithreading capability)
- Example

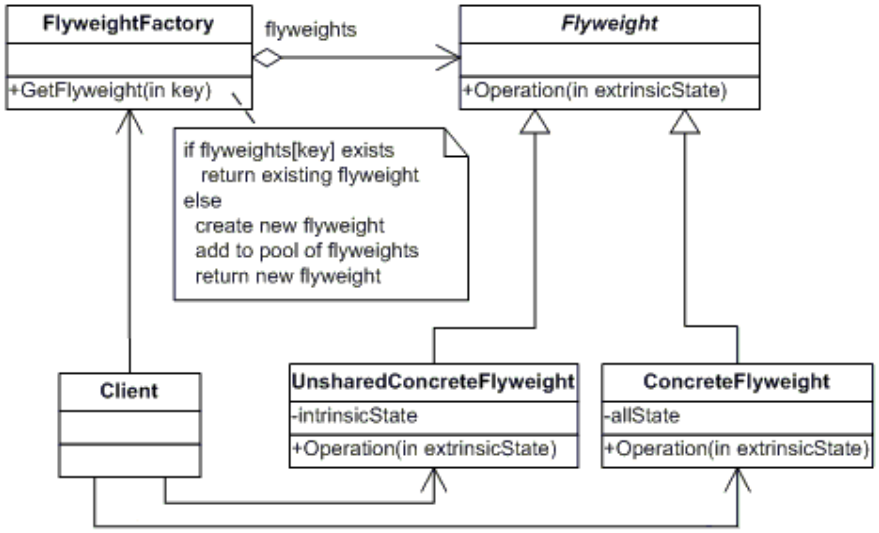


Design Patterns - Structural

- Related to class and object composition
- Inheritance is to compose interface
- Define ways to compose objects into new functionality
- Patterns:
  - Adapter
  - Bridge
  - Composite
  - Decorator
  - Facade
  - **Flyweight**
  - **Proxy**

Flyweight Pattern

- Use Sharing to support large number of objects efficiently
- Example

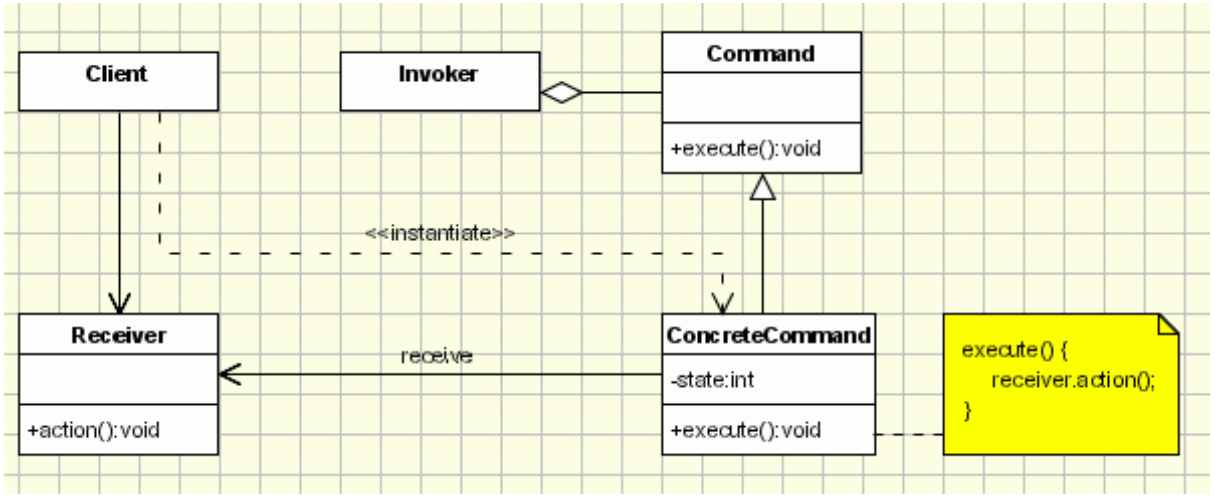


Design Patterns – Behavioral

- Communication between objects
- Patterns:
  - Chain of Responsibility
  - **Command**
  - Interpreter
  - Iterator
  - Mediator
  - Momento
  - Observer
  - **State**
  - **Strategy**
  - Template
  - Visitor

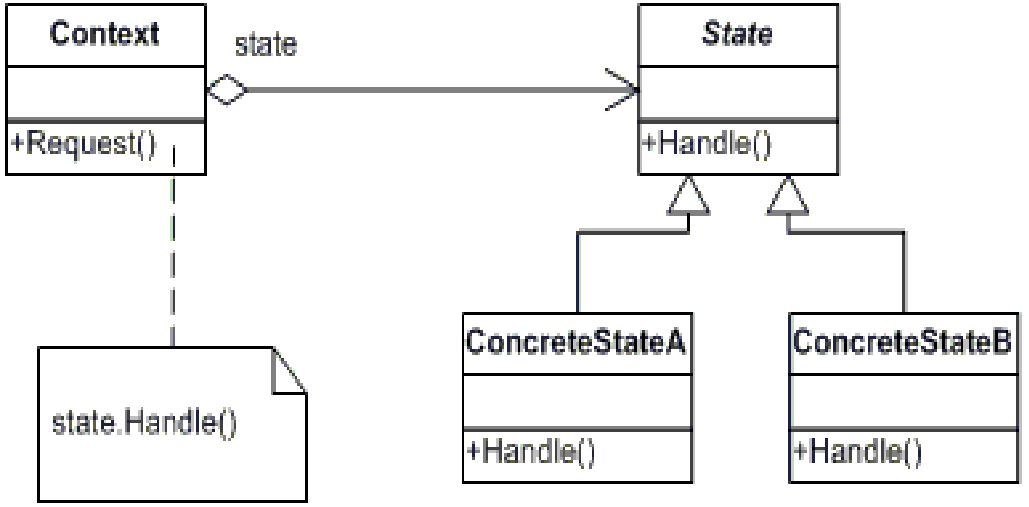
Command Pattern

- FRC JAVA/C++ WPILIB uses
- Encapsulate a request as an object
- Clients can handle different requests generically
  - Easier to customize gamepad buttons
- Supports undo type processing
- Example



State Pattern

- Behavior changes based on state
- Each State is a Class
- Example



Strategy Pattern

- Family of algorithms
- Encapsulate each one
- Interchangeable
- Example