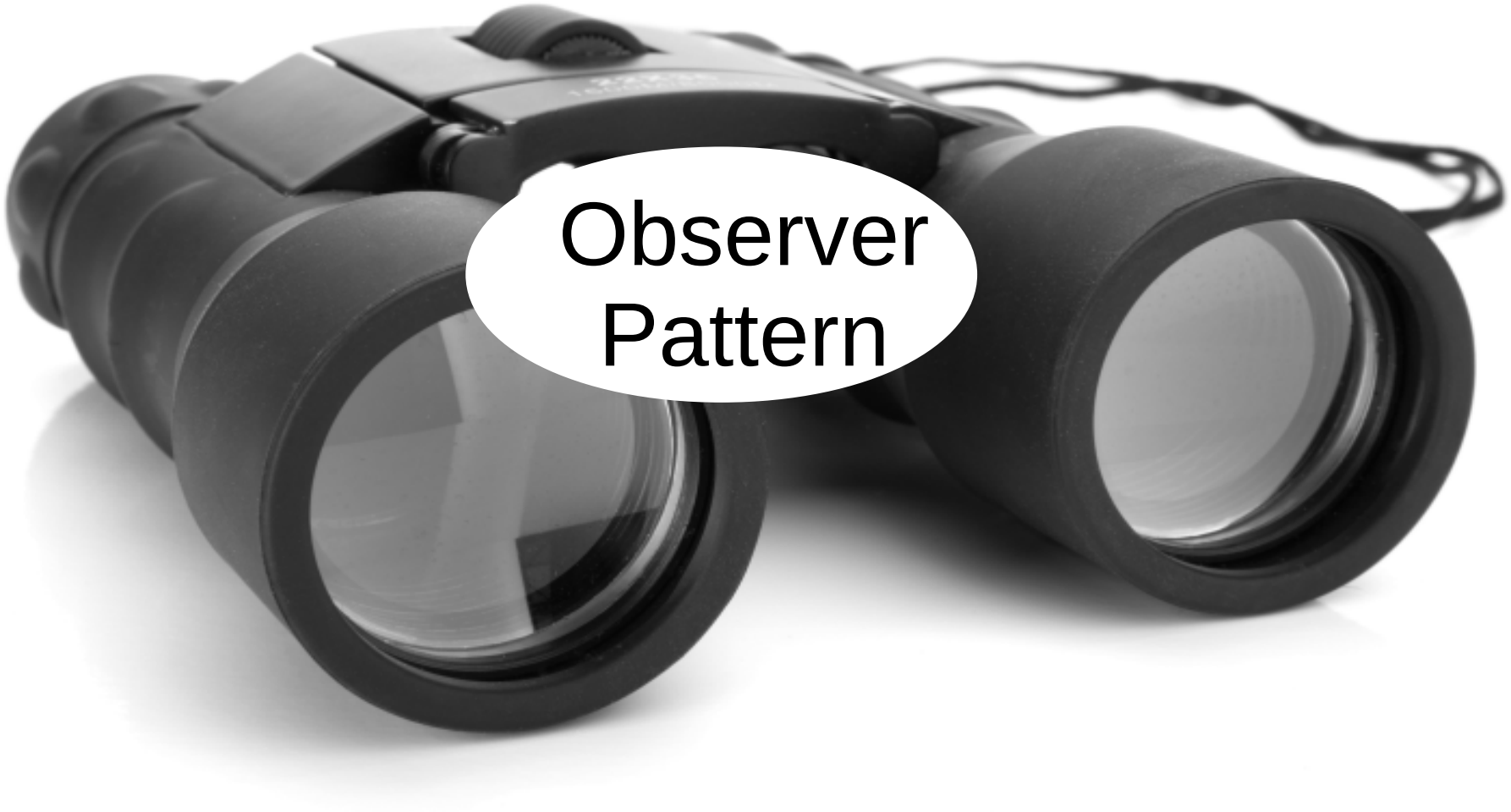
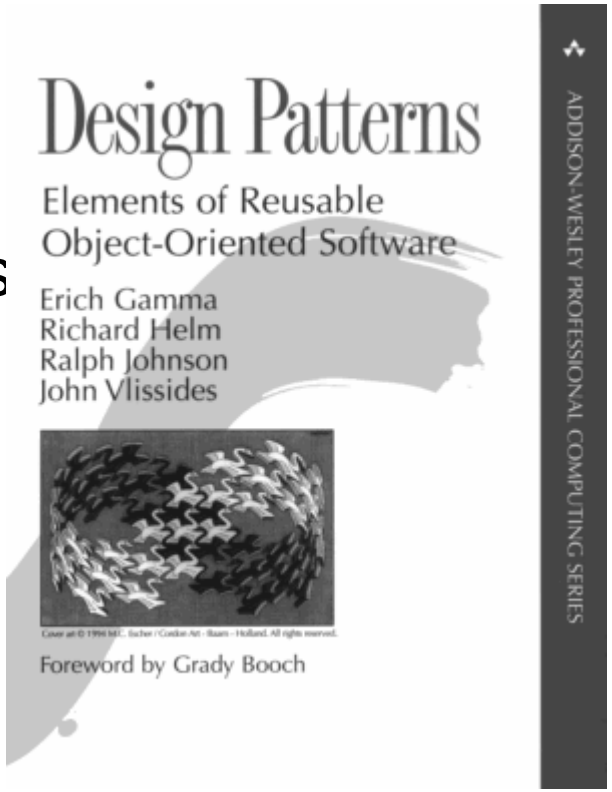


Pattern

- Software Design Pattern:
  - **a description of a common software design problem and the essence of its solution**
  - Allows discussion, implementation, and reuse of proven software designs
- Gang of Four
  - A pioneering book on design patterns by 4 authors: Gamma, Helm, Johnson, Vlissides.



Observer pattern motivation

For billionaires!

- Imagine you are writing an automatic day-planner:
  - It reads in the user's interests, plus information about the world, and suggest what they should do.
- Possible design idea:
  - You want to use different objects for cultural planning, sports planning, and sight-seeing.
  - Some objects bring in information about the world; your planning-objects use these info objects.
- Challenge:
  - All of these objects need to know the weather.
  - Your weather object gets updates now and then.
  - How do you tell.. **all the objects new data is available?**

Possible Idea

- Have the weather object call each info. object:

```
class Weather
    void newDataUpdate() {
        String weatherData = ...;
        culturePlanner.update(weatherData);
        sportsPlanner.update(weatherData);
        sightseeingPlanner.update(weatherData);
        // Change here EVERY time you get a new planner.
    }
```

- Bad because:
  - Weather object is...

    tightly coupled to every planner!
  - Every new planner you get, you'll have to change the weather object's code, recompile, and re-run.

The observer pattern

- Observer Pattern:

  It allows objects to "register for updates with another object at run-time

- Produces a one to many relationship:
  - one object observed (called the subject)
  - many objects observing (called the observers).
- Great because it loosely couples objects:
  - Object with something to report does not need a hard-coded list of who to tell; ...

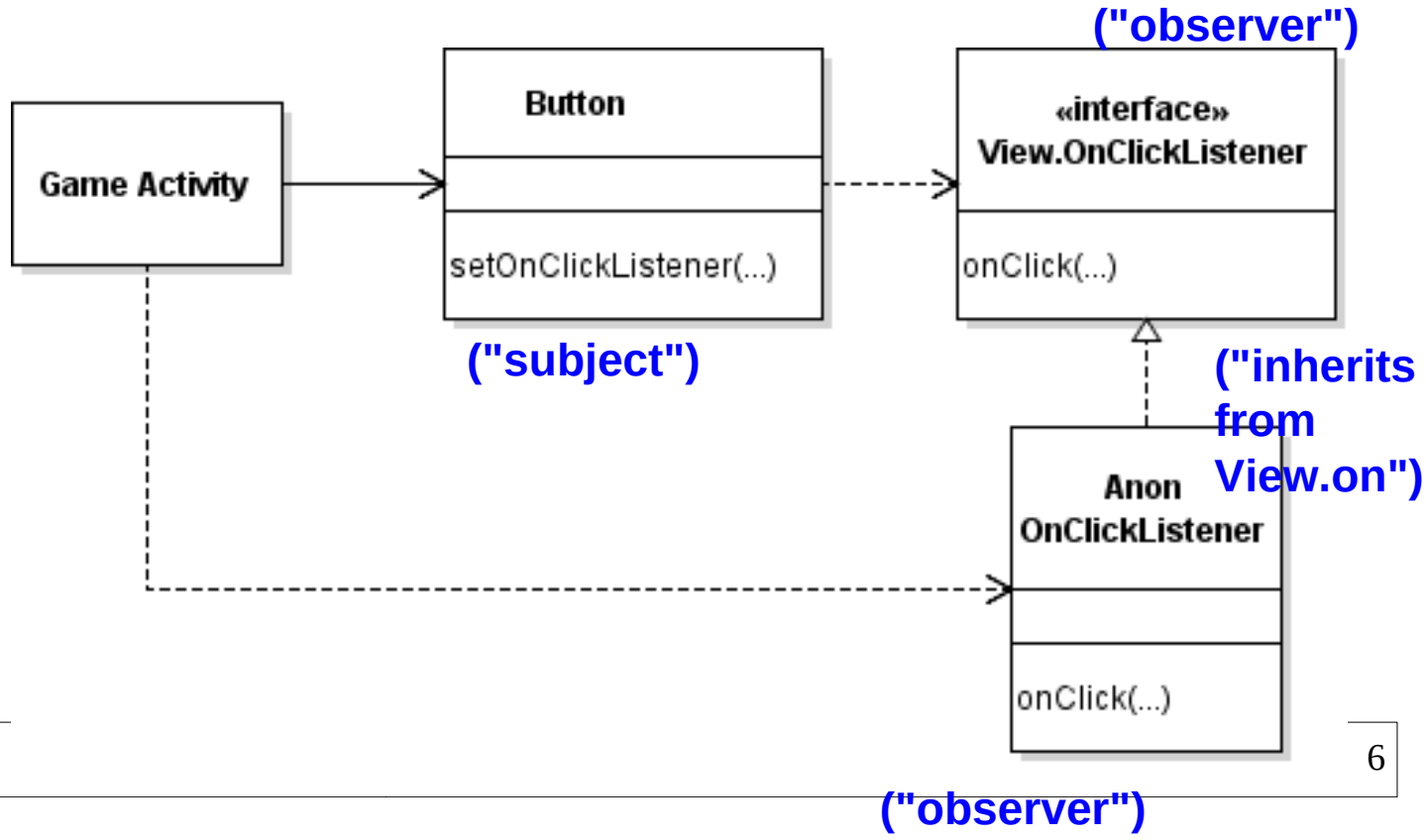
  it simply looks up its observer list

Observer

- Button Example
  - Button knows of a click; Game Activity *wants* to know.
  - Activity creates anonymous OnClickListener
    - Activity registers it with button as a listener.
- Benefit:..

Decoupling

Button knows nothing of program



Observer Pattern

- Context
  - An object, called the subject, is source of events
  - One or more observer objects want to be notified when such an event occurs.
- Solution
  - Define an observer interface type.
    All.. **concrete observers implement it**
  - Subject maintains a collection of observers.
  - Subject supplies methods for attaching and detaching observers.
  - Whenever an event occurs, the subject.. **notifies all observers**