The Observer Pattern

Gerald Britton
IT SPECIALIST

@GeraldBritton www.linkedin.com/in/geraldbritton

Overview

Classification: Behavioral

One to many relationship ...

Between a set of objects

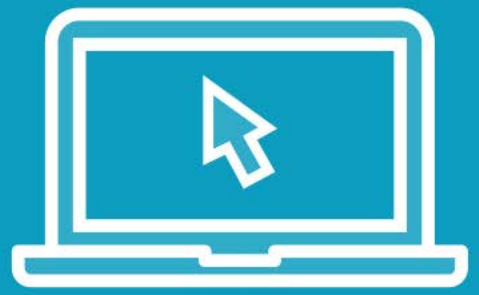
When the state of one changes...

Its dependents are notified

Also know as

- Dependents pattern
- Publish-Subscribe pattern

Demo



Motivating Example:

Dashboard for a tech support center

KPIs:

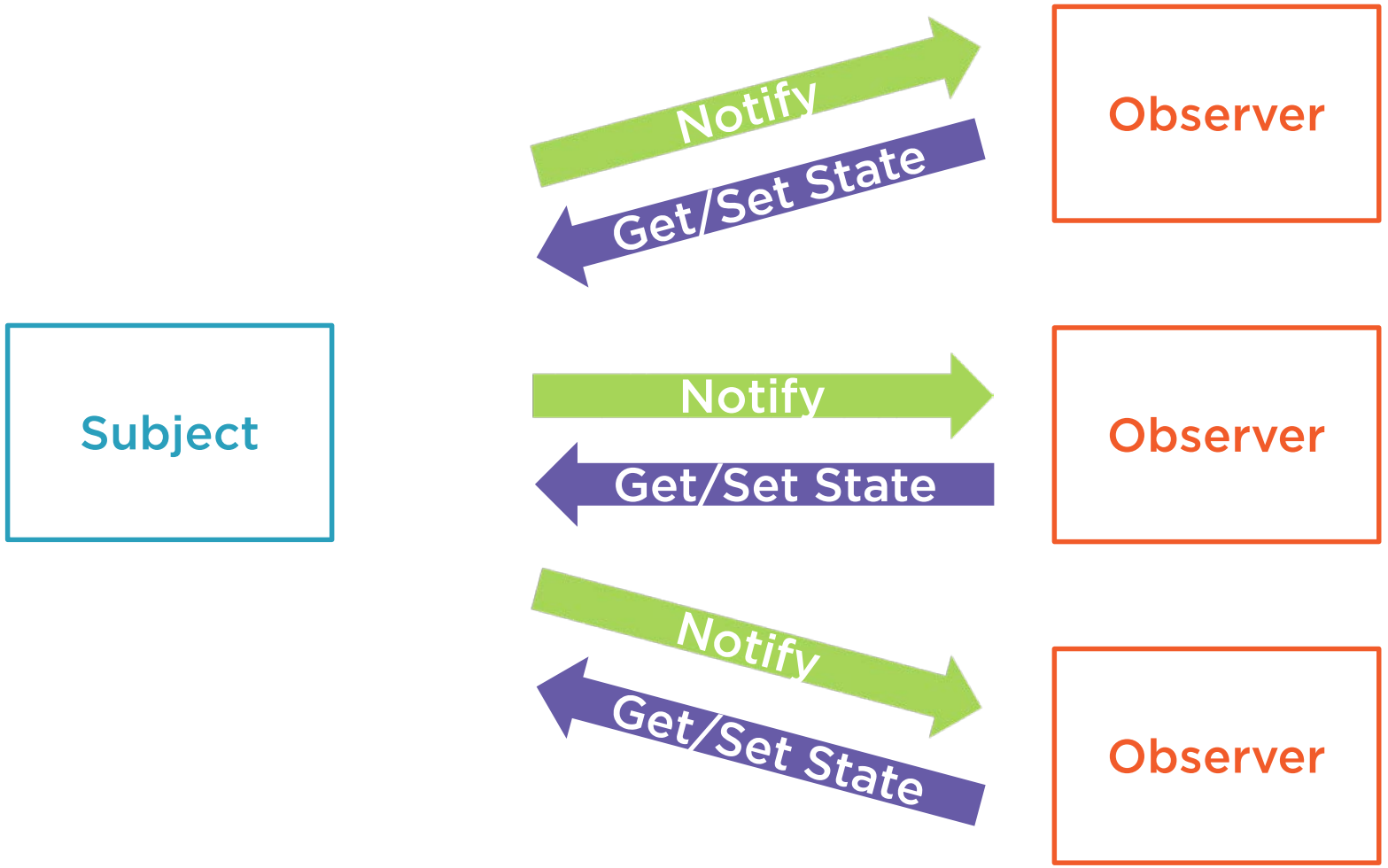
- Open Tickets
- New Tickets in last hour
- Closed tickets in last hour

Dashboard is the observer

KPI source is the subject or publisher

Notes for demo 1

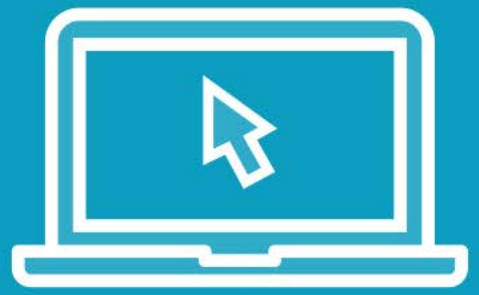
Observer Pattern



Observer Pattern UML

- Separation of concerns
- Single responsibility principle
- Interface segregation principle
- Open/Closed principle
- Dependency inversion principle
- Encapsulate what varies

Demo



- Implement the classic pattern
- Use ABCs for Subject and Observer
- Build concrete classes using the ABCs
- Rebuild the main program
- Use two observers

Notes for demo 2

Notes for demo 2, continued

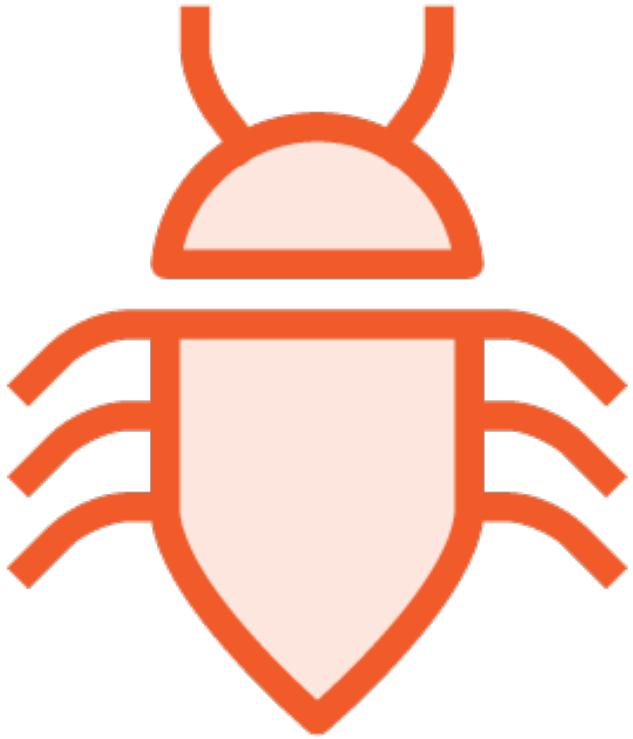
What have we achieved?

Implemented the Observer pattern

Separated the concerns of subject and observer

Easy to add new observers

One subtle bug!



Python runs managed code

Uses reference counters for objects

Set of observers holds references

Need to detach each observer

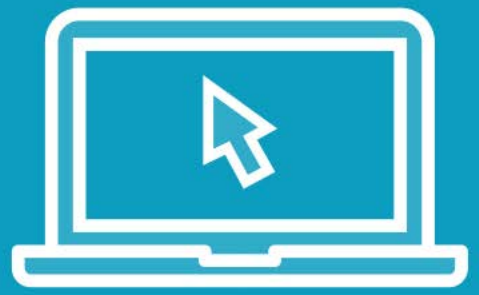
Why?

If not detached, reference count > 0

Stops garbage collection!

Dangling reference

Demo



Use a Python context manager

Change the main program to use “with”

Observers will detach themselves

Subjects will clean up observers

No more dangling references!

Notes for demo 3

Summary

Define a one-to-many relationship

Notify the many when the one changes

Many applications, especially GUIs

MVC pattern:

- Model = Subject, Viewer = Observer

One more thing...

Extra logic in AbsSubject notify method

Enables “push” notifications

See Assignment for details