An example
Tom
GAME
CAT
START
DOG
POW!

GAME START
OBSERVER!
Define a one-to-many dependency between objects so that when one object changes state, all its dependents are NOTIFIED and UPDATED automatically.
Define a one-to-many dependency between objects so that when one object changes state, all its dependents are NOTIFIED and UPDATED automatically.
GAME
CAT START
DOG
HEALTH OBSERVER

GAME CAT START DOG
MEMORY OBSERVER

GAME CAT START DOG
POW!

GAME
CAT
START
DOG
NOTIFY OBSERVERS
MEMORY OBSERVER

GAME START

MEMORY OBSERVER
ONE EVENT TO NOTIFY THEM ALL
ONE EVENT TO NOTIFY THEM ALL
BENEFITS

- Loose coupling between modules
- Subject does not have to know the class type of an observer
- Broadcast communication
- Model/View/Controller frameworks
BENEFITS

- **Loose coupling between modules**
- Subject does not have to know the class type of an observer
- Broadcast communication
- Model/View/Controller frameworks
BENEFITS

- Loose coupling between modules
- **Subject does not have to know the class type of an observer**
- Broadcast communication
- Model/View/Controller frameworks
BENEFITS

- Loose coupling between modules
- Subject does not have to know the class type of an observer
- Broadcast communication
- Model/View/Controller frameworks
BENEFITS

- Loose coupling between modules
- Subject does not have to know the class type of an observer
- Broadcast communication
- **Model/View/Controller frameworks**
KEEP IN MIND

- Method calls are synchronized
- Deleting an observer instance does not automatically detach the observer from the subject
- Garbage collector does not clean up attached observers
KEEP IN MIND

- Method calls are synchronized
- Deleting an observer instance does not automatically detach the observer from the subject
- Garbage collector does not clean up attached observers
KEEP IN MIND

- Method calls are synchronized
- Deleting an observer instance does not automatically detach the observer from the subject
- Garbage collector does not clean up attached observers
- Method calls are synchronized
- Deleting an observer instance does not automatically detach the observer from the subject
- **Garbage collector does not clean up attached observers**
LAPSED LISTENER PROBLEM
common source of memory leaks for OOP languages
ALWAYS DETACH OBSERVERS
OBSERVER IN C#

Language level support:
- delegate
- event and eventhandler
- action (action <T>, action <T1, T2> etc)
- func<TResult> (func<T,TResult>, etc)

Interface level support:
- System.IObservable<T>
- System.IObserver<T>
TODO

- Read Observer chapter from Game Programming patterns and CGLearn
  http://www.gameprogrammingpatterns.com/observer.html

- Do the Observer task from CGLearn
  https://cglearn.codelight.eu/student/tasks

- Also recommend reading dirty flag pattern for optimization, as they are often used together.
  http://gameprogrammingpatterns.com/dirty-flag.html