Advanced Programming

Two ways to create thread

- implements Runnable
- extends Thread

Both require defining run() method that gets called on starting the thread

- Which one's preferred?
- Where’s the difference?

Runnable vs Thread

- **implements Runnable** preferred over **extends Thread**
  - Why?

Runnable vs Thread

- Obvious reason:
  - **extends Thread** : you can’t extend any other class which you required. (multiple inheritance not allowed in Java)

  - **implements Runnable**: you can save a space for your class to extend any other class

Runnable vs Thread

More Significant reason

- When you **extends Thread** class, each of your thread creates unique object and associate with it.

- When you **implements Runnable**, it shares the same object to multiple threads.

- Example

Synchronized

Given a class with synchronized method A, and a normal method C.

If you have two threads in one instance of a program,

1. Can they call A at the same time?
2. Can they call A and C at the same time?

Synchronized

Given a class with synchronized method A, and a normal method C.

If you have two threads in one instance of a program,

1. Can they call A at the same time?

Ans. No. If one thread is executing a synchronized method, all other threads which want to execute any of the synchronized methods on the same objects get blocked.

2. Can they call A and C at the same time?

Ans. Yes

Synchronized

Consider two synchronized methods- m1 and m2 in a class

If Thread t1 access the m1 method (synchronized method), could Thread t2 thread access m2 method (synchronized method) simultaneously?

Synchronized

Consider two synchronized methods- m1 and m2 in a class

If Thread t1 access the m1 method (synchronized method), could Thread t2 thread access m2 method (synchronized method) simultaneously?

Answer: **No! The synchronized keyword applies on object level, and only one thread can hold the lock of the object.**

Then what would be the state of t2 while t1 accesses the method m1?

Synchronized

Then what would be the state of t2 while t1 accesses the method m1?

Answer: I would wait for t1 to release the lock (return from the method or invoke Object.wait()).

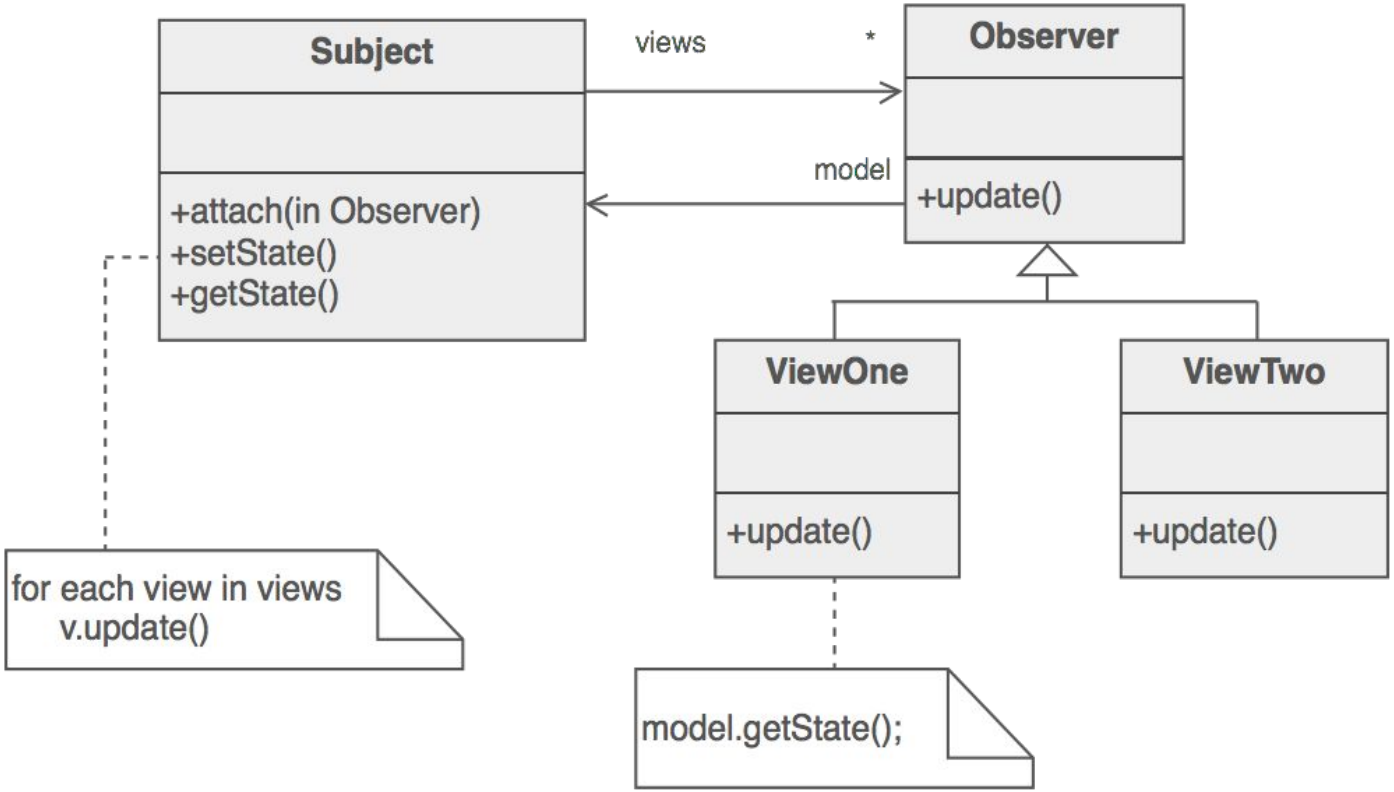
Specifically, it will be in a BLOCKED state.

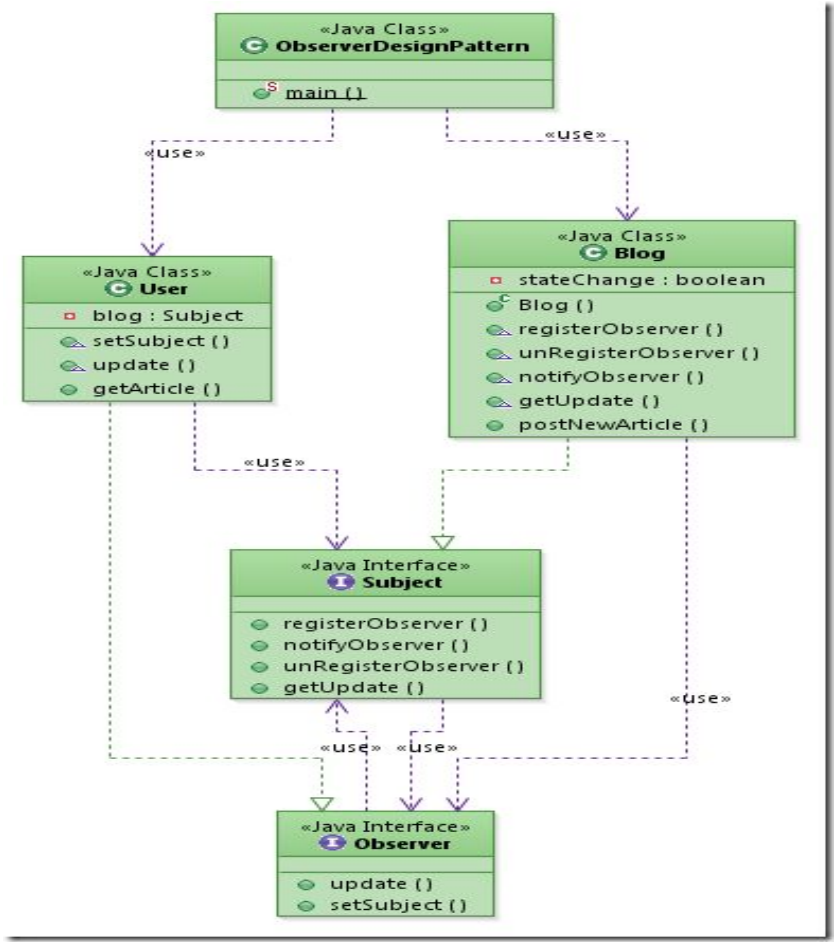
Interleave thread execution

Designing a Producer Consumer Problem

- Using semaphores?

Observer Design Pattern




«Java Class»
ObserverDesignPattern
main ()
«use»
«use»
«Java Class»
User
blog : Subject
setSubject ()
update ()
getArticle ()
«Java Class»
Blog
stateChange : boolean
Blog ()
registerObserver ()
unRegisterObserver ()
notifyObserver ()
getUpdate ()
postNewArticle ()
«use»
«Java Interface»
Subject
registerObserver ()
notifyObserver ()
unRegisterObserver ()
getUpdate ()
«use»
«use»
«use»
«Java Interface»
Observer
update ()
setSubject ()