Diagramme de classe Modèle M3 : dans un univers sans appel asynchrone d'opération, 1 thread par client, n threads scheduler: application de Activer Object pour ObserverDeCapteur <<Interface>> Update <<Interface>> <<Interface>> Capteur Callable<V> Future<V> + call(): V + attach(o:CapteurAsync): void + detach(o:CapteurAsync): void + getObservers(): Set<ObserverDeCapteurAsync> + getValue():Integer MethodInvocation Future + updateLockedValue(): void ConcreteMethodInvocation + releaseLock(): void + tick(): void **ScheduledExecutorService** Client -Д _Scheduler= + schedule(c:Callable<V>): Future<V> :ActiveObject CapteurImpl - observers: Set<ObserverDeCapteurAsync> - algorithmeDiffusion: AlgorithmeDiffusion - value: Integer lockedValue: Integer <<Interface>> ObserverDeCapteur Service ConcreteServant + attach(o:CapteurAsync): void + update(s:CapteurAsync): void + detach(o:CapteurAsync): void + getObservers(): Set<ObserverDeCapteurAsync> - getData(): List<Integer> + getValue():Integer + updateLockedValue(): void + releaseLock(): void + tick(): void **Afficheur** <<Interface>> data: List<Integer> Context **ObserverDeCapteurAsync** - maxValue: int - object: Object - update(): Future<Void> setCapteur(Capteur capteur): void + update(s:CapteurAsync): void :Strategy + getData: List<Integer> Strategy <<Interface>> Canal AlgoDiffusion - scheduledExecutorService: ScheduledExecutorService + configure(c:Capteur): void - afficheur: ObserverDeCapteur + execute(): void - capteur: Capteur + semaphoreReleaseOnce(): void + update(): Future<Void> + setCapteur(Capteur capteur): void ConcreteStrategy **DiffusionAtomique DiffusionSequentielle DiffusionCausal** capteur: Capteur capteur: Capteur capteur: Capteur - semaphore: Semaphore semaphore: Semaphore - semaphore: Semaphore + configure(c:Capteur): void + configure(c:Capteur): void + configure(c:Capteur): void + execute(): void + execute(): void + execute(): void + semaphoreReleaseOnce(): void + semaphoreReleaseOnce(): void + semaphoreReleaseOnce(): void