Diagramme de classe Modèle M3: dans un univers sans appel asynchrone d'opération, 1 thread par client, n threads scheduler: application de Active Object pour Capteur <<Interface>> GetValue <<Interface>> <<Interface>> Capteur Future<Integer> Callable<V> + call(): V + attach(o:ObserverDeCapteurAsync): void + detach(o:ObserverDeCapteurAsync): void getObservers(): Set<ObserverDeCapteurAsync> + getValue():Integer MethodInvocation Future + updateLockedValue(): void ConcreteMethodInvocation + releaseLock(): void + tick(): void **ScheduledExecutorService** Servant \_Scheduler + schedule(c:Callable<V>): Future<V> :ActiveObject CapteurImpl - observers: Set<ObserverDeCapteurAsync> - algorithmeDiffusion: AlgorithmeDiffusion - value: Integer - lockedValue: Integer <<Interface>> ObserverDeCapteur Service + attach(o:ObserverDeCapteurAsync): void ConcreteClient + update(s:CapteurAsync): void + detach(o:ObserverDeCapteurAsync): void + getData(): List<Integer> + getObservers(): Set<ObserverDeCapteurAsync> + getValue():Integer + updateLockedValue(): void + releaseLock(): void + tick(): void **Afficheur** <<Interface>> data: List<Integer> Context CapteurAsync maxValue: int - object: Object getValue(): Future<Integer> + 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 + getValue():Future<Integer> 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 Δ