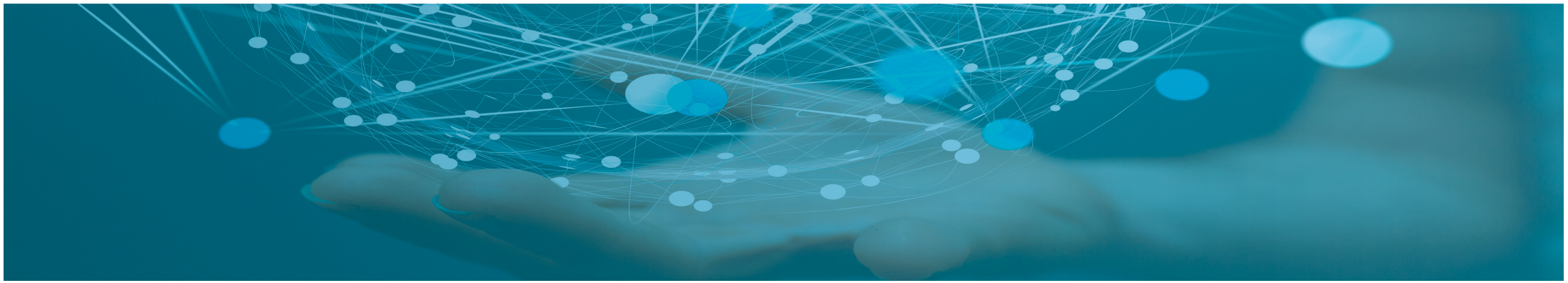
UNIVERSITÉ CÔTE D'AZUR

Correction des TDs

Michel.RIVEILL@univ-cotedazur.fr



Les outils vus en cours

UNIVERSITÉ CÔTE D'AZUR

(	FSP	Java

Verrou	`VERROU = (prendre->rendre->VERROU)`	`synchronized aMethod () { blabla }`
		`synchronized (anObject) { blabla }`
		
		`new Lock`
		`new ReentrantLock`
		`new ReentrantReadWriteLock`

Sémaphore	`SEMAPHORE(N=0) = SEMA[N],`
	`SEMA[v:Int] = (up->SEMA[v+1]`
	`|when(v>0) down->SEMA[v-1]),`
	`SEMA[Max+1] = ERROR.`
		`new Semaphore(valeur, boolean)`

Moniteur	`MONITOR = SPACES[N],`
	`SPACES[vide:T] =`
	`(when(vide>0)arrive->`
	`SPACES[vide-1]`
	`|when(vide<N) depart ->`
	`SPACES[vide+1])`
		`class MONITOR {`
		`vide = 0`
		`synchronized arrive () {`
		`while vide<=0 wait();`
		`vide -= 1;`
		`notifyAll();`
		`}`
		`…`
		`}`

Exemple 1 : section critique (Le jardin)

UNIVERSITÉ CÔTE D'AZUR

- On entre dans un jardin d'ornement par un des deux tourniquets
- La direction souhaite connaître à chaque instant combine de personnes sont dans le jardin.

- La mise en œuvre consiste en deux threads qui se partage un objet compteur ayant une méthode incrémentée.
- Le code FSP d'une thread peut être modélisée par
  - **`TURNSTILE = (arrive->INCREMENT),`**
    `INCREMENT = (value.read[x:T] -> value.write[x+1]->TURNSTILE).`
  - La section critique est en rouge

Exemple 1 : section critique (Le jardin)

UNIVERSITÉ CÔTE D'AZUR

	Dans main	Dans thread fille

Verrou		synchronized (this) {
		  int temp = value_; //read
		  simulate.interrupt();
		  ++temp; //add1
		  value_=temp; //write
		}

Sémaphore	Créer un mutex
	
	m = new Semaphore(1, true)
		m.acquire()
		int temp = value_; //read
		simulate.interrupt();
		++temp; //add1
		value_=temp; //write
		m.release()

Moniteur	Pas adapté	

Exemple 2 : le parking

UNIVERSITÉ CÔTE D'AZUR

- Un parking est constitué d'une entrée et d'une sortie. Un objet contrôleur règle le bon fonctionnement du parking.
- Une thread est associée à l'entrée. Elle appelle la méthode `arrive` de l'objet contrôleur à chaque arrivée. La thread doit être bloquée s'il ne reste plus de place dans le parking et doit être active tant qu'il y a une place libre dans le parking.
- Une autre thread est associée à la sortie. Elle appelle la méthode `départ` de l'objet contrôleur à chaque départ. La thread doit être bloquée s'il n'y a pas de place occupée dans le parking et doit active tant que toutes les places ne sont pas vide.
- L'objet contrôleur, partagé par les deux thread, possède deux méthodes `arrive` et `depart`. Il doit bloquer ou activer les threads selon le comportement attendu

Exemple 2 : Le parking

UNIVERSITÉ CÔTE D'AZUR

	Dans objet partagé entre les différentes threads
Verrou	Pas adapté

Sémaphore	`class obj_partagé {`
	`  obj_partagé () {`
	`  vide = new Semaphore(N, true);`
	`  plein = new Semaphore(0, true);`
	`  }`
	`  arrive () {`
	`  vide.acquire();`
	`  plein.release();`
	`  }`
	`  …`

Moniteur	`class obj_partagé {`
	`  vide = N`
	`  synchronized arrive () {`
	`  while vide<=0 wait();`
	`  vide -= 1;`
	`  notifyAll();`
	`  }`
	`  …`

Exemple 3 : la voie unique

UNIVERSITÉ CÔTE D'AZUR

- Des voitures arrivant du nord ou du sud doivent passer un pont à voie unique.
- Les voitures ayant la même direction peuvent passer le pont au même moment, mais les voitures ayant des directions opposées ne le peuvent pas.
- Chaque voiture sera implémentée par un thread différent, qui, une fois passée sur le pont, recommence toujours dans la même direction.

Exemple 3 : La voie unique

Verrou	Pas adapté
Sémaphore	Possible mais pas facile à mettre en oeuvre
Moniteur	

Exemple 4 : les lecteurs-rédacteurs

UNIVERSITÉ CÔTE D'AZUR

- Supposons qu'une base de données ait des lecteurs et des rédacteurs, et qu'il faille programmer les lecteurs et les rédacteurs de cette base de données.
- Les contraintes sont les suivantes :
  - plusieurs lecteurs doivent pouvoir lire la base de données en même temps ;
  - si un rédacteur est en train de modifier la base de données, aucun autre utilisateur (ni rédacteur, ni même lecteur) ne doit pouvoir y accéder.

Exemple 4 : Les lecteurs / rédacteurs

UNIVERSITÉ CÔTE D'AZUR

Verrou	Oui en utilisant si verrou ReadWrite existe Sinon pas adapté
Sémaphore	Possible mais pas facile à mettre en oeuvre
Moniteur	

Exemple 5 : le diner des philosophes

UNIVERSITÉ CÔTE D'AZUR

- La situation est la suivante :
  - cinq philosophes (initialement mais il peut y en avoir beaucoup plus) se trouvent autour d'une table ;
  - au milieu de la table se trouve un plat de spaghetti ;
  - sur la table, devant chaque philosophe se trouve une assiette et entre chaque assiette, se trouve une fourchette.
- Un philosophe n'a que trois états possibles :
  - penser pendant un temps indéterminé ;
  - être affamé (pendant un temps déterminé et fini sinon il y a famine) ;
  - manger pendant un temps déterminé et fini.
- Des contraintes extérieures s'imposent à cette situation :
  - quand un philosophe a faim, il va se mettre dans l'état « affamé » et attendre que les fourchettes soient libres ;
  - pour manger, un philosophe a besoin de deux fourchettes : celle qui se trouve à gauche de sa propre assiette, et celle qui se trouve à droite (c'est-à-dire les deux fourchettes qui entourent sa propre assiette) ;

- Le problème consiste à trouver un ordonnancement des philosophes tel qu'ils puissent tous manger, chacun à leur tour. Cet ordre est imposé par la solution que l'on considère comme celle de Dijkstra avec sémaphores. En 1984, K. M. Chandy et J. Misra proposèrent une nouvelle solution permettant à un nombre arbitraire *n* d'agents *identifiés par un nom quelconque* d'utiliser un nombre *m* de ressources.

Exemple 6 : le modèle producteur-consommateur

UNIVERSITÉ CÔTE D'AZUR

- Soit un système avec p processus producteur et n processus consommateurs qui communiquent via un tampon ayant b cases.
- Un producteur dépose des messages dans le tampon.
  - Bien évidemment, si le tampon n'a plus de case libre alors le producteur attend.
- Un consommateur prend les messages déposés dans le tampon.
  - Bien évidemment, si aucun message est disponible dans le tampon, le consommateur attend.
- Les consommateurs doivent prendre les messages dans l'ordre de leur dépôt et une fois le message pris le message disparaît du tampon.

Pour s'entrainer : le Drapeau hollandais

UNIVERSITÉ CÔTE D'AZUR

- Une collection de boules de couleur est répartie entre N processus.
- Il y a M ≥ N couleurs différentes de boules.
- L'objectif est pour les processus d'échanger des boules de sorte que quand l'algorithme termine, pour tout i, le processus i a toutes les boules de couleur i.
- Le nombre total de boules est inconnu aux processus