Pràctiques de laboratori

Els cinc filòsofs comensals (1 sessió)

Concurrència i Sistemes Distribuïts

Introducció

L'objectiu d'aquesta pràctica és dissenyar i implementar diferents solucions al problema de l'interbloqueig (*deadlock*). Quan haja conclòs sabrà:

- compilar i executar programes concurrents.
- detectar interbloquejos en un programa concurrent.
- dissenyar solucions que resolguen el problema de l'interbloqueig.
- implementar aquestes solucions.

Com a llenguatge de programació utilitzarà Java.

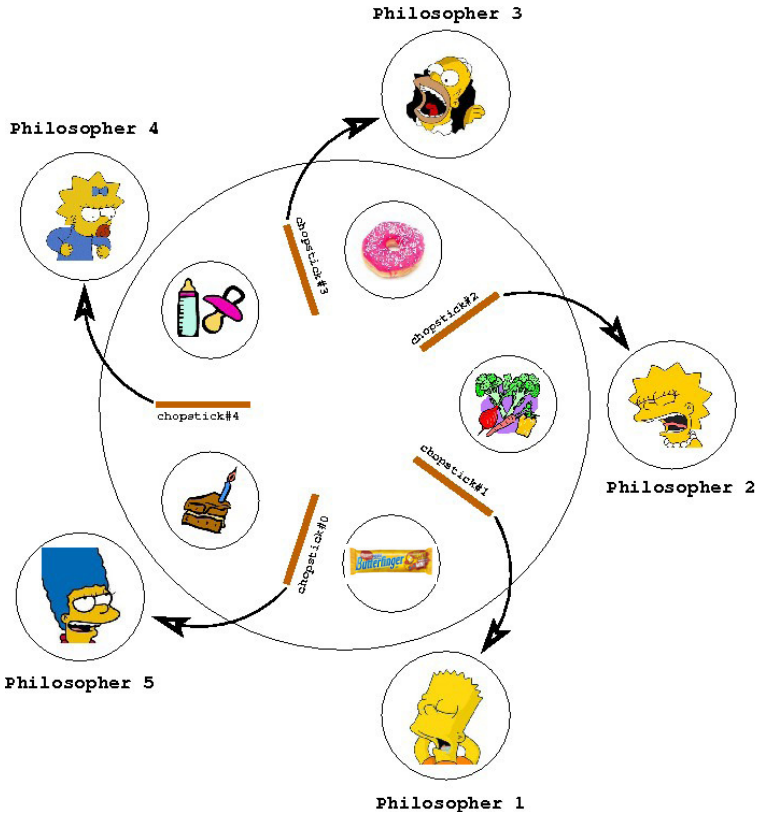
Aquesta pràctica es realitza en grup, on cada grup ha d'estar format per dues persones.

La durada esperada d'aquesta pràctica és d'una setmana. A la sessió de laboratori podrà presentar al professorat de pràctiques els seus progressos i resoldre els dubtes que li sorgisquen. Tinga en compte no obstant això que necessitarà destinar un poc de temps del seu treball personal per a concloure la pràctica. Utilitze per a això els laboratoris docents de l'assignatura i les aules informàtiques del centre en horaris de lliure accés. Pot utilitzar també el seu ordinador personal.

Al llarg de la pràctica veurà que hi ha una sèrie d'exercicis a realitzar. Es recomana resoldre'ls i anotar els seus resultats, per a facilitar l'estudi posterior del contingut de la pràctica.

El problema dels cinc filòsofs comensals

Per a il·lustrar el problema dels interbloquejos utilitzem el conegut exemple dels cinc filòsofs (durant aquest curs apareix descrit com a exemple acadèmic en el document d'exemples corresponent al primer tema, i posteriorment s'ha treballat sobre el mateix en el tema 4).



Diem que pot existir interbloqueig si hi ha alguna possibilitat que es produïsca, independentment que aquesta probabilitat siga gran o xicoteta. Per exemple, si un filòsof pren les dues forquetes en un interval de temps curt és menys probable arribar a un interbloqueig que si passa molt temps entre l'acció d'agafar una forqueta i la següent. Per això, el fet de no observar un interbloqueig no ens assegura que no es puga arribar a produir.

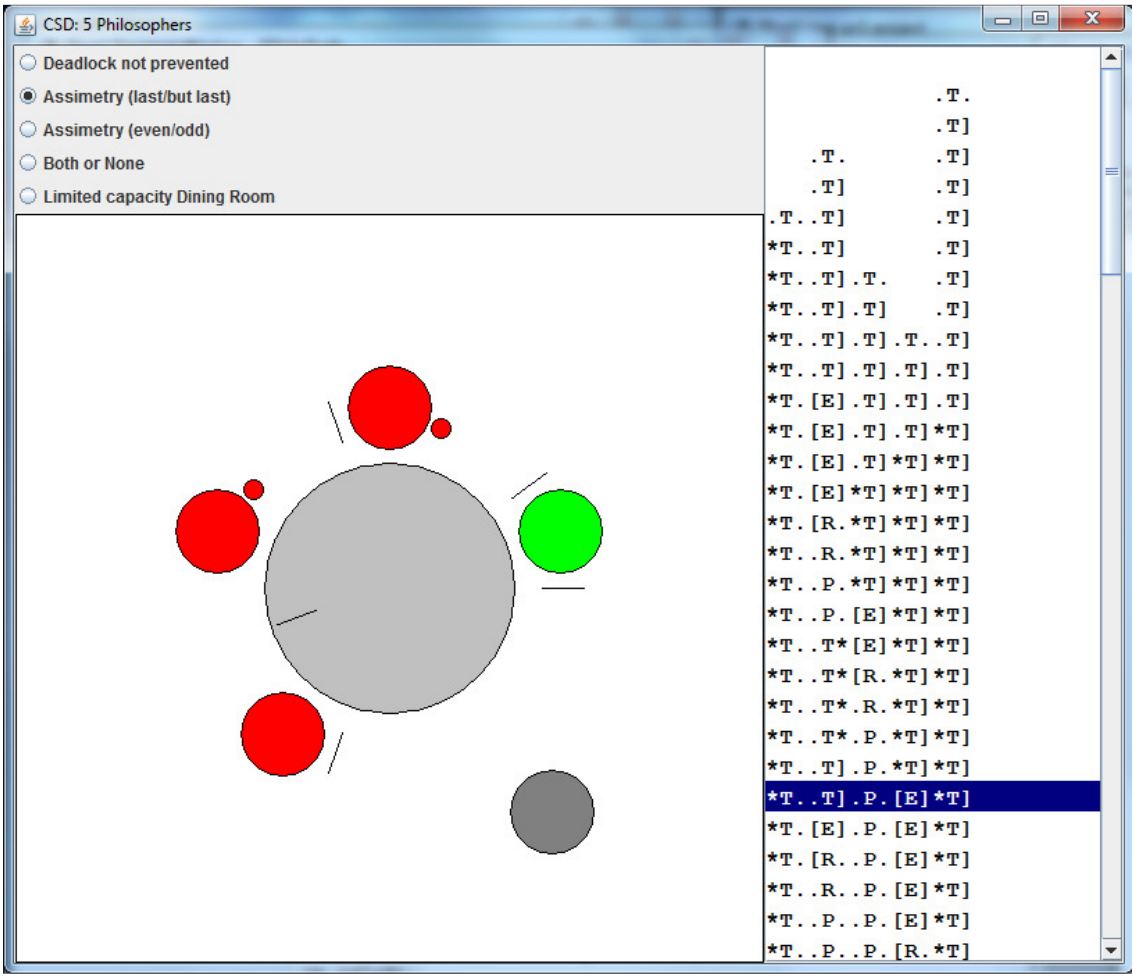
Descàrrega i utilització de la solució parcial al problema

Per a fer aquesta pràctica li proporcionem una solució parcial al problema dels cinc filòsofs. És parcial perque no garanteix que no es produïsca un interbloqueig. Pot descarregar el codi des del lloc PoliformaT de CSD. Està en la carpeta "Recursos / Materiales para el laboratorio / Práctica 2: Los cinco filósofos comensales" i s'anomena "PPhilo.jar".

La interfície i mecanisme de funcionament és similar a l'utilitzat en la pràctica anterior (Piscina Compartida). En la part superior esquerra apareixen les diferents versions seleccionables, de manera que en prémer sobre una opció s'executa la simulació

corresponent. En la part dreta apareix la seqüència d'estats corresponents a aquesta simulació, i la part inferior esquerra mostra la representació gràfica de l'estat seleccionat.

En el codi proporcionat únicament està completa la versió 0 (*Deadlock not prevented*). La resta de versions són inicialment equivalents a la bàsica, i la tasca de l'alumne és completar el codi de les mateixes fins a obtenir la funcionalitat sol·licitada.



La següent taula ajuda a interpretar els estats possibles d'un filòsof i la seua representació textual i gràfica. En el cas del text s'utilitzen 3 caràcters per a indicar l'estat de cada filòsof.

Estat	**Txt**	**Gràfic**
Inactiu (ja ha acabat)		
Espera asseure's a la taula (entrar)	.*.	Cercle roig allunyat de la taula
Assegut a la taula (talking)	.T.	Cercle roig al costat de la taula
Assegut a la taula, amb la forqueta dreta en la mà	.T]	Cercle roig al costat de la taula, línia en la part dreta indicant forqueta
Assegut a la taula, amb la forqueta esquerra en la mà	[T.	Cercle roig al costat de la taula, línia en la part esquerra indicant forqueta
Menjant (eating)	[E]	Cercle verd al costat de la taula, línies en ambdós costats indicant forquetes
Reflexionant (pondering)	.P.	Cercle gris allunyat de la taula
Assegut a la taula, sense forquetes i esperant la forqueta dreta	.T*	Cercle roig al costat de la taula, cercle roig xicotet en la part dreta indicant que espera aqueixa forqueta

Assegut a la taula, sense forquetes i esperant la forqueta esquerra	*T.	Cercle roig al costat de la taula, cercle roig xicotet en la part esquerra indicant que espera aqueixa forqueta
Assegut a la taula, amb la forqueta dreta i esperant la forqueta esquerra	*T]	Cercle roig al costat de la taula, línia en la part dreta i cercle roig xicotet en la part esquerra
Assegut a la taula, amb la forqueta esquerra i esperant la forqueta dreta	[T*	Cercle roig al costat de la taula, línia en la part esquerra i cercle roig xicotet en la part dreta
Assegut a la taula, esperant ambdós forquetes	*T*	Cercle roig al costat de la taula, cercles rojos xicotets en ambdós costats
Assegut a la taula, descansant després de menjar (resting), però encara amb la forqueta esquerra	[R.	Cercle blanc al costat de la taula, línia en la part esquerra. NOTA.- s'ha de deixar sempre primer la forqueta dreta, i després l'esquerra
Assegut a la taula, descansant (després de menjar, i abans de tornar a meditar)	.R.	Cercle blanc al costat de la taula

S'assumeix que el filòsof de la part superior és el 0, i els següents en sentit horari es numeren consecutivament (1, 2, 3, 4). En els missatges de text els filòsofs están ordenats del 4 al 0. L'estat representat en la imatge utilitzada com exemple d'execució indica:

Filòsof	4	3	2	1	0
Estat	*T.	.T]	.P.	[E]	*T]

Que s'interpreta com:

- El filòsof 4 està xarrant (“talking”, esperant menjar): espera la forqueta esquerra (l'està usant el filòsof 0), i no té la forqueta dreta.
- El filòsof 3 està xarrant (“talking”, esperant menjar), i disposa de la forqueta dreta.
- El filòsof 2 està meditant (“pondering”).
- El filòsof 1 està menjant (“eating”).
- El filòsof 0 està xarrant (“talking”, esperant menjar): disposa de la forqueta dreta, i està esperant l'esquerra (perquè l’usa el filòsof 1).

S’ha de tindre en compte que des del programa ha de invocar-se el mètode corresponent de *state* (objecte de la classe *StateManager*) si es desitja realizar una acció per a canviar l’estat del filòsof (veure com s'utilitza en la classe *RegularTable* del codi proporcionat). La següent taula mostra els mètodes de *StateManager* que permeten canviar l’estat. Cal tindre en compte que algunes accions només són vàlides per a alguns tipus de taules.

Mètode de la classe *StateManager*	**Acció del filòsof**
begin	Inici
end	Fi
wenter	Espera per a seure (entrar) a la taula
enter	Seu (entra) a taula
exit	S’alça (eix) de la taula
takeR	Agafa la forqueta dreta
takeL	Agafa la forqueta esquerra
takeLR	Agafa les dues forquetes
eat	Comença a menjar
ponder	Comença a pensar
wtakeR	Espera agafar la forqueta dreta
wtakeL	Espera agafar la forqueta esquerra
wtakeLR	Espera agafar les dues forquetes
dropR	Solta la forqueta dreta
dropL	Solta la forqueta esquerra

Activitat 0

Aquesta activitat té com a objectiu constatar que es pot produir un interbloqueig en executar la solució subministrada i també que aquests tipus d'error són difícils de detectar perquè en situacions reals la probabilitat d'interbloqueig és molt baixa.

En primer lloc, dedique uns minuts a explorar la solució. Per a això, analitze el contingut de la classe Philo i la clase RegularTable (que implementa la interfície Table):

arxiu	descripció
Philo	Els objectes d'aquesta classe són fils que realitzen les accions dels filòsofs.
RegularTable	Aquesta classe implementa un monitor que controla l'ús que fan els filòsofs de les forquetes. Resol correctament l'ús de les forquetes en exclusió mútua i la suspensió i reactivació dels filòsofs en funció de l'estat de cada forqueta. No garanteix en canvi l'absència d'interbloquejos.

A continuació, pare esment a aquest fragment de codi de l'arxiu Philo.java, mètode run().

```
table.takeR(id); delay(msegDelay); table.takeL(id);
```

Per a augmentar la probabilitat que es produïsca l'interbloqueig, s'ha afegit un retard des que el filòsof agafa la forqueta dreta fins que agafa la forqueta esquerra. El valor d'aquest retard es pot indicar com a argument en llançar l'execució del programa. S'admeten valors entre 1 i 10 (mil·lisegons), i el valor per defecte (si no s'indica argument o s'indica un valor no vàlid) és 10.

Exercici 0.1: Execute 10 vegades el programa base (opció Deadlock not prevented) utilitzant com argument del programa cadascun dels valors de N de la taula següent, i anote quants interbloquejos es produeixen en cada cas.

N	Nombre d'interbloquejos
1	
5	
10	

NOTA.- Després de l'execució, en consola apareix la frase DEADLOCK o bé OK. També es pot observar analitzant l'estat final (si tot ha anat bé els filòsofs han desaparegut i totes les forquetes estan sobre la taula, i en cas d'interbloqueig tots els filòsofs estan esperant al voltant de la taula).

Exercici 0.2: ¿Com influeix el valor de N en la probabilitat d'interbloqueig?

Activitats

Proposem quatre mecanismes per a prevenir interbloquejos:

Versió	Descripció
1	Tots els filòsofs agafen primer la seua forqueta dreta, excepte el número 4 (l'últim), que agafa primer la seua forqueta esquerra
2	Els filòsofs parells agafen primer la seua forqueta dreta mentre que els filòsofs imparells agafen primer la seua forqueta esquerra
3	Els filòsofs, o bé agafen les dues forquetes o, si algun està ocupat, no n'agafen cap
4	Com a màxim poden haver-hi quatre filòsofs asseguts a la taula

L'objectiu de la pràctica és desenvolupar i provar el funcionament de les diferents versions. És important destacar que han de realitzar-se les crides corresponents als métodes públics de la classe *StateManager*.

La implementació de la classe *StateManager* s'encarrega de verificar que no s'intenten operacions il·legals (ex.- alliberar una forqueta que no es té, agafar una forqueta utilitzada per un altre filòsof, etc.). Si s'intenta una acció il·legal, el programa indica l'error i avorta.

Versions 1 i 2: Asimetria

Les versions 1 i 2 de la taula anterior es basen en asimetria, per al que s'ha previst una classe ***LefthandedPhilo*** que correspon a un filòsof que ha d'agafar les forquetes en ordre contrari al de Philo, però manté l'ordre en què s'alliberen.

Exercici 1.1: Complete la classe ***LefthandedPhilo*** y realitze les modificacions oportunes a la classe PPhilo, per a donar solució tan a la versió 1 com a la versió 2. Tinga en compte que per a que el codi funcione de forma correcta, la classe *LefthandedPhilo* deu extendre a Philo.

NOTA.- Recorde que, amb independencia d'ordre amb que s'agafen les forquetes, sempre s'ha de deixar primer la forqueta dreta y després l'esquerra, ja que açó es una restricció imposada per la classe StateManager (es a dir, per l'interfície gràfica).

Exercici 1.2: Comprove que no es poden produïr ja interbloquejos, executant varies vegades el programa, tant en la versió *Assimetry (last/but last)* com en la versió *Assimetry (even/odd)*.

Preguntes:

1) Quina condició o condicions de Coffman es trenquen en la solució proposada per a les versions 1 i 2?

2) Tenim garantia de que no es produiran mai interbloquejos?

Versió 3: Tot o res

En la versió 3 dels mecanismes proposats a la taula anterior, el mecanisme per a previndre l'interbloqueig consisteix en el fet que "els filòsofs, o bé agafen les dues forquetes o, si algú està ocupat, no agafen cap". En este cas, els filòsofs deuràn sol-licitar les forquetes amb l'operació ***takeLR***. Esta operació (a implementar per l'alumne) permetrà al filòsof agafar les dues forquetes al mateix temps, o bé cap (si alguna d'elles està ocupada).

Exercici 2.1: Realitze les modificacions oportunes per a donar solució a la versió 3.

Preguntes:
1) Quina condició o condicions de Coffman es trenquen amb la solució proposada per a la versió 3?

2) S'ha utilitzat la mateixa taula *RegularTable* que en les versions anteriors o bé s'ha utilitzat una taula diferent? Per què?

3) S'ha utilitzat el filòsof *Philo* de la versió 1? S'ha utilitzat el filòsof *LefthandedPhilo*? Ha sigut necessari implementar un nou tipus de filòsof? Per què?

Versió 4: taula amb capacitat limitada

Finalment, la versió 4 proposa com a mecanisme de prevenció d'interbloquejos que com a màxim puga haver-hi quatre filòsofs assentats a la taula. Per a això, es proposa fer ús dels mètodes **enter**/**exit** de la taula, de manera que els filòsofs hauran de sol·licitar entrar i eixir (enter/exit) de la taula en cada iteració.

Excercici 3.1: Realitze les modificacions oportunes per a donar solució a la versió 4.

Preguntes:
1) Quina condició o condicions de Coffman es trenquen amb la solució proposada per a la versió 4?

2) S'ha utilitzat la mateixa taula *RegularTable* que en les versions anteriors o bé s'ha utilitzat una taula diferent? Per què?

3) S'ha utilitzat el filòsof *Philo* de la versió 1? Ha sigut necessari implementar un nou tipus de filòsof? Per què?