Pràctiques de laboratori
Ús compartit d'una piscina
(2 sessions)

Concurrència i Sistemes Distribuïts

Introducció

Aquesta pràctica té com a objectiu analitzar i completar un programa concurrent en el qual s'apliquen diferents condicions de sincronització entre fils. Quan haja conclòs sabrà:

- Compilar i executar programes concurrents.
- Detectar errors de sincronització en un programa concurrent.
- Analitzar els requisits de sincronització que apareixen en els programes concurrents.
- Dissenyar solucions que complisquen amb els requisits analitzats.
- Implantar 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 d'aquesta pràctica és de dues setmanes. Cada setmana ha d'assistir a una sessió de laboratori on podrà comentar amb el professorat de pràctiques els seus progressos i resoldre els dubtes que li puguen sorgir. Tinga en compte no obstant açò que necessitarà destinar part de temps del seu treball personal per a concloure la pràctica. Utilitze per a açò 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 de l'ús compartit d'una piscina

En esta pràctica es requereix modelar el funcionament d'una piscina compartida per xiquets i instructors, en la qual els xiquets aprenen a nadar sota la supervisió dels instructors.

- Existeixen per tant dos tipus de nadadors:
  - ***Kid*** (xiquet), que està aprenent a nadar
  - ***Instructor***, que supervisa l'entrenament d'un o més xiquets
- En les instal·lacions de la piscina distingim dues zones:
  - Got de la piscina (on es nada). - tot nadador (ja siga xiquet o instructor) entra en l'aigua i roman un temps nadant fins que ix a descansar.
  - Zona externa (terrassa).- On descansen els nadadors durant un temps entre bany i bany.

Tot nadador (entrenador-instructor o xiquet) executa el següent pseudo-codi:

Nadador
repetix un nombre de cicles nada descansa

Cada nadador es representa mitjançant un fil, i executem de forma concurrent diversos fils de tipus ***Kid*** i diversos fils de tipus ***Instructor*** (ambdues quantitats són configurables). Cada vegada que es nada o descansa, la durada d'aquesta acció és aleatòria.

L'objecte compartit pels diferents fils és la piscina (**Pool**). Per a utilitzar la piscina, els fils han d'invocar les operacions corresponents, que es comentaran més endavant.

Classe abstracta Piscina *(Pool)*

Tant xiquets com instructors han de respectar les normes d'ús de la piscina. Quan un fil intenta una acció que incompleix les regles de la piscina, aquest fil ha d'esperar fins que l'estat de la piscina convertisca aquesta acció en legal (sincronització condicional).

Distingim 5 casos possibles (que modelem com a diferents tipus de piscines), cada un amb normes més estrictes que l'anterior *(per a Pool3 cal complir també les de Pool2 i Pool1, ..)*:

Tipus de piscina	***Regles per a K xiquets i I instructors***
Pool0	Bany lliure (no hi ha regles) *(free access)*
Pool1	Els xiquets no poden nadar sols (ha d'haver-hi algun entrenador-instructor amb ells) *(kids cannot be alone)*
Pool2	Poden nadar un màxim de K/I xiquets per entrenador-instructor *(max kids/instructor)*
Pool3	No es pot superar l'aforament màxim permès de nadadors (establit com (K+I)/2 nadadors) *(max capacity)*
Pool4	Si hi ha instructors esperant eixir, no poden entrar xiquets a nadar *(kids cannot enter if there are instructors waiting to exit)*

Per exemple, en Piscines tipus 1 (*Pool1*) o superior, si no hi ha cap instructor nadant i un xiquet desitja nadar, no se li permet entrar en l'aigua fins que entre a nadar un instructor.

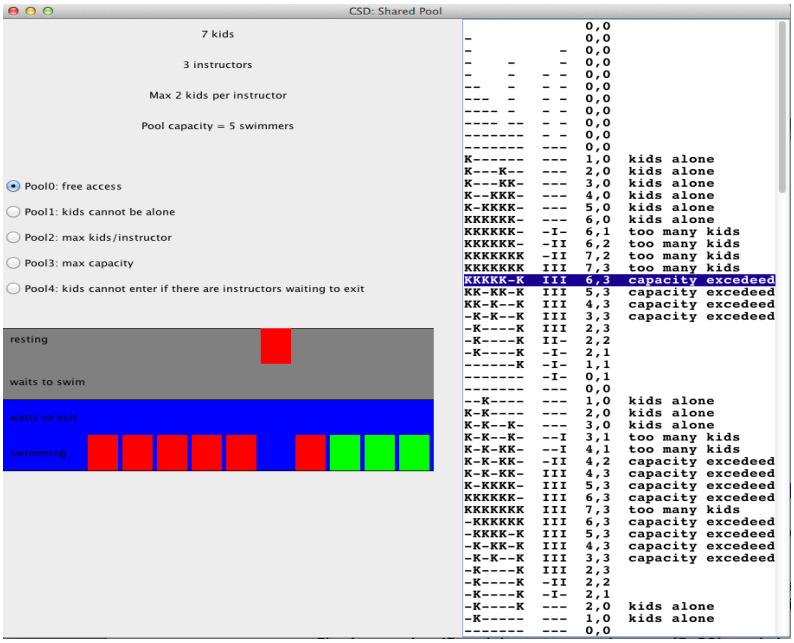
Per a la resolució d'aquesta pràctica partim de l'aplicació que suporta piscines de tipus 0. Pot descarregar aquest codi des del lloc Poliformat de l'assignatura (carpeta "Recursos / Materiales para el laboratorio / Practica 1: Uso compartido de una piscina", fitxer "PPool.jar"). L'objectiu d'aquesta pràctica és introduir progressivament les modificacions necessàries en el codi perquè també suporte els restants tipus de piscines.

Codi proporcionat

La aplicació proporcionada està completa (es pot compilar i executar directament), però posseeix funcionalitat limitada (tracta tots els tipus de piscina com a piscines tipus 0, o siga sense regles). En conseqüència, no haurà de modificar el codi per a realitzar les primeres proves, però sí en activitats posteriors.

Per a treballar amb el codi proporcionat, pot realitzar-ho des de l'entorn BlueJ o des de qualsevol terminal que tinga instal·lat un compilador Java. Per exemple, pot obrir en BlueJ el fitxer *PPool.jar* com un nou projecte (és a dir, seleccione Project/Open Project i òbriga PPool.*jar), amb el que tindrà accés a tot el conjunt de classes que es proporcionen en la pràctica. La classe *PPool* conté el mètode principal *main*, que podrà llançar la execució sense arguments. D'altra banda, des d'un terminal, pot executar directament l'aplicació sense arguments (amb la instrucció *java –jar PPool.jar*).

En tots dos casos, l'aplicació mostra una pantalla (veure següent figura) en la qual es pot seleccionar el tipus de piscina. En triar un tipus de piscina es llança la seua simulació que finalitza quan el semàfor es mostra en verd. Si al llançar la simulació d'una piscina observem que el semàfor es queda en color roig indefinidament i que en la zona dreta de la pantalla s'ha detingut l'escriptura, voldrà dir que s'ha produït una situació d'interbloqueig (els fils s'han quedat bloquejats entre sí).



La pantalla mostrada de l'aplicació s'estructura en les àrees que es detallen a continuació.

En la part superior esquerra apareixen els paràmetres de la simulació: 7 xiquets (*kids*), 3

instructors, màxim 2 xiquets per instructor, aforament (*pool capacity*) 5 nadadors.

- Per defecte s'utilitzen 7 xiquets i 3 instructors, però en llançar l'execució es pot indicar com a paràmetre el nombre desitjat (ex. *java -jar PPool.jar **12 4*** si desitgem 12 xiquets i 4 instructors).
- El nombre de xiquets ha d'estar en l'interval [5..20], i el d'instructors en [2..5]. En cas contrari, el programa assumeix el valor per defecte.
- El màxim de xiquets per instructor correspon a l'operació K/I (o siga, *numXiquets/numInstructors*).
- L'aforament màxim correspon a l'operació (K/I)/2 (o siga, (*numXiquets + NumInstructors*)/2).

En la part central esquerra apareixen els diferents tipus de piscina possibles. Inicialment no hi ha cap entrada seleccionada. En prémer sobre alguna de les entrades, s'executa una nova simulació amb aquest tipus de piscina tal com s'ha indicat anteriorment.

NOTA.- en el codi proporcionat les piscines tipus 1..4 es comporten igual que la 0, per la qual cosa únicament observarem les diferències en anar completant el codi.

En la part dreta apareix la seqüència d'estats pels quals passa la simulació, expressats en forma de cadena de caràcters (una línia per estat). Es tracta d'una llista sobre la qual ens podem desplaçar i seleccionar qualsevol estat. En una línia observem diferents columnes:

- *NumXiquets* caràcters (un per xiquet). Els valors possibles són:
  - No ha començat o ja ha acabat.- apareix un espai en blanc
  - Descansa.- apareix el caràcter '-'
  - Espera nadar.- apareix el caràcter '*'
  - Espera eixir.- apareix la lletra 'k'
  - Nada.- apareix la lletra 'K'
- *NumInstructors* caràcters (un per instructor). Els valors possibles són:
  - No ha començat o ja ha acabat.- apareix un espai en blanc
  - Descansa.- apareix el caràcter '-'
  - Espera nadar.- apareix el caràcter '*'
  - Espera eixir.- apareix la lletra 'i'
  - Nada.- apareix la lletra 'I'
- El nombre de xiquets i d'instructors actualment en l'aigua (inclou aquells que naden i aquells que esperen eixir de l'aigua).
- Si s'incompleix alguna regla, apareix el corresponent missatge d'avís. Si aquesta regla **hauria de** complir-se en el tipus de piscina actual, l'avís es converteix en un missatge d'error (en **color roig**). En eixe cas el semàfor mostra la llum groga.
- Si tots els xiquets acaben, s'indica el missatge "finished". Si no queden instructors (han finalitzat tots ells) però hi ha xiquets que seguixen actius, s'indica "out of instructors"."

En la part inferior esquerra observem la representació gràfica de l'estat que hàgem seleccionat en la part dreta (és a dir, la representació de l'estat 'actual').

- Cada nadador es representa mitjançant un rectangle de color (roig per als xiquets, verd per als instructors). Cada nadador sempre apareix en la mateixa columna, i únicament modifica la seua posició vertical per a indicar l'estat en el qual es troba. Quan un nadador no ha iniciat la seua execució o ja l'ha finalitzat, no es visualitza el seu rectangle
- La zona amb fons gris representa la terrassa, on apareixen els nadadors que descansen (pegats a la part superior) o intenten entrar a nadar però han d'esperar (pegats a la part inferior).

- La zona amb fons blau representa l'aigua (got de la piscina), on apareixen els nadadors que estan nadant (pegats a la part inferior) o estan dins de l'aigua però esperant eixir (pegats a la part superior).

NOTA.- en la piscina tipus 0 els nadadors no esperen mai, per la qual cosa apareixen sempre descansant o nadant.

ANÀLISI DEL CODI

El codi proporcionat implementa diferents classes, que podem classificar en:

- Classes opaques.- Necessàries per al correcte funcionament de l'aplicació, però sense interès per a l'alumne. **No han de modificar-se**.
  - *State, Box, StateRenderer, Light*

- Classes semi-transparents.- Necessitem conèixer la seua interfície (per a invocar operacions des de les classes a desenvolupar per l'alumne), però no hem de conèixer la seua implementació. **No han de modificar-se**.
  - *Pool*.- Classe abstracta, amb operacions abstractes que **han de ser implementades en les piscines Pool0 ... Pool4**. Estes operacions inclouen el mètode init, que permet la inicialització dels paràmetres de la piscina, i les operacions que invoca un nadador quan desitja un canvi d'estat (veure el codi de Swimmer, Instructor i Kid).

```
public abstract void init(int ki, int cap);
public abstract void kidSwims() throws InterruptedException;
public abstract void kidRests() throws InterruptedException;
public abstract void instructorSwims()  throws InterruptedException;
public abstract void instructorRests()  throws InterruptedException;
```

  - *Log.- Classe amb operacions que cal invocar quan hi ha un canvi d'estat d'un nadador (p. e. Quan es queda esperant per a nadar, quan entra en la piscina, etc.) D'esta forma l'aplicació podrà mostrar l'estat correcte dels nadadors.*
    En esta pràctica caldrà fer invocacions als següents mètodes d'esta classe:

```
public void waitingToSwim()
public void swimming()
public void waitingToRest()
public void resting()
```

- Classes transparents.- Interessa aprofundir en la seua implementació (almenys en algun aspecte de la mateixa). Però **no han de modificar-se**.
  - *Swimmer*.- classe abstracta de la qual deriven *Instructor* i *Kid*. El seu mètode *run*() correspon al pseudo-codi indicat al principi d'aquest text

```
public abstract class Swimmer extends Thread{
  final int DELAY=60;
  Random rd=new Random();
  Pool pool; //piscina utilizada
  protected void delay() throws InterruptedException {
          Thread.sleep(DELAY+rd.nextInt(DELAY));} }
  public Swimmer(int id0, Pool p) { super(""+id0); pool=p;}
  public void run() {       //código que ejecuta
    try{
        pool.begin(); delay();
        for (int i=0; i<6 && !this.isInterrupted(); i++) {
          swims();   delay();
          rests();    delay();
        }
        pool.end();
     }catch (InterruptedException ex) {}
```

```
  }
   //a implementar en Instructor y en Kid
  abstract void swims() throws InterruptedException;
  abstract void rests() throws InterruptedException;
}
```

- *Instructor*

```
public Instructor(int id, Pool p) {super(id,p);}
void swims() throws InterruptedException{pool.instructorSwims(); }
void rests() throws InterruptedException {pool.instructorRests(); }
```

- *Kid*

```
public Kid(int id, Pool p) {super(id,p);}
void swims() throws InterruptedException { pool.kidSwims(); }
void rests() throws InterruptedException {pool.kidRests(); }
```

- *PPool.-* És la **classe principal**. D'esta classe interessa conèixer cert codi que es realitza en el mètode *simulate*, en concret:
  - La inicialització de la piscina, cridant al mètode *init* de la classe *Pool*, on s'indica el màxim nombre de xiquets per instructor (KI) i l'aforament màxim de la piscina (CAP).

```
p.init(KI,CAP);
```

  - La creació i arrencada dels fils:

```
// K=number of kids, I=number of instructors
Swimmer[] sw= new Swimmer[K+I]; //declara y crea nadadores
....
for (int i=0; i<K+I; i++)
    sw[i]= i<K? new Kid(i,p): new Instructor(i,p);
....
for (int i=0; i<K+I; i++)
    sw[i].start(); //arranca los nadadores
```

- *Pool0.-* Implementa una piscina d'accés lliure (**tipus 0**). Tot tipus de piscina ha d'estendre la classe abstracta Pool. Observe com s'utilitzen en esta classe els mètodes de la classe Log (objecte log).

```
public class Pool0 extends Pool {
  public void init(int ki, int cap)      {}
  public void kidSwims() { log.swimming(); }
   public void kidRests() { log.resting();}
   public void instructorSwims() {log.swimming();}
   public void instructorRests() {log.resting();}
}
```

- **Classes a modificar**.- Classes el codi de les quals hem de completar.
  - *Pool1, Pool2, Pool3, Pool4.* Actualment implementen el mateix codi que Pool0. Les activitats 1..4 indiquen què ha d'implementar cadascuna d'eixes classes.

Activitat 0 (Pool0)

Llance diverses vegades l'execució de *PPool* amb piscina tipus 0 (bany lliure, o siga sense regles). Pot adonar-se'n que l'evolució de cada execució és diferent a l'anterior (diferent durada dels temps per a nadar/descansar, possiblement decisions diferents a nivell de planificació, etc.). En alguns casos l'ús de la piscina incompliria les regles de piscines més

estrictes (tipus 1, 2, 3).

La següent taula mostra els diferents mètodes de *Pool0*, i per a cadascun quan cal esperar (és a dir, que no es pot completar el mètode perquè l'estat de la piscina no ho permet), com modifica l'estat de la piscina, i a qui cal avisar una vegada modificat aquest estat:

Pool0	cal esperar si …	Modifica estat …	Avisa a …
kidSwims()	*no espera mai*	*No hi ha estat*	*Ningú*
kidRests ()	*no espera mai*	*No hi ha estat*	*Ningú*
instructorSwims()	*no espera mai*	*No hi ha estat*	*Ningú*
instructorRests()	*no espera mai*	*No hi ha estat*	*Ningú*

Observe que en la piscina tipus 0 no hi ha un estat en la piscina que puga canviar i impedir o no que es complete determinada operació (sempre és possible completar l'operació, i per tant mai cal esperar). En conseqüència, no s'utilitza sincronització condicional. Com cap fil esperarà mai, tampoc és necessari reactivar a cap fil en espera en cap cas.

Preguntes Piscina Tipus 0 (bany lliure):

1) Comprove si s'ha utilitzat la paraula *synchronized* al implementar les operaciones de la piscina de tipus 0 (*kidSwims*, *kidRests*, *instructorSwims*, *instructorRests*). S'observaria alguna diferència d'execució entre utilitzar la paraula synchonized o no utilitzar-la?

No s'ha utilizat, no es notaria la diferencia perque cadascu mira les seues variables, no hiha variables compratides

2) Es poden produir condicions de carrera? Per què?

No perque no sutilitzen les variables dels atres fils

3) Com s'ha representat l'estat intern de la piscina? Per què?

En 2 estats, descansant i nadant.
Per a poder complir les condicions que ens demanen.

4) En la piscina de tipus 0, quines transicions es poden produir? Poden els xiquets i instructors passar de "resting" a "swimming" directament, sense passar per estats intermedis? Per què?

Nadar o Descansar ya que no hiha ninguna condicio pertant mai es tindran que esperar.
Si que poden.
Perque no hiha cap restrinccio.

Activitat 1 (Pool1)

Exercici 1.1: Indique com representar l'estat de la piscina de tipus 1 per a poder complir les normes d'ús associades a aquesta piscina. Per a açò, complete la taula corresponent a *Pool1*:

Pool1	cal esperar si …	modifica estat …	Avisa a ….
kidSwims()	*No hi ha cap instructor nadant*	Si	No
kidRests()	No	Si	Instructors
instructorSwims()	No	Si	A xiquets
instructorRests()	Si hiha algun xiquet nadant i es l'ultim instructor nadant	Si	No

Per a fer esperar als fils que sol·liciten una operació contrària a les regles, utilitzem el següent esquema:

```
while (condició d'espera) {
   wait();
}
Actualitza estat
Avisa del nou estat a altres fils que puguen estar esperant aquest canvi
```

Com la instrucció wait() pot retornar amb una interrupció (si el fil és interromput durant la seua espera), en la declaració dels mètodes kidSwims(), kidRests(), instructorSwims(), instructorRests() s'ha d'afegir l'expressió "throws InterruptedException" quan així fóra necessari.

Per exemple, el mètode *kidSwims* de la classe *Pool1* queda de la següent manera (l'alumne ha de completar el que falta):

```
public synchronized void kidSwims() throws InterruptedException {
     while (condició d'espera) {  //COMPLETAR la condició
           log.waitingToSwim();//per a visualitzar la posició del nadador
           wait();
      }
      ...   //Actualitza estat   (COMPLETAR)
      ...   //Si necessari, avisa del nou estat a altres fils
           // amb notifyAll();
      log.swimming();  //per a visualitzar la posició del nadador
}
```

Exercici 1.2: Modifique el mètode *kidSwims* de la classe *Pool1* perquè el fil invocant espere si no hi ha cap instructor nadant (tal com s'indica en la taula, i seguint l'esquema anteriorment descrit). Utilitze les variables internes que considere oportunes per a **comptabilitzar els xiquets i instructors** que entren/ixen de la piscina.

Compile i execute. Observarà que els xiquets que desitgen nadar quan no hi ha cap instructor en l'aigua es comporten correctament (esperen en lloc d'entrar), però segueix havent-hi situacions il·legals.

Exercici 1.3: Analitze la traça de l'execució i determine quins problemes o situacions il·legals apareixen i quines situacions sí que s'han resolt.

Exercici 1.4: Revise la resta de mètodes de la classe *Pool1* (és a dir, *kidRests*, *instructorSwims*, *instructorRests*) i modifique aquells que siga necessari, per a reflectir les entrades/eixides a la piscina dels xiquets i instructors de forma apropiada. Anote a continuació els canvis que ha realitzat en el codi.

Exercici 1.5: Comprove que les regles de la piscina tipus 1 es compleixen ara. Repetiu l'execució modificant el nombre de xiquets i/o instructors. Comprove que l'execució continua sent correcta.

Important! Comprove que el programa no es queda mai bloquejat quan s'executa la piscina 1. Si així fora, revise les condicions d'espera i d'activació dels fils.

Recorde que el **color del semàfor** indica:

- Semàfor verd: execució correcta. Es compleixen totes les normes de la piscina.
- Semàfor grog: l'execució ha terminat, però alguna de les normes de la piscina no es compleix.
- Semàfor roig: algun dels fils (xiquets o instructors) s'ha quedat bloquejat indefinidament, per la qual cosa l'execució no podrà acabar.

Preguntes Piscina Tipus 1 (els xiquets no poden nadar sols):

1) Comprove si s'ha utilitzat la paraula *synchronized* al implementar les operacions de la piscina de tipus 1. S'observaria alguna diferència d'execució entre utilitzar la paraula *synchronized* o no utilitzar-la?

Si que s'ha utilizat, si que hi hauria diferencia perque si no estagera es podrien produir condicions de carrera

2) Es poden produir condicions de carrera? Per què?

No ya que hem utilizat el synchronized per a fer atomica la seccio critica.

3) Com s'ha representat l'estat intern de la piscina? Per què s'ha fet així?

Amb dos contadors, per a diferenciar els que estan nadant dels que esperen a poder entrar
i dels que estan descansant als que esperen a poder eixir

4) Per a avisar del nou estat, s'ha fet ús de *notifyAll*(). Es podria haver utilitzat simplement *notify*()? ¿Per què?

No perque notify desperte a un xiquet en concret i el que volem es despertarlos a tots els xiquets per a que comproben si poden segir.

5) Al final de quins mètodes de Pool1 s'ha fet ús de *notifyAll*()? S'ha utilitzat al final de tots els mètodes? Si es així, analitze si és obligatori utilitzar-lo en tots els mètodes, o si seria possible (o més eficient) utilitzar-lo només en uns quants (els que ho requeriren).

En kidsSwims i InstructorRest no faria falta ya que no es necessari notificar a ningu que un xiquet entra a nadar.

Activitat 2 (Pool2)

Exercici 2.1: Indique com representar l'estat de la piscina per a poder complir les normes d'ús indicades per a la piscina tipus 2 (recorde que també ha de complir amb les normes de la piscina tipus 1). Per a açò, complete la taula corresponent a *Pool2*:

Pool2	cal esperar si ...	modifica estat ...	Avisa a ....
kidSwims()	No hiha instructor nadant o hihan masa xiquets nadant en un instructor	si	No
kidRests()	No	Si	Instructors que esperen per descansar
instructorSwims()	No	Si	Xiquets, que esperen per nadar
instructorRests()	Si, si no hiha prou instructors per xiquets.	Si	No

Exercici 2.2: Modifique el codi de *Pool2* necessari perquè es complisquen les regles d'utilització de la piscina tipus 2.

Exercici 2.3: Verifique el funcionament de *Pool2* modificant el nombre de xiquets i/o instructors.

Preguntes Piscina Tipus 2 (màxim de xiquets per instructor):

1) Per a representar l'estat del objecte compartit, seria suficient amb utilitzar una variable de tipus enter (ex. *nSwimKids*) i una variable de tipus *boolean* (ex. *instInPool*)? ¿Per què?

No, perque nessecita saber quants instructors hihan nadant per a saber quants xiquets poden entrar a nadar.

2) A quins mètodes afecta esta nova regla? (és a dir, en quins mètodes de Pool2 s'han realitzat modificacions?)

kidSwims(), instructorRests() i init().

3) Quan un instructor entra en la piscina Pool2, s'ha d'invocar *notifyAll*()? Per qué?

Per a avisar a als xiquets que estigen esperant a poder entrar a la picina per falta de instructors

Activitat 3 (Pool3)

Exercici 3.1: Indique com representar l'estat de la piscina de tipus 3 per a poder complir amb totes les seues normes d'ús. Per a açò, complete la taula corresponent a *Pool3*:

Pool3	cal esperar si ...	modifica estat ...	Avisa a ....
kidSwims()	Si no hihan monitors Hiha masa xiquets per monitor Aforament maxim	Si	No
kidRests()	No	Si	Tots
instructorSwims()	Si aforament maxim	Si	Xiquets
instructorRests()	Si no hi ha prou monitors	Si	Tots

Exercici 3.2: Modifique el codi de Pool3 perquè es complisquen les regles d'utilització de la piscina de tipus 3.

Exercici 3.3: Verifique el funcionament de Pool3 modificant el nombre de xiquets i/o instructors. Anote a continuació les seues observacions.

Preguntes Piscina Tipus 3 (capacitat màxima):

1) Per a representar l'estat de la piscina, es requereix afegir alguna altra variable a l'estat de la piscina respecte a la implementació de Pool2? Quina? Per què?

Si, la capacitat maxima, per a controlar la gent que pot entrar dins la picina, kids + instructors

2) Podríem dir que la piscina Pool3 està actuant como un "monitor"? I les piscines anteriors?

Si ya que estan ordenant les instruccions que es van a permetre, aço tambe ocorre en el pool1 i en el pool2

Activitat 4 (Pool4)

Exercici 4.1: Indique com representar l'estat de la piscina de tipus 4 per a poder complir amb totes les seues normes d'ús. Per a açò, complete la taula corresponent a *Pool4*:

Pool4	cal esperar si …	modifica estat …	Avisa a ….
kidSwims()	Si hiha instructors Si hiha prou instructors per xiquets No supera el aforo No hiha un intructor esperant per a eixir	Si	No
kidRests()	No	Si	Tots
instructorSwims()	Si supera el aforo	Si	Xiquets
instructorRests()	Si no hi ha prou monitors	Si	Tots

Exercici 4.2: Modifique el codi de *Pool4* perquè es complisquen les regles d'utilització de la piscina tipus 4.

Exercici 4.3: Verifique el funcionament de *Pool4* modificant el nombre de xiquets i/o instructors.

Preguntes Piscina Tipus 4 (si instructors esperen, no poden entrar xiquets):

1) Per a representar l'estat de la piscina, es requereix afegir alguna altra variable a l'estat de la piscina respecte a la implementació de Pool2? Quina? Per què?

Si, la variable instructorEspera, per a controlar els monitors que volen eixir.

2) Al final de quins mètodes de Pool4 s'ha fet ús de *notifyAll*()? S'ha utilitzat al final de tots els mètodes? Si es així, analitze si és obligatori utilitzar-lo en tots els mètodes, o si seria possible (o més eficient) utilitzar-lo només en uns quants (els que ho requeriren).

Tots menys kidSwims(), ya que ningu espera a que acabe.