Universite Paris Diderot – Paris 7 **Programmation reseau**
L Informatique **Anne** **ème semestre**

TD n°

Programmation concurrente en Java

Exercice 1 [r a , a n É c r n c a q ´ m a r r r a . p r m a r r a o a c r o "Bonjour !" n a a n p a r n c o n n r c a q a c a g . m r a o a c r c n q o "Salut !", n a a n p a r c o n n r c a q a c a g .

Indication : La méthode statique `sleep` *de la classe* `Thread` *permet de mettre un thread en attente.*

Exercice 2 [a r a par ag´ n r r a n o a ro r o c r n c ' n ´ m n an n a a onn´. o r c a, a a ong p q ' n con an A E , on p o p r n c r a r c r c n ´p n a m n an c o ´, n con an a con par a n no a r a ra a an n para . É c r r n programm c an a r c r c c m an r ; programm ra a c r n m ag po r c aq o cc rr n c ro ´ , p a c r nom r o a 'o cc r n c ' ´ m n an a- a . o r c r c c , o po r a c a CompteurConcurrent q o ro r c - o .

```
c a Comp rCon c rr n {
    pr va n cp ;

    p c n g va { r rn cp ; }

    p c S r ng oS r ng { r rn cp ; }

    yn c ron p c o n cr m n {
        cp ;
    }

    yn c ron p c o n cr m n n {
        cp cp ;
    }
}
```

Exercice 3 [S yn c ron a on n r r a ' ag c ' mp ´m n r n y m pro c r / con o m a r . C pro c r / con o m a r c om n q n

par n r par ag´ po an con n r n c a n c ara c r . S r
, n pro c r a ro r ap r gna n a o on
q ’ ’a r ap , a con ra r r con n n c a n c ara c r c’ -a-
r q ’ p n , pro c r o a n r q ’ n con ormaa r con ormaa
c q ’ a an r. an a con ormaa r, aa r c q ’ a
an r, a arr , r gna c a a o on , non
a n q ’ n pro c r pro n onn´ an r.

1. Dan n pr a r aps, c a a Consommateur Producteur n an
a c a ArrayBlockingQueue po r r a c par ap
pro c n con ormaa n.

. Dan n a aps, r´ ´c r c a a Consommateur Producteur
a c c o -c n c a Buffer ap ´a n ´ par o o n.

Indication : Pour cet exercice, on utilisera les méthodes `wait` *et* `notifyAll` *de la classe* `Thread`*. Lorsque la méthode* `wait` *est invoquée à partir d'une méthode* `synchronized`*, en même temps que l'exécution est suspendue, le verrou posé sur l'objet par lequel la méthode a été invoquée est relâché. Dès que la condition de réveil survient, le thread attend de pouvoir reprendre le verrou et continuer l'exécution. La méthode* `notify` *réveille un seul thread. Si plusieurs threads sont en attente, c'est celui qui a été suspendu le plus longtemps qui est réveillé. Lorque plusieurs threads sont en attente et qu'on veut tous les réveiller, il faut utiliser la méthode* `notifyAll`*.*