MODALITÀ DI LABORATORIO ONLINE

Per le esercitazioni di oggi, troverete il meeting “Lezione 9” nel canale “Lezioni” del Teams del corso. Il Docente presenterà gli esercizi da svolgere: per ogni esercizio avremo una breve introduzione seguita dallo svolgimento da parte degli studenti. Potete usare la chat per richiedere aiuto da parte del Docente. Usare il microfono solo per comunicare qualche urgenza.

Esercizio 1: Interferenza tra Thread

Scrivere un programma che incrementa un contatore intero per 40000 volte creando una classe Counter e una classe Incrementatore con il main che istanzia e usa Counter, verificando poi il valore di Counter stampandolo. Implementare tre versioni:

1. Senza thread.
2. Creando 4 thread che tutti insieme incrementano di 10000 volte il contatore, SENZA curarsi della race condition, e vedere il risultato.
3. Creando 4 thread, curandosi della race condition, e verificare il risultato.

Esercizio 2: Efficienza con i Thread

Scrivere un programma Java per inizializzare un array di 1200000 interi al valore 42. Il programma deve misurare il tempo di esecuzione e stamparlo Implementare tre versioni:

1. Senza creare nuovi thread (ovvero, con il solo main thread).
2. Con 2 thread.
3. Con 4 thread.

Calcolare lo speedup del programma con 4 thread.

Esercizio 3: Efficienza con i Thread

Scrivere un programma Java che usi i thread per **calcolare la somma di un array** di 1200000 elementi, utilizzando un numero di thread variabile pari a 1, 2, 4, 8, 16 (oppure da 1 fino a 16) e stampando i tempi ottenuti in ciascuno dei casi, e calcolando lo speedup.

Esercizio 4: Efficienza con i Thread

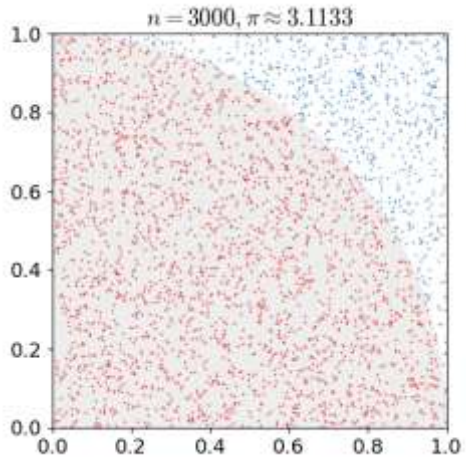
Scrivere un programma programma che usa solo i metodi indicati per stampare solo smiley " : -)". Utilizzare il codice sottostante e inserire il proprio codice solo dove indicato.

```
public class public class Smiley extends Thread {

 public void run() {
   while(true) {
     try {
       // INSERISCI IL TUO CODICE QUI
     } catch (InterruptedException e)
     { e.printStackTrace(); }
   }
 }

 private void printParentesi() throws InterruptedException{
   System.out.println(")"); Thread.sleep(100);
 }

 private void printTrattino() throws InterruptedException{
   System.out.print("-"); Thread.sleep(100);
 }

 private void printDuePunti() throws InterruptedException{
   System.out.print(":"); Thread.sleep(100);
 }

 public static void main(String[] args) {
   new Smiley().start();
   new Smiley().start();
 }

}
```

Esercizio 6: Calcolo di PI

Scrivere un programma che calcola di π con il Metodo Montecarlo.

Questo quadrato ha area 1, mentre l'area interna è $\pi/4$ (questo perche l'area di un cerchio è πr^2 , in questo caso il raggio e' 1, quindi l'area interna è π.
Se prendiamo punti random con x e y tra 0 e 1, la probabilità che cadano dentro al cerchio è (area_semicerchio/area_quadrato) = $\pi/4$
Per vedere se cadono dentro, basta vedere se x*x+y*y <1. Quindi se scegliamo N punti, e C di questi sono dentro, la frazione C/N dovrebbe approssimare $\pi/4$.
Quindi, e' possibile calcolare un valore approssimato di π calcolando 4*C/N, per N abbastanza grande.



Scrivere un programma Java per **calcolare** π. Implementare prima una versione sequenziale e, successivamente, una versione con piu thread, utilizzando un numero di thread variabile pari a 1, 2, 4, 8, 16 (oppure da 1 fino a 16), stampando i tempi ottenuti in ciascuno dei casi, e calcolando lo speedup.

Altri Esercizi

Scrivere un programma con 2 thread, che va sempre in deadlock.

Scrivere un programma con 3 thread, che va sempre in deadlock.

Scrivere un programma dove un numero n (alto) di thread fa incremento di un contatore (inizializzato a 0) per un numero m (alto) di volte. Quando l'incremento del contatore non è in mutua esclusione, causare un errore (alla fine il contatore NON è m*n). Inserire tecniche di mutua esclusione di tipo vario e verificare l'impatto sulle prestazioni.