

Cognome e Nome:

Matricola:

Prova 3 (fino a 4 punti)

Si descriva l'output che può produrre la seguente applicazione Java e dire se l'applicazione termina oppure no.

```
public class Prova3_20180622_A {
    public static Semaphore sem = new Semaphore(1);

    public static void main(String[] args) throws InterruptedException {
        MyThread[] threads = new MyThread[8];
        for (int i = 0; i < threads.length; i++) {
            threads[i] = new MyThread(i, threads);
            threads[i].start();
        }
    }

    static class MyThread extends Thread {
        private int myId;
        private MyThread[] threads;

        public MyThread(int id, MyThread[] t) {
            this.myId = id;
            this.threads = t;
        }

        public void run() {
            try {
                if (myId % 2 == 0) {
                    sem.acquire();
                }
                System.out.println("T" + myId + " " + getState());
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Prova 4 (fino a 18 punti)

Si consideri un trenino panoramico di un parco avventura che gira su una rotaia circolare. Il trenino è composto da 10 cabine (cabine[]) ed ogni cabina può ospitare al massimo 10 turisti. Ogni cabina impiega 10 “scatti” per fare un giro completo del parco avventura e le varie cabine sono equidistanti una dalle altre. I turisti possono salire sulle cabine da un unico punto (*punto di accesso*) dove troveranno un *impiegato* che effettua le seguenti operazioni in maniera ciclica:

- 1) Permette ai turisti che si trovano nella cabina che è appena arrivata al punto di accesso di scendere.
- 2) Permette ad un gruppo di 10 turisti di salire nella cabina che è appena arrivata al punto di accesso (per ragione di semplicità, vengono scelti 10 turisti a caso). Se non sono presenti almeno 10 turisti passa direttamente all'operazione successiva.
- 3) Da il comando per effettuare uno *scatto*. Uno scatto del trenino dura 30 secondi. Tutte le cabine vengono mosse in avanti.

Si modelli il sistema descritto in Java, dove *impiegato* e *turisti* sono dei thread che interagiscono tramite un oggetto *Trenino* che espone (almeno) i seguenti metodi:

- **void turSali():** il turista vuole salire sul trenino. Si tratta di un metodo che blocca il turista sul punto di accesso fin quando non sale su una delle cabine del trenino.
- **void turScendi():** il turista scende dal trenino. Si tratta di un metodo che blocca il turista fin quando

Cognome e Nome:

Matricola:

la cabina non ritorna al punto di accesso.

- **void impFaiScendere():** se nella cabina che è appena arrivata al punto di accesso sono presenti dei turisti l'impiegato fa scendere i turisti.
- **void impFaiSalire():** l'impiegato fa salire un gruppo di 10 turisti nella cabina che è arrivata al punto di accesso.
- **void impMuovi():** l'impiegato dà il comando per far fare uno scatto alle cabine del trenino.

Prova 4a (fino a 9 punti)

Si implementi la classe **Trenino** (astratta), **Impiegato** e **Turista**, e una soluzione che riproduca il funzionamento del problema sopra descritto utilizzando la classe **Semaphore** (usare solo i metodi *acquire* e *release*) del package **java.util.concurrent**.

Prova 4b (fino a 9 punti)

Si implementi una soluzione che riproduca il funzionamento del problema sopra descritto utilizzando gli strumenti di mutua esclusione e sincronizzazione del package **java.util.concurrent.locks**.

N.B.: 1): Si chiede di commentare il codice e/o aggiungere delle stampe a video. 2): Si chiede di scrivere un breve commento per ogni semaforo o condition. 3) Si prega di numerare i fogli dove sono svolti gli esercizi. 4) Si prega di usare una calligrafia leggibile.