Esercizio 2

Si consideri un sistema software $\mathcal{S}_2$ che contenga $r > 0$ *risorse* numerate da 0 a $r - 1$ mediante un identificativo univoco. Ogni risorsa è un oggetto che implementa la seguente interfaccia:

```
package it.unipr.informatica.exercise;

public interface Resource {
        public int getID();
        public int use();
        public void release();
}
```

Le risorse nel sistema vengono tutte create da un *singleton* detto *manager delle risorse*. Ogni risorsa è inizialmente in stato *libera* e passa in stato *acquisita* solo quando il manager delle risorse la restituisce mediante il metodo:

```
public Resource[] acquire(int id);
```

che ritorna un array formato dalle tre risorse con identificativi `id`, `(id+1) %` r e `(id+2) %` r purché tutte e tre le risorse siano in stato libera. Se almeno una delle tre risorse non è in stato libera, allora il metodo `acquire(int)` si mette in attesa che tutte e tre le risorse siano libere. Una risorsa ritorna in stato libera solo quando viene chiamato il metodo `release()` sulla risorsa. Solo le risorse in stato acquisita possono essere usate mediante il metodo `use()`. Il metodo `use()` chiamato su una generica risorsa con identificativo univoco `id` si limita a ritornare un numero casuale intero tra `id` e `id+99`, estremi inclusi.

Il sistema $\mathcal{S}_2$ contiene $w = r$ *worker* pensati per essere eseguiti con il massimo grado possibile di parallelismo. Ogni worker è numerato mediante un identificativo univoco tra 0 e $w - 1$. Un generico worker con identificativo univoco `id` è un oggetto con un proprio thread di esecuzione che ciclicamente chiede al manager delle risorse di acquisire tre risorse mediante `acquire(id)`. Una volta acquisite le tre risorse, il worker le usa invocando il seguente metodo:

```
public void useAndPrint(Resource r1, Resource r2, Resource r3) {
        int t = r1.use() + r2.use() + r3.use();

        System.out.println(t);
}
```

sul *singleton* di classe `it.unipr.informatica.exercise.Logger`. Il ciclo di esecuzione di ogni worker prevede che dopo ogni chiamata a `useAndPrint(Resource, Resource, Resource)`, il worker liberi le risorse utilizzate e si metta in attesa per 100ms.

Realizzare il sistema software $\mathcal{S}_2$ in Java nell'ipotesi $w = r = 9$ aggiungendo la classe `Exercise2` nel package `it.unipr.informatica.exercise` per attivare il sistema.