Esercizio 4: Azienda agricola

Un'azienda agricola è specializzata nella vendita di sacchetti di terriccio per giardino. Ogni cliente che arriva in azienda effettua le seguenti operazioni:

- Decide quanti sacchi di terriccio acquistare (un numero random compreso tra 1 e 10);
- Si presenta in cassa e paga i sacchetti (ogni sacchetto costa 3€);
- Va in magazzino e ritira i sacchetti da spostare in auto. *Ogni cliente ritira i sacchi di terriccio uno alla volta*, spendendo 1 minuto per ogni spostamento.

Si noti che i sacchi di terriccio presenti in magazzino non sono illimitati. Inizialmente sono presenti 200 sacchi di terriccio. Ogni volta che i sacchi si esauriscono un addetto magazziniere li riporta al valore iniziale impiegando per questa operazione 10 minuti. Sia davanti la cassa sia davanti al magazzino si possono formare delle code in quanto il pagamento e il ritiro dei sacchi avviene in maniera FIFO.

Azienda agricola

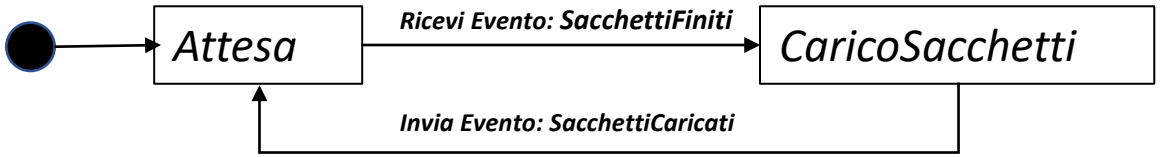
Si modelli il sistema descritto in Java, dove i *clienti* e il *magazziniere* sono dei thread che interagiscono tramite un oggetto chiamato *AziendaAgricola*. Si implementino due soluzioni che riproducano il funzionamento del problema sopra descritto utilizzando:

- gli strumenti di mutua esclusione e sincronizzazione del package java.util.concurrent.locks;
- la classe Semaphore (usare solo i metodi *acquire* e *release*) del package java.util.concurrent

Si scriva infine un main d'esempio che faccia uso di una delle due soluzioni precedenti. A tal fine bisogna simulare una giornata lavorativa in cui si presentano in azienda 100 clienti. Dopo che tutti i clienti hanno completato le loro operazioni si stampi su video l'incasso complessivo dell'azienda.

Azienda agricola

Magazziniere: *CaricaSacchetti*



Cliente: *RitiraUnSacchetto*