L'obbiettivo di questa serie d'esercizi è di imparare a riconoscere e correggere i problemi di race-conditions. Inoltre, l'obbiettivo è anche quello di imparare ad utilizzare i locks per la sincronizzazione dei threads, sia tramite la keyword "*synchronized*", sia utilizzando la classe "*ReentrantLock*".

Esercizio 1

Copiate il codice sorgente 'S2Esercizio1.java' ed analizzatelo nel dettaglio. Provate a compilarlo ed eseguirlo. Cercate di capire la natura del problema di race-condition di cui soffre. In seguito, provate a risolvere il problema utilizzando gli intrinsic locks, sia tramite synchronized block, sia tramite synchronized method. Per concludere provate a sostituire gli intrinsic locks con dei lock espliciti.

Esercizio 2

Copiate il codice sorgente 'S2Esercizio2.java' e analizzatelo nel dettaglio. Provate a compilarlo ed eseguirlo. Cercate di capire la natura del problema di race-condition di cui soffre. In seguito, provate a risolvere il problema utilizzando per iniziare gli intrinsic locks, sia tramite synchronized block, sia tramite synchronized method ed infine con dei lock espliciti.

Esercizio 3

Scrivete un programma che simuli il prelievo di soldi da un conto bancario in comune fra 5 utenti. Il programma principale dovrà creare i 5 utenti, avviare la simulazione e attendere che tutti gli utenti abbiano terminato le proprie operazioni di prelievo. Al termine della simulazione il programma dovrà verificare che il totale delle somme prelevate dagli utenti equivalga alla cifra inizialmente presente sul conto bancario.

Si utilizzi un thread indipendente per ogni utente. Per ogni iterazione del thread, l'utente dovrà prelevare un quantitativo random di soldi (tra 5$ e 50$). Ogni utente dovrà verificare che i soldi disponibili sul conto siano sufficienti ad eseguire la propria operazione. Una volta accertata la disponibilità, l'utente dovrà procedere con il prelievo. In caso contrario, il thread dovrà prelevare ciò che rimane sul conto e terminare la propria simulazione, scrivendo a schermo la somma che è riuscito a prelevare (es: *Utente2: sono riuscito a prelevare solo 15$ invece di 35$*). Contrariamente, ad operazione correttamente avvenuta, ogni utente dovrà scrivere a schermo lo stato del conto prima di prelevare, la quantità di soldi prelevata ed il nuovo stato del conto dopo l'operazione (es: *Utente3: prelevo 10$ dal conto contenente 1000$. Nuovo saldo 990$*). Per ogni utente si dovrà stabilire, in fase d'inizializzazione dei threads, un tempo di attesa tra un prelievo e quello successivo (numero casuale tra 5 e 20 ms). Il programma dovrà finire quando tutti gli utenti avranno riscontrato budget insufficiente sul conto.

Per l'implementazione si utilizzino i lock espliciti.