L'obbiettivo di questa serie d'esercizi è di esplorare le funzionalità messe a disposizione dai Concurrent Building Blocks di Java. Gli esercizi proposti si concentreranno sulle synchronized e concurrent collections e sugli algoritmi non blocking.

Esercizio 1

Copiate il codice sorgente 'S7Esercizio1.java' e analizzatelo nel dettaglio. Provate a compilarlo ed eseguirlo. Di quali problemi di concorrenzialità soffre? Provate a risolvere il problema, inizialmente con blocchi synchronized, in seguito con il supporto messo a disposizione dalle concurrent collections (principalmente quello per le compound actions).

Esercizio 2

Copiate il codice sorgente 'S7Esercizio2.java' e analizzatelo nel dettaglio. Provate a compilarlo ed eseguirlo. Osservate l'output prodotto in console. Di quali problemi legati all'esecuzione multi-thread soffre? Inizialmente, risolvete il problema utilizzando le tecniche tradizionali (metodi o blocchi synchronized). In seguito, sviluppatene una versione che utilizzi le synchronized collection e una che sfrutti le concurrent collections. Documentate le modifiche necessarie e confrontate le prestazioni delle tre versioni.

Esercizio 3

Scrivete un programma che simuli il flusso di lettere tra 2 amici. Ogni amico avrà a disposizione 150 lettere che userà per rispondere all'altro. All'inizio, ognuno dovrà inviare un numero casuale tra 2 e 5 lettere e, in seguito, potrà iniziare a rispondere alle lettere che gli sono arrivate. Introducete degli sleep (random fra 5 ms e 50 ms) fra la ricezione della lettera e l'invio della risposta. Ogni amico dovrà essere eseguito da un thread e la casella di posta in entrata dovrà essere implementata utilizzando una Collection di java. Inizialmente, provate ad usare una semplice ArrayList, assicurandovi che il programma sia thread-safe. In seguito sostituite la casella di posta con le varianti di concurrent queues discusse a lezione.

Esercizio 4

Scrivete un programma che sfrutti l'idioma di sincronizzazione CAS applicato ad una concurrent HashMap. Inizialmente, la map dovrà contenere tutti i giorni della settimana come chiavi (usate un enumeration) ed una stringa di caratteri casuali di lunghezza 10'000 come valore per ogni chiave. Inoltre, l'applicazione dovrà creare 30 threads, con il compito di scegliere a caso un giorno della settimana e d'accorciare la stringa ad esso associato di un carattere. L'operazione di sostituzione della stringa dovrà avvenire esclusivamente attraverso l'operazione di *replace* sfruttando l'idioma di sincronizzazione CAS discusso a lezione. Se necessita più di un tentativo all'interno del loop di sostituzione, ogni thread dovrà scrivere quanti tentativi ha impiegato al termine di una sostituzione riuscita.

Esempio di output:

```
...
RemoveWorker6: Updated WEDNESDAY  after 3 tries
RemoveWorker2: Updated SATURDAY  after 2 tries
RemoveWorker2: Updated MONDAY  after 4 tries
RemoveWorker4: Updated TUESDAY  after 2 tries
RemoveWorker7: Updated WEDNESDAY  after 4 tries
...
```