Esercizio 1

Si consideri un sistema software $\mathcal{S}_1$ che contenga degli oggetti *launcher* e degli oggetti *monitor* utilizzabili, rispettivamente, per attivare l'esecuzione di metodi in thread dedicati e per mettersi in attesa della loro terminazione. Gli oggetti launcher e gli oggetti monitor implementano le seguenti interfacce Java:

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

public interface Launcher {
  public Monitor launch(Runnable r);

}
```

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

public interface Monitor {
  public void await()
    throws InterruptedException;
}
```

La classe `LauncherImpl` nel package `it.unipr.informatica.exercise` può essere utilizzata per creare oggetti launcher secondo la seguente specifica:

1. Il metodo `launch(Runnable)` attiva un thread per ogni chiamata per eseguire il *corpo* del proprio argomento;
2. Il metodo `launch(Runnable)` ritorna *immediatamente* un oggetto che implementa `Monitor`;
3. È possibile utilizzare il monitor ritornato dal metodo `launch(Runnable)` per mettersi in attesa, mediante il metodo `await()`, della terminazione del corpo dell'argomento passato al metodo `launch(Runnable)`.

Si noti che un oggetto monitor si dice *sbloccato* se il metodo ad esso associato è terminato e, quindi, se una qualsiasi chiamata ad `await()` non ha più alcun effetto. Ad esempio, il seguente stralcio di codice richiede l'esecuzione del metodo `m()` dell'oggetto `this` in un thread dedicato e garantisce che il codice che segue `r.await()` venga eseguito solo dopo la terminazione del metodo `m()`:

```
Launcher l = new LauncherImpl();
Monitor r = l.launch(this::m);
r.await();
System.out.println("Esecuzione di this.m() terminata");
```

Il sistema $\mathcal{S}_1$ mette anche a disposizione degli oggetti *monitor set* che implementano l'interfaccia `MonitorSet` del package `it.unipr.informatica.exercise`. L'interfaccia `MonitorSet`, e la relativa implementazione `MonitorSetImpl`, anch'essa nel package `it.unipr.informatica.exercise`, offrono i seguenti metodi:

1. Il metodo `add(Monitor)`, che aggiunge un monitor al monitor set e ritorna `true` se il monitor è stato effettivamente aggiunto; e
2. Il metodo `await()`, che si mette in attesa che almeno uno dei monitor del monitor set risulti sbloccato, eventualmente lanciando una `InterruptedException` nei casi opportuni.

Per lo svolgimento dell'esercizio, si ipotizza che nel sistema $\mathcal{S}_1$ sia presente una classe `Exercise1` nel package `it.unipr.informatica.exercise`. Il metodo principale della classe `Exercise1` crea un monitor set, aggiunge al monitor set gli $n = 100$ monitor ottenuti dalle n richieste di esecuzione del metodo `work()` presente nella classe `Exercise1` e poi si mette in attesa che tutti gli n monitor risultino sbloccati. Il metodo `work()` ha il seguente comportamento:

1. Viene generato un intero casuale $0 \leq k < 100$;
2. Viene eseguita un'attesa di $100 + k$ millisecondi; e
3. Viene stampato k a video.

Realizzare il sistema software $\mathcal{S}_1$ in Java aggiungendo tutte le interfacce e le classi necessarie al proprio funzionamento.