Note sullo svolgimento della prova

- Non è consentito l'uso di vecchi testi d'esame, libri, o appunti.
- Utilizzare NetBeans per editare, compilare ed eseguire i programmi.
- Creare un file **Cognome_matricola.zip** contenente i soli file *.java* da consegnare.
- Al termine della prova, dopo il caricamento degli elaborati, il docente mostrerà una possibile soluzione. Dopo aver visto la soluzione, gli studenti avranno la possibilità di riprendere il proprio elaborato.

Esercizio

Un *SortingTask* è un'operazione di ordinamento di una lista di interi da eseguire in uno specifico istante nel futuro. Una volta che un *SortingTask* è stato eseguito, la lista ordinata viene memorizzata in un file ed è possibile recuperare il risultato dell'operazione attraverso il path del file. Creare una classe *SortingTask* dotata almeno dei seguenti metodi e costruttori:

- *SortingTask(List<Integer> list, long scheduledTime, int allowedDelay):* crea un *SortingTask* dove *list* è la lista di interi da ordinare, *scheduledTime* è l'istante in cui deve essere eseguito il task espresso in millisecondi secondo la convenzione *di System.currentTimeMillis()*, *allowedDelay* è il ritardo massimo con cui può essere eseguito il task rispetto al suo *scheduledTime.*
- *void execute()*: esegue l'ordinamento e memorizza il risultato in un file di testo che ha nome *scheduledTime*.txt.
- *String getResult() throws Exception:* restituisce il path del file che contiene il risultato (la lista ordinata); il metodo restituisce immediatamente il risultato se possibile, altrimenti blocca il chiamante fino a quando il task non è stato eseguito. Se l'operazione non va a buon fine (per esempio perché non è possibile salvare il risultato sul file previsto o se il ritardo rispetto al suo istante di esecuzione è troppo grande) il metodo non restituisce niente e lancia un'eccezione.
- *boolean isComplete():* restituisce *true* se il task è terminato, *false* altrimenti.
- *int getAllowedDelay():* restituisce il ritardo massimo con cui il task può essere eseguito rispetto all'istante previsto.
- *long getScheduledTime():* restituisce l'istante in cui il task deve essere eseguito, secondo la convenzione di *System.currentTimeMillis()*.
- *void setException(Exception e):* imposta l'eccezione che viene restituita al chiamante di *getResult()* nel caso in cui le cose non vadano a buon fine.

Realizzare una classe *FutureExec* che permette di eseguire task in istanti futuri usando un proprio flusso di esecuzione (un solo flusso per tutti i task). L'esecuzione di un task può essere soggetta a ritardo se il task precedente ha dei tempi di esecuzione troppo lunghi. La classe *FutureExec* deve essere dotata di un metodo *void add(SortingTask s)* che permette di aggiungere il task *s* all'insieme di task gestiti dall'esecutore. Il *FutureExec* esegue i task che ha in gestione secondo il loro *scheduledTime*. Un *SortingTask* potrebbe non essere eseguito all'istante previsto perché un task precedente è ancora in esecuzione. In questo caso il massimo ritardo ammesso è pari a *allowedDelay*. Se il ritardo è maggiore del massimo consentito il task non viene eseguito e il suo metodo *getResult()* lancerà un'eccezione quando chiamato.

Le due classi possono quindi essere utilizzate come illustrato dal seguente esempio:

```
public class Prova {
    private static List<Integer> creaLista() {
        List<Integer> l = new ArrayList<Integer>();
        for(int i=0; i<1000000; i++)
            l.add((int)(Math.random()*10000));
        return l;
    }
    public static void main(String[] args) throws Exception {
        FutureExec ef = new FutureExec();
        SortingTask c1 = new SortingTask(creaLista(), System.currentTimeMillis() + 10000, 1000);
        ef.add(c1);
        System.out.println("First SortingTask added");
        SortingTask c2 = new SortingTask(creaLista(), System.currentTimeMillis() + 10010, 100);
        ef.add(c2);
        System.out.println("Second SortingTask added");

        String r1 = c1.getResult();
        System.out.println("File path: " + r1);

        String r2 = c2.getResult();
        System.out.println("File path: " + r2);

    }
}
```

Possibile output:

```
First SortingTask added
Second SortingTask added
…
<Ten seconds>
…
File path: /Users/vecchio/Documents/prg-compiti-2023/2023-02-
03/1675367603158.txt
Exception in thread "main" esercizio1.TooMuchDelay
     at esercizio1.FutureExec.run(FutureExec.java:38)

      at java.base/java.lang.Thread.run(Thread.java:834)
```