Gestione allevamenti di capre per la produzione di latte

Il **latte di capra** è stato importante da sempre nella storia dell'umanità, ma è caduto in disuso all'inizio del XX secolo. Recentemente è tornato in auge e sta comportando la creazione di nuove imprese, soprattutto giovanili che lo propongono come **alternativa al latte di mucca**. Negli allevamenti di capre per la produzione di latte è molto importante tenere traccia di vari aspetti legati alla vita degli animali e negli ultimi anni questa attività viene fatta attraverso l'ausilio di sistemi software che permettono di gestire nel migliore dei modi tutte le attività legate all'allevamento.

Si propone quindi di realizzare un sistema che permetta, attraverso un'interfaccia grafica, di effettuare le seguenti operazioni:

- leggere da un file di testo, il cui contenuto è dettagliato alla sezione 4, i dati relativi a un insieme di capre e visualizzare i dati relativi a ogni capra in un'interfaccia simile a quella presente in figura 1, che viene mostrata a solo scopo esemplificativo (ognuno è libero di implementare la GUI nel modo che ritiene adeguato).
- poter navigare tra le capre caricate nell'applicazione in maniera sequenziale (avanti/indietro)
- poter modificare tramite la GUI i dati di ognuna delle capre inserite tramite il file di testo (non viene richiesto di inserire nuove capre oltre a quelle già presenti nel file di testo)
- scrivere i dati delle capre, con le eventuali modifiche, in un file con lo stesso formato di quello letto
- produrre un file XML con le informazioni sulle capre, il cui formato deve essere come quello descritto nella sezione 5
- ordinare l'elenco delle capre utilizzando il Merge Sort, di cui si propone una versione descritta in pseudo-codice nella sezione 6, utilizzando la data di nascita come criterio di ordinamento
- implementare l'algoritmo di ordinamento Merge Sort scritto in precedenza utilizzando i thread

Le richieste del testo sono molte, quindi difficilmente sarà possibile esaurirle tutte nelle 5 ore concesse. Si è ritenuto comunque di procedere in questo modo per dare a ognuno la possibilità di poter dimostrare le proprie competenze negli ambiti che gli sono più familiari, trascurando eventualmente le parti meno note.

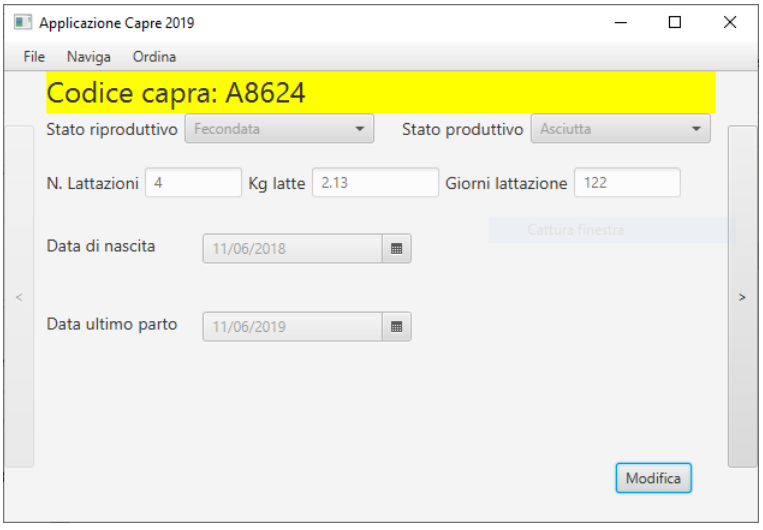

Fig. 1: Esempio di GUI

1 Criteri di valutazione

La valutazione terrà conto dei seguenti aspetti:

1. Comprensione delle specifiche. (10% del punteggio)
2. Correttezza e completezza della soluzione proposta. (20% del punteggio)
3. Chiarezza e leggibilità del codice. (10% del punteggio)
4. Utilizzo delle strutture dati più adatte. (10% del punteggio)
5. Realizzazione della GUI. (20% del punteggio)
6. Gestione degli errori (10% del punteggio)
7. Implementazione corretta dell'algoritmo di Merge Sort (10% del punteggio)
8. Utilizzo dei thread nell'implementazione del Merge Sort. (10 % del punteggio)

2 Modalità di svolgimento

La durata della prova è di 5 ore e non può essere utilizzato nessun tipo di strumento, di qualsiasi genere, che non sia stato messo a disposizione dall'organizzazione.
Eventuali domande di chiarimento, solo in forma scritta, dovranno essere poste con la seguente modalità:

- il candidato scrive la domanda in modo chiaro e leggibile su un foglio, apponendo anche il proprio nome e cognome e la consegna a un commissario incaricato della sorveglianza
- il commissario, dopo averne discusso con i colleghi, può rispondere in uno dei seguenti modi:
  - **No comment**: in questo caso la domanda non può avere una risposta perchè si ritiene che la risposta dovrebbe essere già conosciuta dal candidato e il foglio viene riconsegnato al candidato
  - **Risposta per tutti**: la domanda è pertinente e il chiarimento successivo verrà dato oralmente dal commissario a tutti i candidati, che saranno invitati a interrompere la prova e a dedicare attenzione a quanto verrà spiegato. In questo caso il foglio verrà tenuto dai commissari e il chiarimento verrà considerato parte integrante del testo.

3 Modalità di consegna

Alla fine delle 5 ore l'applicazione sviluppata andrà consegnata nel seguente modo:

- deve essere creata una cartella nominata **cognome.nome** che conterrà tutto quello che si intende consegnare
- all'interno di questa cartella dovrà essere presente:
  - il file o la cartella di progetto, in dipendenza del linguaggio/IDE utilizzato, che contenga tutto il necessario per poter aprire il progetto, visionare il codice ed eseguirlo con successo all'interno dell'IDE
  - l'eseguibile o il file JAR o altro, a seconda del linguaggio utilizzato, con le eventuali risorse esterne (file, immagini, DLL, ecc.), che possa essere eseguito facendogli doppio click sopra
  - un file README.txt, contenente tutte le osservazioni che si ritiene utile far conoscere ai professori che procederanno con la correzione. A solo titolo di esempio potrebbero essere indicazioni su come deve essere mandato in esecuzione il progetto, se necessita di accorgimenti particolari, sul perchè alcune parti non sono state sviluppate, su eventuali semplificazioni che sono state adottate, ecc.

Start	Len	Tipo	Campo	Formato
1	8	Stringa	Codice della capra	
9	2	Intero	Stato riproduttivo • 00 = Inattiva • 01 = Da fecondare • 02 = Fecondata • 03 = Gravida • 04 = Asciutta • 05 = Puerpera	00
11	2	Intero	Stato produttivo • 00 = Asciutta • 01 = In lattazione	00
13	2	Intero	Numero lattazioni	00
15	5	Reale	Ultima lattazione Kg latte	00.00
20	3	Intero	Ultima lattazione giorni	000
23	8	Data	Nascita	dd/mm/yy
31	8	Data	Ultimo parto	dd/mm/yy

Tab. 1: Descrizione del formato del file delle capre

4 Descrizione del file contenente i dati delle capre

Il file che contiene i dati su ogni singola capra è un file testuale con campi a lunghezza fissa, il cui tracciato record si può vedere nella tabella 1: come si può notare la lunghezza di ogni riga del file è di 39 bytes.

Non è garantito che ogni riga del file sia necessariamente valida, in particolare:

- possono essere presente righe vuote, che come tali vanno ignorate
- possono essere presenti errori, come campi con valori non validi o righe di lunghezza inferiore a quanto specificato, nel qual caso la riga deve essere ignorata, ma deve essere segnalata la situazione all'utente nel modo che si ritiene più opportuno
- righe corrette, ma che contengono altri caratteri in coda, che vanno lette e vanno ignorati i caratteri eccedenti.

Un esempio, che può anche essere trovato nei materiali di gara come file *file_capre.txt*, è il seguente:

```
A8624   02000402.1312211/06/1811/06/19
A8625   02000603.4512312/07/1811/06/19
A8674   01010205.3412413/11/1811/06/19
A8693   04010404.4612514/05/1811/06/19
A8724   03010303.3312622/07/1811/06/19
A8725   05000404.7112723/07/1811/06/19
```

5 Descrizione del file XML

Il file XML deve essere fatto nel seguente modo:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<capre>
 <capra codice="A8624␣␣␣">
  <stato_riproduttivo>Fecondata</stato_riproduttivo>
  <stato_produttivo>Asciutta</stato_produttivo>
  <numero_lattazioni>4</numero_lattazioni>
  <kg_latte>2.13</kg_latte>
  <ultima_lattazione>122</ultima_lattazione>
  <data_nascita>10/06/2017</data_nascita>
  <data_ultimo_parto>11/06/2019</data_ultimo_parto>
 </capra>
 <capra codice="A8693␣␣␣">
  <stato_riproduttivo>Asciutta</stato_riproduttivo>
  <stato_produttivo>In lattazione</stato_produttivo>
  <numero_lattazioni>4</numero_lattazioni>
  <kg_latte>4.46</kg_latte>
  <ultima_lattazione>125</ultima_lattazione>
  <data_nascita>14/05/2018</data_nascita>
  <data_ultimo_parto>11/06/2019</data_ultimo_parto>
 </capra>
</capre>
```

Un esempio si trova nel materiale di gara, all'interno del file *capre.xml*

6 Algoritmo di Merge Sort (fonte Wikipedia)

Il merge sort è un algoritmo di ordinamento basato su confronti che utilizza un processo di risoluzione ricorsivo, sfruttando la tecnica del Divide et Impera, che consiste nella suddivisione del problema in sottoproblemi della stessa natura di dimensione via via più piccola. Fu inventato da John von Neumann nel 1945.

Concettualmente, l'algoritmo funziona nel seguente modo:

1. Se la sequenza da ordinare ha lunghezza 0 oppure 1, è già ordinata. Altrimenti:
   (a) La sequenza viene divisa (divide) in due metà (se la sequenza contiene un numero dispari di elementi, viene divisa in due sottosequenze di cui la prima ha un elemento in più della seconda)
   (b) Ognuna di queste sottosequenze viene ordinata, applicando ricorsivamente l'algoritmo (impera)
   (c) Le due sottosequenze ordinate vengono fuse (combina). Per fare questo, si estrae ripetutamente il minimo delle due sottosequenze e lo si pone nella sequenza in uscita, che risulterà ordinata

6.1 Implementazione in pseudo codice

```
function mergesort (a[], left, right)
    if left < right then
        center := (left + right) / 2
        mergesort(a, left, center)
        mergesort(a, center+1, right)
        merge(a, left, center, right)
```

```
function merge (a[], left, center, right)
    i := left
    j := center + 1
    k := 0
    b := array temp size= right-left+1

    while i <= center and j <= right do
       if a[i] <= a[j] then
          b[k] := a[i]
          i := i + 1
       else
          b[k] := a[j]
          j := j + 1
       k := k + 1
    end while

    while i <= center do
       b[k] := a[i]
       i := i + 1
       k := k + 1
    end while

    while j <= right do
       b[k] := a[j]
       j := j + 1
       k := k + 1
    end while

    for k := left to right do
       a[k] := b[k-left]
```