Programmazione Funzionale e Parallela

Corso di Laurea in Ingegneria Informatica e Automatica - A.A. 2019-2020

Esercitazione del 23 marzo 2020

Istruzioni per l'esercitazione:

- Aprite il form di consegna in un browser e loggatevi con le vostre credenziali `uniroma1`.
- Scaricate e decomprimete sulla scrivania il codice dell'esercitazione. Vi sarà una sotto-directory separata per ciascun esercizio di programmazione. Non modificate in alcun modo i programmi di test `E*Main.scala`.
- Rinominare la directory chiamandola `cognome.nome`. Sulle postazioni del laboratorio sarà `/home/studente/Desktop/cognome.nome/`.
- È possibile consultare appunti/libri e il materiale didattico online.
- Rispondete alle domande online sul modulo di consegna.
- **Finiti gli esercizi**, e non oltre le 23:00:
  - **zippate la directory di lavoro** in `cognome.nome.zip` (`zip -r cognome.nome.zip cognome.nome/`).
- **Per consegnare**:
  - inserite nel form di consegna come autovalutazione il punteggio di ciascuno dei test forniti (inserite zero se l'esercizio non è stato svolto, non compila, o dà errore di esecuzione).
  - fate **upload** del file `cognome.nome.zip`.
- È possibile consultare la documentazione delle API di Scala, in particolare quelle sulle liste, e la dispensa Scala.
- **Se avete domande** accedete a Google Meet all'indirizzo meet.google.com/rey-xdve-rug durante orario 14:00-16:00 stabilito per l'esercitazione accedendo con la vostra **mail istituzionale**. Troverete online il docente e il tutor del corso. In alternativa, scrivete via email.

Per maggiori informazioni fate riferimento al regolamento delle esercitazioni.

Esercizio 1 (verifica su un albero binario)

Si vuole verificare la proprietà di un albero binario che l'elemento contenuto in ogni suo nodo `v` sia maggiore o uguale all'elemento nella radice del sottoalbero sinistro di `v` (se non vuoto) e minore o uguale all'elemento nella radice del sottoalbero destro di `v` (se non vuoto). Scrivere un metodo `treeTest` che, dato un albero binario con elementi interi, restituisce `true` se l'albero soddisfa la proprietà, e `false` altrimenti.

Scrivere la soluzione nel file `E1.scala` e usare il programma di prova `E1Main.scala`.

Nota: Per estrarre informazioni sui sottoalberi si suggerisce di usare un `match ... case` annidato in quello esterno.

Esercizio 2 (un semplice modello 2D)

Scrivere un metodo `getModel` che, dato un intero `n`, restituisce una lista di `n` cerchi di cui l'i-esimo cerchio, per `i=1..n`, ha `x=y=r=0.5*i/n`, dove `(x,y)` sono le coordinate del centro e `r` è il raggio. L'origine degli assi è nell'angolo inferiore sinistro e il disegno è confinato in uno spazio quadrato di coordinate comprese tra `0.0` e `1.0`. Usare la classe `case class Circle(x:Double, y:Double, r:Double) extends Shape` definita in `Frame2D.scala`, dove x e y sono le coordinate del centro ed r il raggio.

Scrivere la soluzione nel file `E2.scala` e compilarla insieme al modulo grafico `Frame2D.scala` e al programma di prova `E2Main.scala`.

Nota: su alcune versioni di Java recenti è necessario cambiare l'ultima riga di codice di `Frame2D.scala` in `setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE)`. Bisogna altresì importare `WindowConstants` da `javax.swing`.

Esercizio 3 (query su database)

Aggiungi alla classe `DB` un metodo `registiConFilm(p:Film=>Boolean):List[Regista]` che estrae tutti i registi che hanno diretto almeno un film con la proprietà `p`.

```
case class Film(id:Int, titolo:String, anno:Int)
case class Regista(id:Int, nome:String)
case class DirettoDa(idFilm:Int, idRegista:Int)

case class DB(film:List[Film], registi:List[Regista], regie:List[DirettoDa]) {
    def registiConFilm(p:Film=>Boolean):List[Regista] = Nil // da completare...
}
```

Nota: Se ritenunto utile, è possibile aggiungere alla classe `DB` variabili di istanza e metodi privati a piacere.

Scrivere la soluzione nel file `E3.scala` e usare il programma di prova `E3Main.scala`.

Esercizio 4 (anagrammi)

Scrivere un metodo `isAnagramOf(a:String, b:String):Boolean` che verifica se `a` è un anagramma di `b`, cioè `a` può essere ottenuto come permutazione delle lettere di `b`. Il test deve essere case sensitive.

Suggerimento: usare il metodo `sorted` applicato alle stringhe.

Scrivere la soluzione nel file `E4.scala` e usare il programma di prova `E4Main.scala`.