Esame di Programmazione II, 20 giugno 2024

(si consegni `AbstractComponent.java`, `FileComponent.java` e `DirectoryComponent.java`)

Si crei un progetto Eclipse e il package ***`it.univr.file`****. Si copino al suo interno le classi del compito. Non si modifichino le dichiarazioni dei metodi e delle classi. Si possono definire altri campi, metodi o costruttori non richiesti dal compito, ma devono essere* ***`private`****. Si possono definire altre classi, che in tal caso vanno consegnate. La soluzione che verrà consegnata dovrà compilare, altrimenti non verrà corretta.*

Un file system permette di memorizzare *componenti*, che possono essere file oppure directory (dette anche *folder* o *raccoglitori*). Un file è un documento con una dimensione in byte. Una directory è un contenitore di zero o più sottocomponenti (i suoi *figli*), che a loro volta possono essere sia file che directory. Si noti che sia i file che le directory hanno un nome, ma solo le directory hanno dei figli. Sia file che directory hanno una dimensione in byte. Nel caso di un file, questa dimensione è specificata dal file stesso (il suo numero di byte) mentre nel caso di una directory la dimensione è data da 100 byte più la dimensione dei sui figli, ricorsivamente.

Per esempio il seguente programma crea un albero di file e directory radicato in `root`, lo stampa, ne stampa la dimensione, stampa la lista dei file raggiungibili a partire da `root` e infine stampa qual è il percorso, a partire da `root`, che porta ai file `dog.gif` e `Pluto.c`:

```
package it.univr.file;

import java.io.FileNotFoundException;

public class MainFiles {

  public static void main(String[] args) throws FileNotFoundException {
    Component f1 = new FileComponent("cat1.jpg", 34590);
    Component f2 = new FileComponent("dog.gif", 12422);
    Component f3 = new FileComponent("cat2.jpg", 52402);
    Component images = new DirectoryComponent("images", f1, f2, f3);
    Component music = new DirectoryComponent("music"); // directory vuota
    Component f4 = new FileComponent("Pippo.java", 3255);
    Component f5 = new FileComponent("Paperino.c", 44341);
    Component work = new DirectoryComponent("work", f4, f5);
    Component f6 = new FileComponent("passwords.txt", 3233);
    Component root = new DirectoryComponent("root", work, images, f6, music);
    System.out.println(root);
    System.out.println();
    System.out.println("total size: " + root.size() + " bytes");
    System.out.println("files: " + root.getFiles());
    System.out.println("dog.gif si trova come " + root.find("dog.gif"));
    System.out.println("Pluto.c si trova come " + root.find("Pluto.c")); // eccezione!
  }
}
```

La sua esecuzione dovrà stampare:

```
root/
  images/
    cat1.jpg
    cat2.jpg
    dog.gif
  music/
  passwords.txt
  work/
    Paperino.c
    Pippo.java

total size: 150643 bytes
files: [cat1.jpg, cat2.jpg, dog.gif, passwords.txt, Paperino.c, Pippo.java]
dog.gif si trova come root/images/dog.gif
Exception in thread "main" java.io.FileNotFoundException: Pluto.c
```

Si noti che la stampa della directory avviene in ordine alfabetico crescente per nome della componente. Si noti inoltre che `Pluto.c` non esiste, quindi la sua ricerca finisce in eccezione.

Esercizio 1 (7 punti). L'interfaccia `Component.java` è già scritta e completa. Non va modificata, ma si legga con attenzione i commenti dei suoi metodi, perché dovranno essere implementati dalle sottoclassi. Si completi quindi la sua sottoclasse astratta `AbstractComponent.java` che implementa le parti comuni a file e directory, cioè soltanto `getName()` e `toString()`: tutti gli altri metodi rimarranno astratti e verrano implementati nelle due sottoclassi concrete nei prossimi esercizi.

Esercizio 2 (10 punti). Si completi la sottoclasse concreta `FileComponent.java` della classe astratta `AbstractComponent.java`. Dal momento che si tratta di una classe concreta, essa dovrà implementare tutti i metodi che ancora sono astratti a questo livello della gerarchia delle classi.

Esercizio 3 (14 punti). Si completi la sottoclasse concreta `DirectoryComponent.java` della classe astratta `AbstractComponent.java`. Dal momento che si tratta di una classe concreta, essa dovrà implementare tutti i metodi che ancora sono astratti a questo livello della gerarchia delle classi.