Esame di Object Orientation (Gr. 1) – Settembre 2022

- *Scrivere immediatamente su ogni foglio che vi è stato consegnato Cognome, Nome, N° Matricola.*
- *Non è consentito consultare appunti, libri, o colleghi, né qualunque dispositivo elettronico, PENA IMMEDIATO ANNULLAMENTO DELLA PROVA*
- *Tempo a disposizione: 3 ore*

Esercizio 1

Si vuole progettare un'applicazione che gestisca i calendari degli impegni di lavoro degli impiegati di una azienda di software.

Il sistema deve essere in grado di gestire il calendario degli impegni di tutti gli impiegati dell'azienda. Per ogni impiegato il sistema deve mantenere nome, cognome ed indirizzo di e-mail. Ogni impiegato è allocato ad un progetto, nell'ambito del quale ricopre un determinato ruolo, essendo responsabile di una determinata mansione. Deve essere anche mantenuta memoria delle date di inizio e fine del periodo nel quale l'impiegato è allocato al progetto. Deve essere possibile reperire, in qualsiasi momento, anche l'insieme completo dei progetti cui ogni impiegato è stato allocato nell'ambito della sua carriera lavorativa nell'azienda.

Ad ogni impiegato deve essere associato un proprio calendario degli impegni. Ogni impiegato può assumere impegni orari, dei quali il sistema deve mantenere orario d'inizio e orario di terminazione. L'impiegato può assumere impegni soltanto nell'ambito del progetto cui è allocato in quel periodo di tempo. Per ogni impegno deve essere mantenuta una descrizione testuale. Infine, un manager può visualizzare un elenco degli impegni di tutti gli impiegati allocati su un dato progetto.

Si richiede di definire (a) Class Diagram e (b) Sequence Diagram per modellare la funzionalità di visualizzazione dell'elenco degli impegni degli impiegati allocati su un progetto, eventualmente aiutandosi con dei Mock-up. Il Class Diagram e il Sequence Diagram, intesi come modello di dominio, si possono rifare all'euristica Entity-Boundary-Control (EBC).

Esercizio 2

```
public class Directory {
      public List<File> listAllFiles(String path) {
         List<File> all = new ArrayList<File>();
         File[] list = new File(path).listFiles();
         if (list != null) {
           for (File f : list) {
             if (f.isDirectory()) {
               all.addAll(listAllFiles(f.getAbsolutePath()));
             } else {
               all.add(f.getAbsoluteFile());
             }
           }
         }
         return all;
      }
}
```

Si modelli con un sequence diagram un'invocazione del metodo `listAllFiles` della classe `Directory` riportata sopra.

Esercizio 3

```
class Animal{
        Animal(){String color="white"; }
        void printColor(){ System.out.println(color); }
}

class Dog extends Animal{
        String color;
        Dog(){ color="black"; }
        void printColor(){
                System.out.println(color);
                System.out.println(super.color);
        }
}
class Main{
        public static void main(String args[]){
                ArrayList<Animal> l=new ArrayList<Animal>();
                l.add(new Dog());
                l.add(new Animal());
                for (Animal a : l)
                        a.printColor();
        }
}
```

Si considerino le classi Java riportate sopra.

a) Il codice compila correttamente? Motivare la risposta e fornire le possibili correzione in caso di risposta negativa.

b) *Dopo aver apportato le eventuali correzioni di cui al punto 1, qual è il risultato di un'invocazione del metodo main della classe Main?*

c) *Nel codice è presente un esempio di overriding? Se sì, indicare dove. Se no, indicare come è possibile modificare il codice per introdurne un esempio.*

d) *Nel codice è presente un esempio di overloading? Se sì, indicare dove. Se no, indicare come è possibile modificare il codice per introdurne un esempio.*

Esercizio 4

Mostrare l'utilizzo delle parole chiave throw e throws nell'ambito della gestione delle eccezioni, fornendo almeno un esempio che le utilizzi entrambe nel dominio dell'interazione con un sensore di temperatura di un edificio.