Esame di Object Orientation (Gr. 2) – Luglio 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 realizzare un sistema gestionale per supportare le attività di biglietteria di un consorzio di stabilimenti termali. Il sistema permette ad un operatore, opportunamente autenticato, di emettere nuovi biglietti. I biglietti sono nominali (ovvero associati a un particolare cliente) e caratterizzati da un costo in €. Inoltre, i biglietti possono avere validità mensile, settimanale, oppure giornaliera. I biglietti mensili sono validi dal primo all'ultimo giorno di un certo mese, mentre i biglietti settimanali sono validi per una settimana dalla data di emissione. I biglietti mensili e settimanali, inoltre, sono validi in tutte le strutture del consorzio. **I biglietti giornalieri, infine, sono validi soltanto per il giorno in cui sono emessi, e soltanto per uno degli stabilimenti del consorzio, selezionato in fase di emissione.** Per motivi di contabilità, si richiede di tenere traccia di quale sia l'*operatore che ha emesso ciascun biglietto.*

Si richiede di definire un Class Diagram concettuale con tutte le classi necessarie a modellare la funzionalità di emissione di un biglietto.
Successivamente si richiede di disegnare Class Diagram e Sequence Diagram di una possibile soluzione di questa funzionalità, ad un livello di dettaglio comprendente classi (o Mock-up) dell'interfaccia utente, classi di controllo e classi di modello, eventualmente organizzate in packages basandosi sul pattern Entity-Boundary-Control (EBC).

Esercizio 2

Si modelli con un sequence diagram un'invocazione del metodo `averageMaleAge` della classe `Utility` riportato sopra.

-

```
public float averageMaleAge(List<Person> list) {
	int sum = 0;
	boolean isEmpty = list.isEmpty();
	if(isEmpty) {
		return 0;
	}
	else {
		int size = 0;
		for(Person p : list) {
			boolean isMale = p.isMale();
			if(isMale) {
				sum += p.getAge();
				size++;
			}
		}
		if(size==0) {
			return 0;
		}
		else {
			return (float)sum/size;
		}
	}
}
```

Esercizio 3

Si considerino le classi Java riportate sotto.

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 `Ex3`?
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.

```
public class Ex3 {
	public static void main(String[] args) {
		Entity e1 = new Entity(), e2 = new Player();
		Player p1 = new Entity(), p2 = new Player();
		try {
			e2.greet(); e1.greet(); //prima e2, poi e1
		} catch(Exception e) {
			System.out.println("Cought!");
		}
		p1.greet(); p2.greet();
	}
}

public class Entity {
	public void greet() {
		System.out.println("I'm an entity");
		throw new Exception("Oops!");
	}
}

public class Player extends Entity {
	public void greet() {
		System.out.println("I'm the Player!");
	}
}
```

Esercizio 4

Spiegare quali sono le motivazioni e i vantaggi per l'adozione di una tecnica di documentazione del codice con javadoc.

Esemplificare la documentazione che potrebbe essere aggiunta nell'ambito del codice dell'Esercizio 2.

Spiegare il comportamento della utility javadoc ed esemplificare l'output che fornirebbe applicato sempre all'esercizio 2 (documentato).