Esame di Object Orientation – 29 Gennaio 2020

- *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 (punti 11)

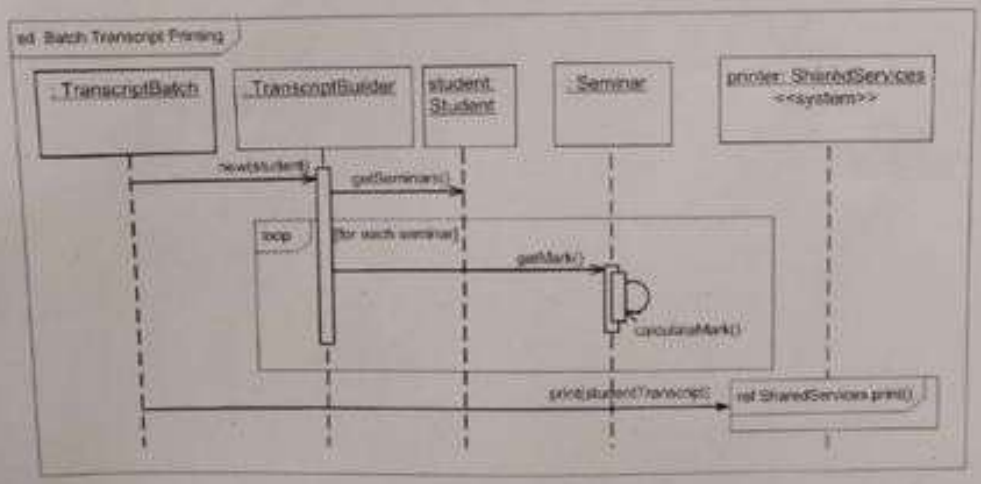
Si vuole realizzare un sistema che offra la possibilità di gestire le vendite di dispositivi elettronici in un negozio. Tali dispositivi possono essere Smartphones, Tablet o Notebook. Il cliente, in fase di acquisto, può specificare, se possibile, colore e quantità di memoria. Per ogni configurazione, ne viene mostrato al cliente il costo. Per effettuare un acquisto, il cliente deve fornire tutte le informazioni per la consegna e il pagamento.

Si richiede di:

1) Definire un Class Diagram, inteso come modello di dominio.
2) Fornire un Sequence Diagram di analisi per il caso d'uso relativo alla vendita di un notebook.

Esercizio 2 (punti 10)

Scrivere tutto il codice Java che è possibile desumere dal seguente Sequence Diagram.



Esercizio 3 (Punti 5)

Che differenza c'è tra Eccezioni Gestite e Non Gestite in Java? Fornire un esempio (diverso da quelli fatti a lezione).

Esercizio 4 (Punti 5)

Date due istanze `o1` e `o2`, che differenza c'è tra `o1.equals(o2)` e `o1 == o2`?

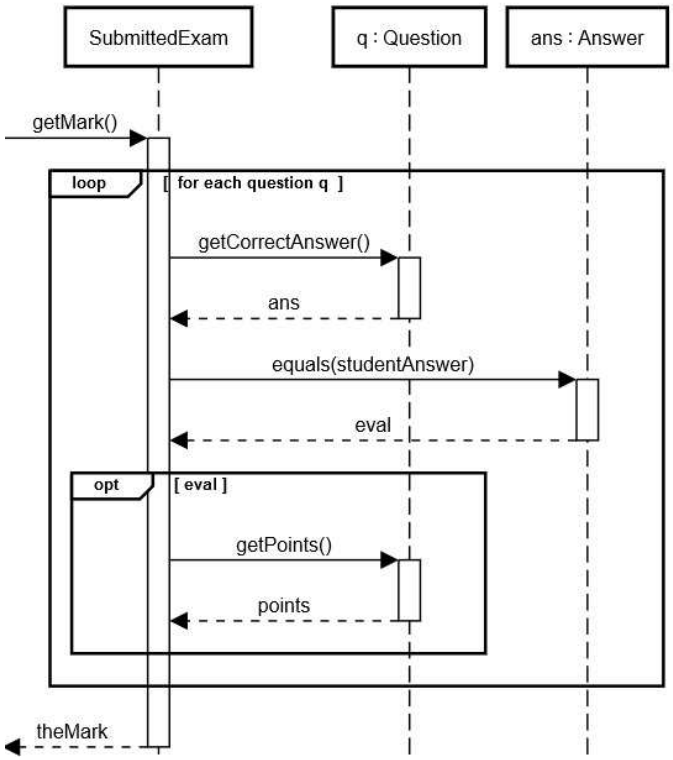
Esame di Object Orientation – 12 Febbraio 2020

- *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 (punti 11)
Si vuole realizzare un sistema informativo che offra la possibilità di gestire la registrazione di esami universitari. In particolare, un docente ha la possibilità di registrare un esame sostenuto da uno studente regolarmente iscritto, a patto che tale esame sia nel piano di studi dello studente e non sia già stato registrato.
Si richiede di:

1) Definire un Class Diagram, inteso come modello di dominio.
2) Fornire Mock-up e un Sequence Diagram di analisi per il caso d'uso relativo alla registrazione dell'esame.

Esercizio 2 (punti 10)
Scrivere tutto il codice Java che è possibile desumere dal seguente Sequence Diagram.



Esercizio 3 (Punti 5)
Descrivere il concetto di associazione in un Class Diagram UML, e come si mappa sul codice O-O, con un esempio (diverso da quanto fatto a lezione).

Esercizio 4 (Punti 5)
Descrivere vantaggi e svantaggi (qualora ce ne siano) del Garbage Collector della JVM.

Si definisca il class diagram relativo al seguente frammento di codice Java:

```
class Esame {
  private List<Domanda> domande;
  /*...*/
  public double getVoto(List<Risposta> r){
    int corrette = 0;
    for(int i; i < r.size(); i++){
      if(r.get(i).equals(domande.get(i).getRispostaCorretta())){
        corrette += 1;
      }
   }
   return ( corrette/domande.size() ) * 100;
  }
}
```

Si vuole realizzare un sistema per l'acquisto on-line di biglietti per cinema. In particolare, un cliente potrà acquistare un biglietto per vedere un particolare film specificando lo spettacolo desiderato (se ci sono ancora posti disponibili).
Uno spettacolo è caratterizzato da una data, un orario, e una sala.
Realizzare un class diagram di dominio dello scenario descritto.

In relazione all'esercizio precedente, Scrivere il sequence diagram dell'acquisto di un biglietto.

```
Class Pet {
    public String name;
    public String getName(){return name;};
    public void faiVerso(){System.out.println("???")};
}
Class Cat extends Pet {}
Class Dog extends Pet {}
Class Bunny extends Pet {}
Class Person {
    public void playWithPet(Pet p){}
    public void playWithPet(Dog d){}
}
```

Si consideri il frammento di codice Java sopra riportato.

1. Rispetta il principio dell'incapsulamento? Motivare la risposta. Se no, come modifichereste il codice per rispettarlo?

```
Class Pet {
    public String name;
    public String getName(){return name;};
    public void faiVerso(){System.out.println("???")};
}
Class Cat extends Pet {}
Class Dog extends Pet {}
Class Bunny extends Pet {}
Class Person {
    public void playWithPet(Pet p){}
    public void playWithPet(Dog d){}
}
```

Si consideri il frammento di codice Java sopra riportato.

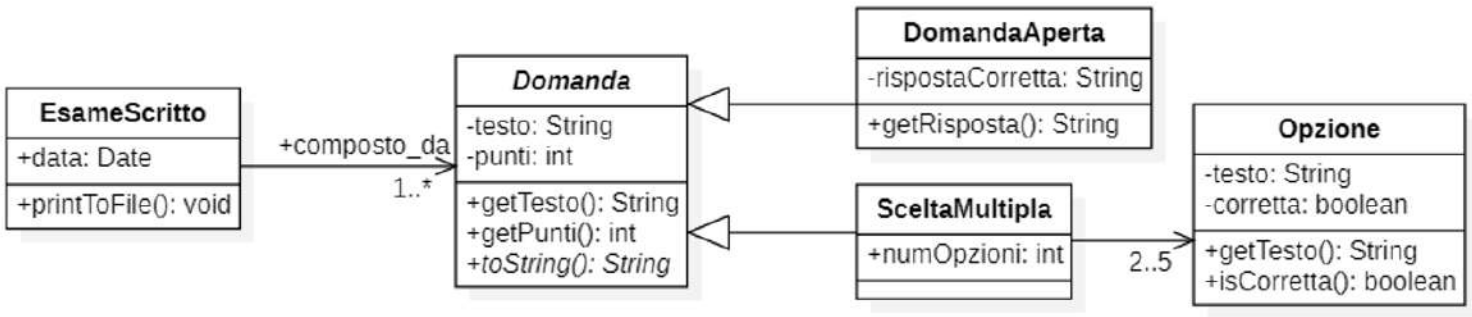
1. È presente un esempio di overriding? Motivare la risposta. Se no, indicare come modificare il codice per includere un esempio di overriding.


```
Class Pet {
    public String name;
    public String getName(){return name;};
    public void faiVerso(){System.out.println("???")};
}
Class Cat extends Pet {}
Class Dog extends Pet {}
Class Bunny extends Pet {}
Class Person {
    public void playWithPet(Pet p){}
    public void playWithPet(Dog d){}
}
```

Si consideri il frammento di codice Java sopra riportato.

1. È presente un esempio di overloading? Motivare la risposta. In caso fosse possibile, come modifichereste il codice per aggiungere un esempio di overloading?

Si scriva tutto il codice Java che è possibile desumere dal seguente class diagram.



Si vuole realizzare un sistema per la prenotazione di prestazioni sanitarie. In particolare, un paziente potrà prenotare una visita specificando il tipo di trattamento, il giorno e l'orario desiderato. Ogni prestazione sanitaria è caratterizzata da un nome, una lista di slot temporali disponibili (su base settimanale) ed una lista di prenotazioni. Il sistema dovrà consentire la prenotazione della prestazione e, nel caso lo slot selezionato non fosse disponibile, dovrà offrire la possibilità di selezionare uno slot disponibile dalla lista degli slot temporali disponibili.

Definire un Class Diagram per modellare la funzionalità di prenotazione di una prestazione, inteso come modello di dominio.

Si vuole realizzare un sistema per la prenotazione di prestazioni sanitarie. In particolare, un paziente potrà prenotare una visita specificando il tipo di trattamento, il giorno e l'orario desiderato. Ogni prestazione sanitaria è caratterizzata da un nome, una lista di slot temporali disponibili (su base settimanale) ed una lista di prenotazioni. Il sistema dovrà consentire la prenotazione della prestazione e, nel caso lo slot selezionato non fosse disponibile, dovrà offrire la possibilità di selezionare uno slot disponibile dalla lista degli slot temporali disponibili.

In relazione all'esercizio precedente, fornire i Mock-up e un Sequence Diagram di analisi della prenotazione di una prestazione sanitaria.

```
Class A {
    ArrayList<Persona> p;
    public void foo() {
        try{
           System.out.println("Length:");
           System.out.println(p.length());
           System.out.println("Done");
        } catch(ArrayIndexOutOfBound e){
           System.out.println("ArrayIndexOutOfBound");
        } catch(NullPointerException e) {
           System.out.println("NullPointerException");
        }
        System.out.println("Exit");
    }
}
```

Si consideri il frammento di codice Java sopra riportato:

- La classe A compila correttamente? (si assuma che tutti gli `import` necessari siano stati correttamente specificati)
- Se la risposta è sì, scrivere l'output di un'invocazione del metodo `foo()`. Se il risultato è il lancio di un'eccezione, specificare di quale eccezione si tratta.
- Se la risposta è no, dire qual è il problema e come si può modificare il codice per farlo compilare.

```java
class Vehicle {
    void makeSound(){};
}

class Car extends Vehicle {
    void makeSound(){System.out.println("Vroom");}
}

class Train extends Vehicle {
    void makeSound(){System.out.println("Choo Choo");}
}
```

```java
class SomeClass {

    public void foo() {
        Train t = new Train();
        Vehicle v1 = t;
        Vehicle v2 = new Car();

        t.makeSound();
        v1.makeSound();
        v2.makeSound();
    }
}
```

Si consideri il codice Java sopra riportato.

- Il codice compila correttamente? (si assuma che gli import necessari siano definiti) In caso negativo, come si potrebbe modificare il codice affinché la compilazione abbia successo?
- Qual è (considerando anche le eventuali modifiche apportate al punto precedente) l'output su `stdout` di un'invocazione del metodo `foo()` in `SomeClass`?
- È presente un esempio di polimorfismo? Motivare la risposta. In caso affermativo, indicare in quale/quali parti di codice è presente. In caso negativo, indicare (se possibile) come modificare il codice per introdurne un esempio.

Si definisca il class diagram relativo al seguente codice.

```
public class Cassa {
	private List<Sconti> sconti;
	/*…*/
	public void calcolaTotale(Carrello c) {
		float totale = 0.0;
		for(Item i : c.getItems()){
			totale = totale + this.calcolaPrezzo(item, sconti);
		}
		return totale;
	}
}
```

Un rinomato lido balneare vuole realizzare un sistema di prenotazione per l'accesso alla spiaggia nel periodo d'emergenza da CoViD-19. In particolare, gli utenti potranno acquistare un accesso di tipo giornaliero, settimanale o mensile caratterizzati da diverse percentuali di sconto. Il costo e la disponibilità dipenderanno dal tipo di accesso, dal numero di persone che intendono accedere alla spiaggia, dalla fila di ombrelloni selezionata in fase d'acquisto. Si noti che ogni fila di ombrelloni ha una capienza limitata.

Definire un Class Diagram per modellare l'acquisto di un abbonamento settimanale, inteso come modello di dominio.

Un rinomato lido balneare vuole realizzare un sistema di prenotazione per l'accesso alla spiaggia nel periodo d'emergenza da CoViD-19. In particolare, gli utenti potranno acquistare un accesso di tipo giornaliero, settimanale o mensile caratterizzati da diverse percentuali di sconto. Il costo e la disponibilità dipenderanno dal tipo di accesso, dal numero di persone che intendono accedere alla spiaggia, dalla fila di ombrelloni selezionata in fase d'acquisto. Si noti che ogni fila di ombrelloni ha una capienza limitata.

In relazione all'esercizio precedente, fornire i Mock-up e un Sequence Diagram di analisi dell'acquisto di un abbonamento settimanale.

```java
public class Yell {

	public void say(String message) {
		System.out.print(message);
		}
}

public class Echo extends Yell{
	public void say(String message, int times) {
		System.out.println("!!!");
		for(int i=0; i<times; i++)
			System.out.println(message);
	}
}
```

```java
class SomeClass {

	public void foo() {
		Yell y = new Yell();
		Echo e = new Echo();

		y.say("HELLO GUYS");
		e.say("HELLO WORD");
	}
}
```

Si consideri il codice Java sopra riportato.

- Il codice compila correttamente? (si assuma che gli import necessari siano definiti) In caso negativo, come si potrebbe modificare il codice affinché la compilazione abbia successo?
- Qual è (considerando anche le eventuali modifiche apportate al punto precedente) l'output su `stdout` di un'invocazione del metodo `foo()` in `SomeClass`?
- È presente un esempio di overriding? Motivare la risposta. In caso affermativo, indicare in quale/quali parti di codice è presente. In caso negativo, indicare (se possibile) come modificare il codice per introdurne un esempio.

```java
public interface Receiver{
	public void update(Object o);
}
public class Sms implements Receiver{
	public void update(Object o) {
		System.out.println("Offer");
		System.out.println(o);
	}
}
public class Mail implements Receiver{
	public void update(Object o) {
		System.out.println("Deal”);
		System.out.println(o);
	}
}

public class OffersFeed {
	private String offer;
	private List<Sms> receivers = new
ArrayList<>();
    public void addReceiver(Receiver rec) {
        this.receiverss.add(rec);
    }
    public void setState(String offer) {
        this.offer = offer;
        for (Receiver r : this.receivers) {
            r.update(offer);
        }
    }
}
```

```java
class SomeClass {

	public void foo() {
		OffersFeed feed =
				new OffersFeed();
		Receiver s1 = new Sms();
		Receiver s2 = new Mail();
		feed.addSender(s1);
		feed.addSender(s2);
		feed.setLastOffer("Big Offer");
	}
}
```

Si consideri il codice Java sopra riportato.

- Il codice compila correttamente? (si assuma che gli import necessari siano definiti) In caso negativo, come si potrebbe modificare il codice affinché la compilazione abbia successo?
- È presente un esempio del desgin pattern Observer? Motivare la risposta. In caso affermativo, indicare in quale/quali parti di codice è presente. In caso negativo, indicare (se possibile) come modificare il codice per introdurne un esempio.
- Qual è (considerando anche le eventuali modifiche apportate al punto precedente) l'output su `stdout` di un'invocazione del metodo `foo()` in `SomeClass`?

Traccia di Object Orientation – 10 Settembre 2020

Esercizio 1

Vi è commissionata la realizzazione di una piattaforma che permetta ai gestori di un ristorante di informatizzare il tracciamento dei propri avventori in ottemperanza alle recenti disposizioni anti-contagio.
Il sistema permette a un cameriere di registrare l'arrivo di una tavolata. In questa fase, il sistema tiene traccia della data e orario di arrivo, del tavolo assegnato, nonché dei nominativi e dei numeri di telefonino di tutti gli avventori. Ad un tavolo, possono sedere al più 6 persone. Successivamente, nel caso uno dei clienti risulti positivo al Covid-19, il sistema permette ad un responsabile della struttura di inserire la segnalazione di contagio di uno degli avventori presente in una certa data. In seguito all'inserimento della segnalazione di contagio, il sistema invia automaticamente un SMS di avviso a tutti i clienti che hanno visitato il locale il giorno della visita del cliente infetto, e nelle date successive, fino alla data corrente.

Definire un Class Diagram per modellare la registrazione di una tavolata, inteso come modello di dominio.

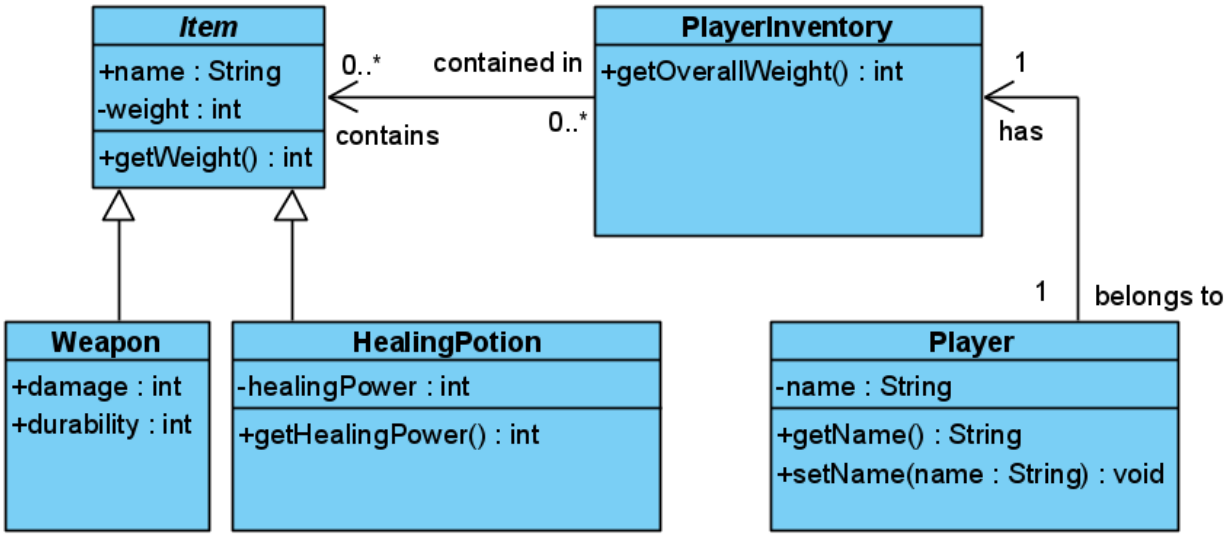
Esercizio 2

Vi è commissionata la realizzazione di una piattaforma che permetta ai gestori di un ristorante di informatizzare il tracciamento dei propri avventori in ottemperanza alle recenti disposizioni anti-contagio.
Il sistema permette a un cameriere di registrare l'arrivo di una tavolata. In questa fase, il sistema tiene traccia della data e orario di arrivo, del tavolo assegnato, nonché dei nominativi e dei numeri di telefonino di tutti gli avventori. Ad un tavolo, possono sedere al più 6 persone. Successivamente, nel caso uno dei clienti risulti positivo al Covid-19, il sistema permette ad un responsabile della struttura di inserire la segnalazione di contagio di uno degli avventori presente in una certa data. In seguito all'inserimento della segnalazione di contagio, il sistema invia automaticamente un SMS di avviso a tutti i clienti che hanno visitato il locale il giorno della visita del cliente infetto, e nelle date successive, fino alla data corrente.

In relazione all'esercizio precedente, fornire i Mock-up e un Sequence Diagram di analisi dello scenario relativo all'inserimento di una tavolata.

Esercizio 3

Si scriva tutto il codice Java che è possibile desumere dal seguente class diagram.



Esercizio 4

```java
class Greeter {
  public void greet(Person p){
    System.out.println("Hi, "+p.getName());
  }

  public void greet(Pirate p){
    System.out.println("Ahoy, "+p.getName());
  }
}

class Person {
  String name;
  public Person(String name){
    this.name=name;
  }
  public String getName(){
    return name;
  }
}
```

```java
class Pirate extends Person {
  public Pirate(String name){ super(name); }
  public String getName(){
    return name + " the pirate";
  }
}

public class TestClass{
  public void test(){
    Greeter g = new Greeter();

    Person person = new Person("Amy");
    Pirate pirate = new Pirate("Bill");
    Person test   = pirate;

    g.greet(person);
    g.greet(pirate);
    g.greet(test);
  }
}
```

Si consideri il codice Java sopra riportato.

- Il codice compila correttamente (si assuma che gli import necessari siano definiti)? In caso negativo, come si potrebbe modificare il codice affinché la compilazione abbia successo?
- Qual è (considerando anche le eventuali modifiche apportate al punto precedente) l'output su `stdout` di un'invocazione del metodo `test()` in `TestClass`?

Esercizio 5

```
class Greeter {
  public void greet(Person p){
    System.out.println("Hi, "+p.getName());
  }

  public void greet(Pirate p){
    System.out.println("Ahoy, "+p.getName());
  }
}

class Person {
  String name;
  public Person(String name){
    this.name=name;
  }
  public String getName(){
    return name;
  }
}
```

```
class Pirate extends Person {
  public Pirate(String name){ super(name); }
  public String getName(){
    return name + " the pirate";
  }
}

public class TestClass{
  public void test(){
    Greeter g = new Greeter();

    Person person = new Person("Amy");
    Pirate pirate = new Pirate("Bill");
    Person test   = pirate;

    g.greet(person);
    g.greet(pirate);
    g.greet(test);
  }
}
```

Si consideri il codice Java sopra riportato.

- È presente un esempio di *overriding*? Motivare la risposta. In caso affermativo, indicare in quale/quali parti di codice è presente.
- È Presente un esempio di *overloading*? Motivare la risposta. In caso affermativo, indicare in quale/quali parti di codice è presente.

Traccia di Object Orientation – 12 Ottobre 2020

Esercizio 1

Si realizzi un sequence diagram corrispondente all'invocazione del metodo `crawl` riportato di seguito. Si assuma che tutti gli import necessari siano definiti correttamente.

```
class Crawler {
    Browser browser   = new FakeBrowser();
    Database database = new Database();

    public void crawl(String url){
        while(url != null){
            WebPage page   = browser.visit(url);
            String nextUrl = page.getNextUrl();
            database.save(page);
        }
    }
}
```

Esercizio 2

Un gruppo di giocatori di poker vi commissiona la realizzazione di un sistema informatico per tenere traccia dei punteggi delle loro sfide. In particolare, il sistema permette a un giocatore di creare una nuova partita, specificando un luogo tra quelli presenti nel sistema, una data e un orario, una quota di partecipazione (espressa in €), e un elenco di almeno due e al più dieci giocatori partecipanti. Successivamente, il giocatore che ha creato la partita può specificare quale dei partecipanti ha vinto. Infine, tutti i giocatori possono visualizzare una schermata di riepilogo che mostra, per ogni giocatore, numero di partite giocate e numero di vittorie.

Definire un Class Diagram per modellare la creazione di una partita, inteso come modello di dominio.

Esercizio 3

Un gruppo di giocatori di poker vi commissiona la realizzazione di un sistema informatico per tenere traccia dei punteggi delle loro sfide. In particolare, il sistema permette a un giocatore di creare una nuova partita, specificando un luogo tra quelli presenti nel sistema, una data e un orario, una quota di partecipazione (espressa in €), e un elenco di almeno due e alpiù dieci giocatori partecipanti. Successivamente, il giocatore che ha creato la partita può specificare quale dei partecipanti ha vinto. Infine, tutti i giocatori possono visualizzare una schermata di riepilogo che mostra, per ogni giocatore, numero di partite giocate e numero di vittorie.

In relazione all'esercizio precedente, fornire i Mock-up e un Sequence Diagram di analisi dello scenario relativo alla creazione di una partita.

Esercizio 4

```
class Foo {
    void doSomething(Object o) {
      try {
            System.out.println("Before");
            o.toString();
            System.out.println("After");
      }
      catch(Exception e) {
            System.out.println("Exception");
      }
      finally {
            System.out.println("Done");
      }
    }
}
```

Si consideri il codice Java sopra riportato. Assumendo che tutte le import siano definite correttamente, il codice compila? Se no, indicare come modificare il codice affinché la compilazione abbia successo. Qual è l'output su `stdout` di un'invocazione del metodo `doSomething` passando come argoment `null`? Se il risultato dell'esecuzione del metodo è il lancio di un'eccezione, indicare di quale eccezione si tratta.

Esercizio 5

```
abstract class Car {
	public int maxSpeed;
	public int acceleration;
	public int range;
	abstract void revEngine();
}

class F1 extends Car {
	void revEngine() {System.out.println("Mweeeee");}
}

class ElectricCar extends Car {}

class RaceSimulator {
	//Ritorna l'automobile vincente tra "a" e "b"
	public F1 simulateRace(F1 a, F1 b){
		F1 winner=null;
		//codice di simulazione gara omesso
		return winner;
	}
}
```

Si consideri il codice Java sopra riportato.

- Il codice compila correttamente? Se no, indicare come modificare il codice affinché la compilazione abbia successo.
- Come modifichereste la classe RaceSimulator per introdurre la possibilità di simulare corse tra veicoli elettrici? E per simulare anche corse “miste”, cioè tra veicoli generici, come ad esempio un veicolo elettrico e una monoposto di Formula 1?
- Si consideri il codice ottenuto dopo le modifiche del punto precedente. È presente un esempio di overriding? È presente un esempio di overloading? Per entramble le precedenti domande, in caso di risposta positiva indicare in quale/quali parti di codice. In caso di risposta negativa, indicare – se possibile – come si potrebbe modificare ulteriormente il codice per introdurne un esempio.

Una compagnia di navigazione vuol realizzare un sistema di prenotazione on-line per l'imbarco su traghetti per la connessione con le isole. In particolare, il sistema dovrà permettere la prenotazione di biglietti per l'imbarco di persone e veicoli sia privati che commerciali. Una prenotazione può essere effettuata per una o più persone e relativi veicoli. Ogni prenotazione sarà caratterizzata da un orario d'imbarco, un orario di partenza/arrivo, la nave su cui imbarcarsi, la presenza o meno di un veicolo ed infine la tipologia di biglietti prenotati (privati o commerciali).

Definire un Class Diagram per modellare la prenotazione dei biglietti per l'imbarco di un gruppo di persone con 2 veicoli. Il Class Diagram, inteso come modello di dominio, si deve rifare all'Entity-Boundary-Control (EBC).

Una compagnia di navigazione vuol realizzare un sistema di prenotazione on-line per l'imbarco su traghetti per la connessione con le isole. In particolare, il sistema dovrà permettere la prenotazione di biglietti per l'imbarco di persone e veicoli sia privati che commerciali. Una prenotazione può essere effettuata per una o più persone e relativi veicoli. Ogni prenotazione sarà caratterizzata da un orario d'imbarco, un orario di partenza/arrivo, la nave su cui imbarcarsi, la presenza o meno di un veicolo ed infine la tipologia di biglietti prenotati (privati o commerciali).

In relazione all'esercizio precedente, fornire i Mock-up e un Sequence Diagram di analisi dello scenario relativo alla prenotazione dei biglietti per l'imbarco di un gruppo di persone con 2 veicoli. Il Sequence Diagram deve sfruttare le classi definite nel Class Diagram del punto precedente (che puo essere parzialmente esteso in questo esercizio).

Si vuole realizzare un sistema informatico per semplificare la gestione delle finanze personali. Un utente registrato può specificare entrate mensili ricorrenti (e.g.: stipendio, pensione) e uscite mensili ricorrenti (e.g.: affitto, rata dell'automobile, abbonamento per i mezzi pubblici), indicando per ciascuna entrata/uscita ricorrente il corrispettivo importo in Euro e una breve descrizione. Inoltre, un utente può inserire nel sistema entrate e uscite occasionali (ovvero, non ricorrenti), indicando una data, una causale e un importo in Euro. Infine, un utente può visualizzare un bilancio provvisorio per il mese corrente. In questo caso, il sistema mostra all'utente il bilancio provvisorio calcolando la differenza tra le entrate del mese (sia ricorrenti che occasionali) e le uscite (sia ricorrenti che occasionali).

Definire un Class Diagram per modellare la visualizzazione del bilancio provvisorio del mese corrente. Il Class Diagram, inteso come modello di dominio, si deve rifare all'Entity-Boundary-Control (EBC).

Si vuole realizzare un sistema informatico per semplificare la gestione delle finanze personali. Un utente registrato può specificare entrate mensili ricorrenti (e.g.: stipendio, pensione) e uscite mensili ricorrenti (e.g.: affitto, rata dell'automobile, abbonamento per i mezzi pubblici), indicando per ciascuna entrata/uscita ricorrente il corrispettivo importo in Euro e una breve descrizione. Inoltre, un utente può inserire nel sistema entrate e uscite occasionali (ovvero, non ricorrenti), indicando una data, una causale e un importo in Euro. Infine, un utente può visualizzare un bilancio provvisorio per il mese corrente. In questo caso, il sistema mostra all'utente il bilancio provvisorio calcolando la differenza tra le entrate del mese (sia ricorrenti che occasionali) e le uscite (sia ricorrenti che occasionali).

In relazione all'esercizio precedente, fornire i Mock-up e un Sequence Diagram di analisi dello scenario relativo alla visualizzazione del bilancio provvisorio del mese corrente. Il Sequence Diagram deve sfruttare le classi definite nel Class Diagram del punto precedente (che puo essere parzialmente esteso in questo esercizio).

```java
public class Player {
	public static void main(String[] args) {
		Content c1 = new VideoContent();
		Content c2 = new AudioContent();
		c1.setNomeFile("Funny Cat Moments");
		c2.showDetails(); c1.showDetails();
	}
}

class Content {
	private String nomeFile;
	public void showDetails() {
		System.out.println("Nome File: "+nomeFile);
	}
	public void setNomeFile(String nome) {this.nomeFile=nome;}
}

class VideoContent extends Content {
	public void showDetails() {
		super.showDetails();
		System.out.println("Video File");
	}
}

class AudioContent extends Content {
	public void showDetails() {
		System.out.println(this.nomeFile);
		System.out.println("Audio File");
	}
}
```

Si consideri il codice Java sopra riportato.

a) Il codice compila correttamente? Se no, indicare come modificare il codice affinché la compilazione abbia successo.

b) Si consideri il codice eventualmente ottenuto dalle modifiche del punto precedente. Qual è l'output di un invocazione del metodo main della classe Player?

c) Si consideri il codice eventualmente ottenuto dalle modifiche dei punti precedenti. È presente un esempio di overloading? In caso di risposta positiva indicare in quale/quali parti di codice. In caso di risposta negativa, indicare – se possibile – come si potrebbe modificare ulteriormente il codice per introdurne un esempio.

Unina Maps è un servizio che consente la ricerca e la visualizzazione di mappe geografiche di buona parte della Terra. Il servizio offre molteplici funzionalità. Un utente generico può inserire in un apposito form un indirizzo, per vederlo visualizzato su mappa (georeferenziazione). Tale utente può quindi modificare il livello di zoom, così come scorrere la mappa sull'asse X e Y. Inoltre è possibile ricercare Punti d'Interesse (PDI), tra cui ristoranti, monumenti, negozi, hotel, etc. Un utente registrato (cioè che ha effettuato un login) può inoltre, dopo aver georeferenziato un indirizzo, anche salvarlo in un suo elenco di PDI preferiti. Infine, un amministratore del sito ha la possibilità di aggiornare periodicamente le fotografie satellitari presenti nel sistema, specificando per ognuna di esse la latitudine, longitudine ed estensione.

Definire i Mock-up e un Class Diagram per modellare il salvataggio di un PDI. Il Class Diagram, inteso come modello di dominio, si deve rifare all'Entity-Boundary-Control (EBC).

Unina Maps è un servizio che consente la ricerca e la visualizzazione di mappe geografiche di buona parte della Terra. Il servizio offre molteplici funzionalità. Un utente generico può inserire in un apposito form un indirizzo, per vederlo visualizzato su mappa (georeferenziazione). Tale utente può quindi modificare il livello di zoom, così come scorrere la mappa sull'asse X e Y. Inoltre è possibile ricercare Punti d'Interesse (PDI), tra cui ristoranti, monumenti, negozi, hotel, etc. Un utente registrato (cioè che ha effettuato un login) può inoltre, dopo aver georeferenziato un indirizzo, anche salvarlo in un suo elenco di PDI preferiti. Infine, un amministratore del sito ha la possibilità di aggiornare periodicamente le fotografie satellitari presenti nel sistema, specificando per ognuna di esse la latitudine, longitudine ed estensione.

Fornire un Sequence Diagram di analisi dello scenario relativo salvataggio di un PDI. Il Sequence Diagram deve sfruttare le classi definite nel Class Diagram del punto precedente (che puo essere parzialmente esteso in questo esercizio). **Si richiede di caricare, come svolgimento di questo esercizio, anche una fotografia dei Mock-up e del Class Diagram realizzati al punto precedente.**

```java
public class Main {
  public static void main(String[] args) {
    GuessNumber g = new GuessNumber();
    try {
      System.out.println(g.guess(123));
      System.out.println(g.guess(-99));
      System.out.println(g.guess(42));
    } catch(InvalidNumberException e) {
      System.out.println(e.msg);
    }
  }
}
```

```java
class GuessNumber {
  private int number;
  public GuessNumber() {
    this.number = 42; //secret, don't tell!
  }
  public boolean guess(int n) {
    if(n<0) {
      throw new InvalidNumberException("!!!");
    }
    return n == this.number;
  }
}
```

```java
class InvalidNumberException extends Exception {
  private String msg;
  public InvalidNumberException(String msg){
    this.msg = msg;
  }
}
```

Si consideri il codice Java sopra riportato.

1. Il codice compila correttamente? Se la risposta è no, indicare quali sono gli errori riscontrati e come è possibile correggerli.
2. Qual è l'output su stdout (tenendo conto delle eventuali correzioni apportate nel punto precedente) di un'invocazione del metodo *main*?
3. Come modifichereste il codice sopra riportato per gestire anche indovinelli con String?

Dato il seguente codice Java, fornire un Sequence Diagram rappresentante un'invocazione del metodo *foo()* della classe *A*.

```java
class B {
	public void function() {
		this.compute();
	}
	private void compute() {
		/*...*/
	}
	/*...*/
}
```

```java
class A {
	private B myB = new B();
	public void setB(B b) {
		this.myB = b;
	}
	public boolean foo(){
		for (int i =0; i<10; i++)
			myB.function();
		return true;
	}
	/*...*/
}
```

Esame di Object Orientation – 21 Luglio 2021

- *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 (punti 11)

Vi è commissionata la realizzazione di un sistema informativo per la gestione di noleggi di auto, moto e camper. In particolare, un commesso, dopo essersi identificato presso il sistema, potrà registrare i noleggi da lui piazzati. Per registrare un noleggio, il commesso deve innanzitutto inserire informazioni anagrafiche (nome, cognome, codice fiscale, n° documento di riconoscimento) del cliente. Oltre ai dati anagrafici, sono richiesti un recapito telefonico e un indirizzo e-mail a cui sarà inviata la fattura. Successivamente, il commesso seleziona uno dei veicoli correntemente disponibili, e conferma il noleggio.

Si richiede di:

1) Definire un Class Diagram, inteso come modello di dominio.
2) Fornire un Sequence Diagram di analisi per il caso d'uso relativo alla registrazione di un nuovo noleggio.

Esercizio 2 (punti 10)

Modellare un Sequence Diagram dal seguente codice:

```
public class ReportPrinter {
    public static void printStudentReport(Student s) {
        CorsoDiLaurea cdl = s.getCorsoDiLaurea();
        Carriera c = s.getCarriera();
        String a = s.getInfoAnagrafiche();
        this.println( a + "\n" + cdl.detailString() + "\nCarriera:" );
        for(EsameSostenuto e: c.getEsamiSostenuti() ){
            this.println( e.nome + "-" + e.cfu + "-" + e.getVoto() );
        }
        this.println( "Media: " + c.getMedia() );
    }
    /* altro codice eventualmente omesso */
}
```

Esercizio 3 (Punti 5)

Che differenza c'è tra Eccezioni Gestite e Non Gestite in Java? Fornire un esempio (diverso da quelli fatti a lezione).

Esercizio 4 (Punti 5)

Date due istanze `o1` e `o2` in Java, descrivere la differenza tra `o1.equals(o2)` e `o1 == o2`, per 2 tipologie di tipi, mostrando con degli esempi cosa accade in memoria.

Esame di Object Orientation – 10 Settembre 2021

- *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 (punti 11)

Vi è commissionata la realizzazione di un sistema informatico per automatizzare la verifica della certificazione verde nella palestra "Gamma's Gym". Il sistema, che verrà installato su un apposito totem, regolerà gli accessi alla struttura. Un cliente, in particolare, richiederà di accedere premendo un apposito pulsante sullo schermo capacitivo, e poi inserendo il proprio codice abbonamento. Se il codice inserito è valido, il sistema richiede che venga esibita anche la certificazione verde, mostrandone il codice QR davanti ad un apposito lettore. La lettura/ validazione delle informazioni nel codice QR della certificazione verde avviene sfruttando il metodo "*boolean check(QR q)*" del package esterno "*VerificaQR*". Se la certificazione verde è valida, il sistema mostra un messaggio di benvenuto, tenendo traccia dell'avvenuto accesso. In caso contrario, il sistema mostra un messaggio d'errore e tiene traccia del tentativo d'accesso fallito.

Si richiede di:

1) Definire un Class Diagram, inteso come modello di dominio.
2) Fornire un Sequence Diagram di analisi per il caso d'uso relativo all'accesso di un utente.

Esercizio 2 (punti 10)

Modellare un Sequence Diagram dal seguente codice:

```
class WebCrawler {
    Browser browser   = new WebBrowser();
    Database database = new PostGresDatabase();

    public void crawl(String url){
        while(url != null){
            WebPage page   = browser.visit(url);
            if (page.type.equals("HTML")
                  String pageCode = page.getHTML();
            database.save(page);
        }
    }
```

Esercizio 3 (Punti 5)

Descrivere se/come il concetto di **associazione** in un Class Diagram UML si può tradurre in Java.

Esercizio 4 (Punti 5)

Descrivere vantaggi e svantaggi (qualora ce ne siano) del Garbage Collector della JVM rispetto al meccanismo di allocazione/deallocazione del C.

[redacted]

Esame di Object Orientation - 27 gennaio 2022

- Scrivere immediatamente, su ogni foglio che vi è stato consegnato, cognome, nome, numero di matricola.
- Non è consentito consultare appunti, libri, colleghi, né qualunque dispositivo elettronico, pena l'immediato annullamento della prova.
- Tempo a disposizione: 3 ore.

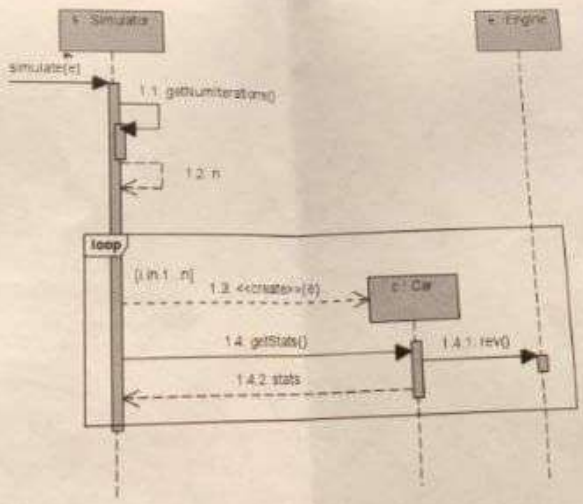
Esercizio 1

Un Corso di Laurea vi commissiona la realizzazione di "PROJEX-9000", un sistema per semplificare e uniformare la gestione dei progetti nei vari insegnamenti. Ciascun progetto è caratterizzato da un nome, una traccia testuale, un numero minimo e massimo di studenti per gruppo, e una deadline per la consegna. Uno studente, previa autenticazione, può partecipare a un progetto non ancora assegnato e formare un gruppo. A tal fine, lo studente indica un nome per il proprio gruppo, e un elenco di partecipanti (specificando gli indirizzi di posta istituzionali). Il sistema verifica che: 1) ciascuno dei partecipanti non sia già presente in altri gruppi; 2) il numero di partecipanti sia compreso tra il numero minimo e massimo di partecipanti ammesso dal progetto. Se i controlli vanno a buon fine, il gruppo viene formato e il sistema mostra a schermo un messaggio di conferma.

Definire Class Diagram e Sequence Diagram per modellare la funzionalità di creazione di un gruppo, 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

Si scriva tutto il codice Java che è possibile desumere dal seguente sequence diagram.



Esercizio 3

```
public class Exercise {
  private boolean toggle = false;

  public void foo() {
    try {
      f(toggle);
    }
    catch(MyEx e) {
      this.toggle = !this.toggle;
      System.out.println(e.getMessage());
    }
    catch(Exception e) {
      System.out.println(e.getMessage());
    }
  }

  private void f(boolean t) throws MyEx {
    if(t)
      throw new Exception("A");
    else
      throw new myException("B");
  }
}
```

```
public class MyEx extends Exception {
  public MyEx (String message) {
    super(message);
  }
}
```

```
public class Main {
  public static void main(String[] args) {
    Exercise ex = new Exercise();
    try {
      ex.foo();
      ex.foo();
      ex.foo();
    }
  }
}
```

Si consideri il codice Java sopra riportato.

a. Il codice compila correttamente? Se no, indicare come modificare il codice affinché compili.
b. Al netto delle eventuali correzioni apportate al punto (a), qual è l'output su standard output di un'invocazione del metodo main della classe?

Esercizio 4

Descrivere i concetti di classe astratta e interfaccia nell'ambito del linguaggio Java, fornendo almeno un esempio (diverso da quelli fatti a lezione) di un caso nel quale possa essere preferibile definire l'una o l'altra.

Esame di Object Orientation (Gr. 2) – 13 Giugno 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 informativo per supportare la gestione dell'attività di vendita e noleggio di imbarcazioni usate *"Stan's Previously Owned Vessels"*.

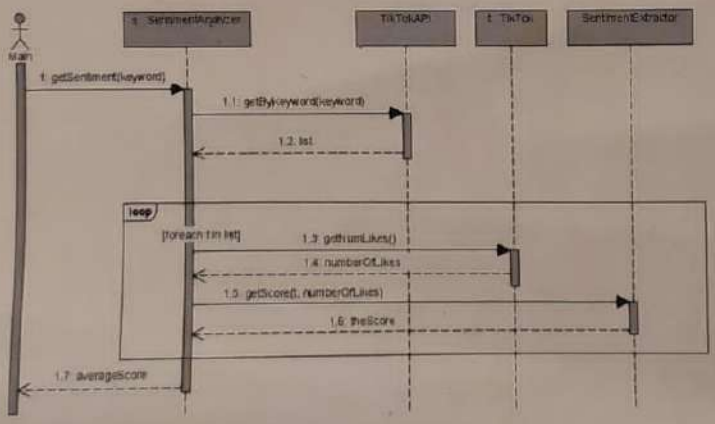
Il sistema permette all'amministratore (Stan) di gestire il parco mezzi aggiungendo, modificando, ed eliminando imbarcazioni. Le imbarcazioni sono identificate da un nome univoco, da una lunghezza (espressa in metri), e possono essere a remi (caratterizzate dal numero di remi), a vela (caratterizzate dal numero di alberi, tipologia di vela ed altro), e a motore (caratterizzate dalla potenza, numero e tipologia dei motori). Inoltre, l'amministratore può anche avviare un'asta per una imbarcazione, indicando una data di termine dell'asta, e una base di partenza (in $). Gli utenti, previa autenticazione, possono visualizzare le aste in corso e, dopo aver selezionato l'asta di interesse, inviare un'unica offerta indicando la cifra proposta. Allo scadere dell'asta, l'amministratore visualizza l'offerta che si è aggiudicata l'imbarcazione e i dati del compratore.

Si richiede di definire un Class Diagram concettuale con tutte le classi necessarie a modellare la funzionalità di inserimento di un'offerta per un'asta.

Successivamente si richiede di disegnare Class Diagram e Sequence Diagram si 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 scriva tutto il codice Java che è possibile desumere dal seguente sequence diagram.



Esercizio 3

```
1  public class Exercise {
2      public static void main(String args[]) {
3          String s1 = new String("CIAO");
4          String s2 = new String("CIAO");
5          try {
6              System.out.println("Confronto.");
7              if (s1 == s2) {
8                  throw new Exception("Non devono essere uguali!");
9              }
10             System.out.println("Confronto effettuato, tutto OK.");
11         } catch (Exception e) {
12             System.out.println("Le stringhe erano uguali.");
13         }
14         System.out.println("Fatto.");
15
16     }
17 }
```

Si consideri il frammento di codice Java mostrato sopra.

1. Il codice compila correttamente? Se no, indicare una possibile correzione del codice affinché compili.
2. Qual è l'output di un'invocazione del metodo `main`?

Esercizio 4

Descrivere i concetti di overloading ed overriding, fornendo almeno esempio di ciascuno dei due concetti nel dominio della gestione di un museo. Relativamente agli esempi, si richiede esplicitamente di fornire un class diagram e la relativa implementazione in linguaggio Java, e di indicare in maniera chiara i punti dove sussistono overloading/overriding.

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).

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.

Esame di Object Orientation (Gr. 1) – Gennaio 2023

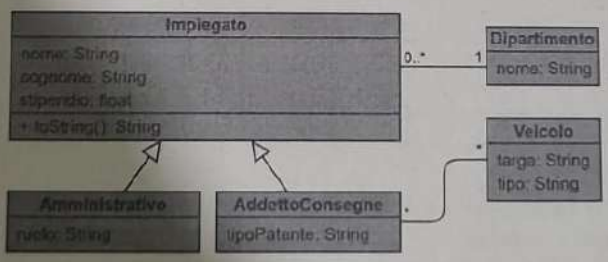
- *Scrivere immediatamente su ogni foglio che vi è stato consegnato Cognome, Nome, N° Matricola.*
- *Non è consentito consultare appunti, libri, o colleghi, né qualunque dispositivo elettronico, PENA IMMEDIATO ANNULLAMENTO DELLA PROVA*
- *Tempo a disposizione: 3 ore*

Esercizio 1

Vi è commissionata la realizzazione di un sistema per gestire i tirocini curricolari (sia interni che esterni) di studenti del Corso di Laurea in Informatica. Il sistema permette agli studenti di inserire una richiesta di tirocinio, indicando la tipologia di tirocinio (interno o esterno), una breve descrizione testuale della tematica trattata, una data prevista di inizio e di fine. Per tirocini interni, inoltre, lo studente deve indicare un docente di riferimento tra i Professori del Corso di Laurea in Informatica. Facoltativamente, lo studente può indicare anche al più due altri co-tutor, da selezionare tra i Professori e/o i Ricercatori del Corso di Laurea in Informatica. Per i tirocini esterni, invece, lo studente deve selezionare un'azienda tra quelle convenzionate con l'Università, e inserire nome, cognome ed e-mail del referente aziendale che seguirà il tirocinio. I membri della commissione tirocini del Corso di Laurea possono visualizzare le domande inviate dagli studenti e decidere se approvarle o rigettarle.

Si richiede di definire (a) Class Diagram e (b) Sequence Diagram per modellare la funzionalità di inserimento di una **richiesta di tirocinio**, eventualmente aiutandosi con delle rappresentazioni grafiche dell'interfaccia utente (mock-up). Il Class Diagram e il Sequence Diagram, intesi come modello di dominio, devono rifarsi all'euristica Entity-Boundary-Control (EBC).

Esercizio 2



Si scriva tutto il codice Java che è possibile desumere dal class diagram riportato sopra.

Esercizio 3

```
public class Main {
    public static void main(String[] args){
        Studente[] studenti = new Studente[5];
        studenti[0] = new Studente("Mario Rossi", "N86/1234");
        studenti[1] = new StudenteLavoratore("Sam Porter Bridges", "N86/5678", "Bridges");
        Corso c = new Corso(studenti);
        try {
            for(int i=0; i<=5 ; i++)
                c.stampaStudente(i);
        } catch(ArrayIndexOutOfBoundsException e) {
            System.out.println("404: Studente non trovato");
        } catch(Exception e){
            System.out.println("Woops");
        }
    }
}

public class Corso {
    Studente[] studenti;
    public Corso(Studente[] studenti){
        this.studenti = studenti;
    }

    public void stampaStudente(int i) {
        System.out.println(i+": "+this.studenti[i].nome);
    }
}

public class Studente {
    String nome; String matricola;
    public Studente(String nome, String matricola){ this.nome=nome; this.matricola=matricola;}
}

public class StudenteLavoratore extends Studente {
    String azienda;
    public StudenteLavoratore(String nome, String matricola, String azienda){
        this.nome=nome; this.matricola=matricola; this.azienda=azienda;
    }
}
```

Si considerino le classi Java riportate sopra (si assuma che ogni classe sia definita in un file apposito).

a) Il codice compila correttamente? Motivare la risposta e fornire le possibili correzioni in caso di risposta negativa.
b) Dopo aver apportato le eventuali correzioni di cui al punto (a), 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

Si spieghi il meccanismo che permette, in Java Swing, di specificare una o più porzioni di codice da eseguire ogni volta che un utente clicca su un pulsante in un'interfaccia grafica.

Esame di Object Orientation (Gr. 2) – 20 marzo 2023

- *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*

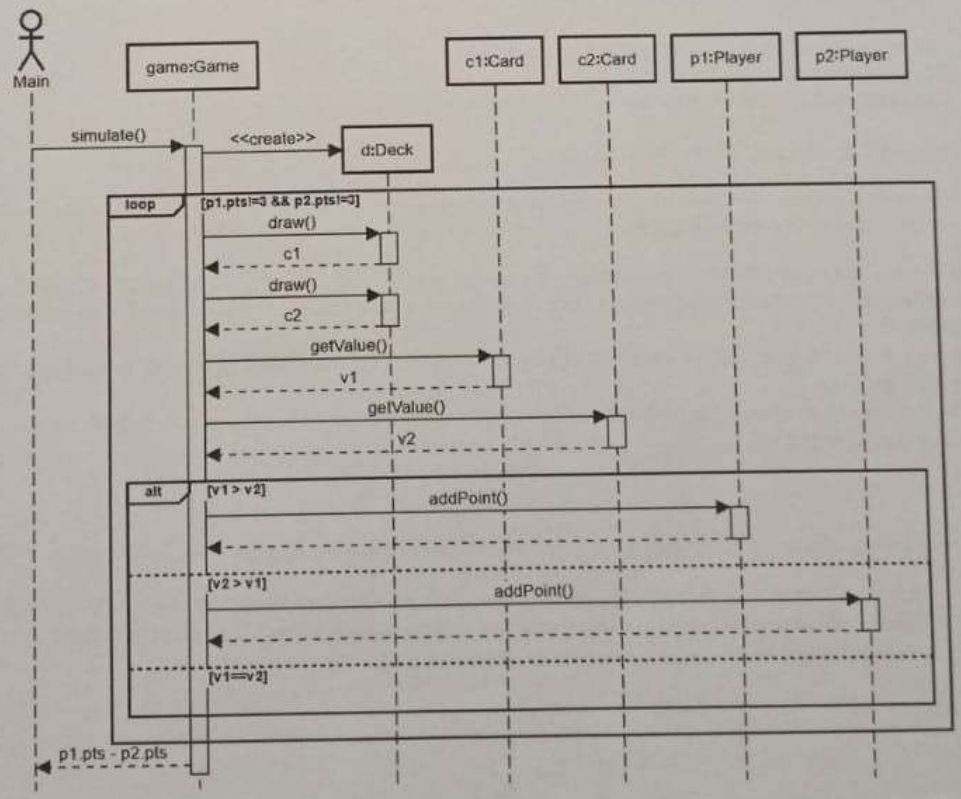
Esercizio 1

Vi viene commissionata la realizzazione di un software per la gestione di Scenari di test di sistemi elettronici (e.g.: decoder satellitari, registratori di cassa, etc.). Uno Scenario di test è caratterizzato da un nome univoco e da una sequenza di Passi (o *Step*) da eseguire nell'ordine dato. Ciascun passo può essere di due tipi: (1) *azione* oppure (2) *asserzione*. Le azioni sono caratterizzate da una descrizione testuale dell'azione da effettuare (e.g.: "*premi il pulsante X*"), mentre le asserzioni sono caratterizzate da una descrizione in testo libero della condizione da verificare (e.g.: "*il led Y deve essere accesso*") e da un livello di severità che può assumere valori in {BASSO, MEDIO, ALTO}. Il sistema permette a un *Test Engineer* di creare (e successivamente modificare) Scenari di test. Per creare uno Scenario di test, il *Test Engineer* seleziona innanzitutto il dispositivo elettronico cui il test è riferito tra quelli presenti nel sistema. Successivamente, l'*Engineer* può specificare una sequenza arbitrariamente lunga di Step. Si sottolinea che è necessario che il sistema tenga traccia dell'Engineer che ha creato un particolare scenario di test.

Si richiede di definire (a) Class Diagram e (b) Sequence Diagram per modellare la funzionalità di **inserimento di uno scenario di test**, eventualmente aiutandosi con delle rappresentazioni grafiche dell'interfaccia utente (mock-up). Il Class Diagram e il Sequence Diagram, intesi come modello di dominio, devono rifarsi all'euristica Entity-Boundary-Control (EBC).

Esercizio 2

Si scriva tutto il codice Java che è possibile desumere dal seguente sequence diagram.



Esercizio 3

```
public class Evaluator {
    List<String> answers;
    public Evaluator(List<String> a ){
        this.answers=a;
    }

    public String evaluate() {
        double vote = 0;
        for(String e : answers){
            vote+= e.length()*6;
        }
        vote=vote/4;
        if(vote<18){
            System.out.println("VOTO MINORE DI 18!");
            throw new Exception("Esame non superato, voto:"+vote);
        }
        return "Esame superato con valutazione: "+vote;
    }
}

public static void main(String args[]) {
        List<String> a = new List<>();
        a.add("");
        a.add("abcd");
        a.add("abcde");
        a.add("ab");
        Evaluator evaluator = new Evaluator(a);
        System.out.println("VALUTAZIONE IN CORSO...");
        try {
            System.out.println(evaluator.evaluate());
        } catch (Exception e) {
            System.out.println("Bocciato!"+e.getMessage());
        }
        System.out.println("FINE VALUTAZIONE");
}
```

Si considerino le classi java riportate sopra (si assuma che ogni classe sia definita in un file apposito).

a) Il codice compila correttamente? Motivare la risposta e fornire possibili correzioni in caso di risposta negativa.
b) Al netto delle eventuali correzioni apportate al punto precedente, qual è l'output di una esecuzione 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

Spiegare la differenza tra associazioni, aggregazioni e composizioni nel contesto di class diagram di dettaglio UML.

Scrivere almeno un esempio per ognuna di tali relazioni, sia in termini di disegno del diagramma che con un opportuno spezzone di codice di esempio. Gli esempi devono riguardare concetti relativi ai componenti di un personal computer (ad esempio processore, scheda madre, memoria, mouse wireless e corrispondente ricevitore bluetooth).

Esame di Object Orientation (Gr. 2) – 16 giugno 2023

- *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 software per la gestione di polisportive, ogni struttura polisportiva è caratterizzata da un nome, un indirizzo, un numero di telefono ed un proprietario, caratterizzato dalle proprie generalità (nome, cognome, data di nascita, codice fiscale, email). Le strutture possono prevedere campi per diversi sport: calcio, tennis, basket e pallavolo. A loro volta i campi possono distinguersi in base al numero di giocatori per squadra ad esempio: i campi da calcio possono essere fatti per squadre da 5 ,8 o 11. Il proprietario può inserire nel sistema nuovi campi sportivi, identificati da nome, sport, tipologia, e prezzo. Gli utenti del sistema per registrarsi devono specificare nome utente, password, indirizzo e-mail e carta di credito. Gli utenti registrati possono richiedere una lista dei campi disponibili lo sport da praticare, la data e l'orario per la prenotazione (valida per una singola ora di gioco). Se non ci sono campi disponibili il sistema chiede all'utente di specificare data e orario differenti; altrimenti, l'utente seleziona un campo disponibile e la spesa viene detratta dalla carta di credito dell'utente.

Si richiede di definire (a) Class Diagram e (b) Sequence Diagram per modellare la funzionalità di **richiesta di una nuova prenotazione (comprende la visione dei campi disponibili)**. Il Class Diagram e il Sequence Diagram, intesi come modello di dominio, devono rifarsi all'euristica Entity-Boundary-Control (EBC).

Esercizio 2

Rappresentare con un Sequence Diagram un'invocazione del metodo *getGoldCoins* della classe Gold riportato sotto.

```
public Coin[] getGoldCoins(Configuration conf){
    GoldVendingMachine g = new GoldVendingMachine();
    float gold_actual_price = g.get_gold_price(conf.gold_coins_type);
    float total_amount = gold_actual_price * conf.get_num_coins_required();
    int transaction_successful = 0;
    if(conf.get_payment_mode() == "card"){
        if(Bank.authorize_card(conf.get_iban())){
            Bank.charge_money(conf.get_iban(),total_amount);
            transaction_successful = 1;
        }else{
            throw new RuntimeException ("card not accepted!");
        }
    }else if(conf.get_payment_mode() == "cash") {
        g.open_money_desk_and_wait();
        if(g.money_inserted()){
            transaction_successful = 1;
        }
        throw  new RuntimeException ("money not inserted!");
    }
    if(transaction_successful == 1){
        return new Coin[conf.get_num_coins_required()];
    }
    return null;
}
```

Esercizio 3

```
public static void main(String[] args){
    try {
        Animal animal = new Dog();
        animal.bark();
        for (int i = 0; i <= 5; i++) {
            animal.eat();
            animal.increase_dog_weight();
        }
    }
    catch (Exception exception ){
        System.out.println("OH NO your animal is died, "+exception.getMessage());
    }
    catch (RuntimeException runtimeException){
        System.out.println(runtimeException.getMessage());
    }
}

public class Dog extends Animal{
    public int weight = 15;
    public void bark(){
        System.out.println("your dog is barking");
    }
    public void increase_dog_weight() {
        weight++;
        if(this.weight >=20){
            throw new RuntimeException("your dog is too fat!");
        }
    }
    public void eat(){
        System.out.println("your dog is eating");
    }
}

public class Animal {
    public void eat(){
        System.out.println("your animal is eating");
    }
}
```

Si considerino le classi java riportate sopra (si assuma che ogni classe sia definita in un file apposito).

a) Il codice compila correttamente? Motivare la risposta e fornire possibili correzioni in caso di risposta negativa.
b) Al netto delle eventuali correzioni apportate al punto precedente, qual è l'output di una esecuzione 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

Qual' è la differenza tra *extends* e *implements*? riportare un esempio di ciascuno nel dominio della polisportiva presentata nell'esercizio 1 (eventualmente introducendo nuove classi ulteriori, rispetto al problema proposto).