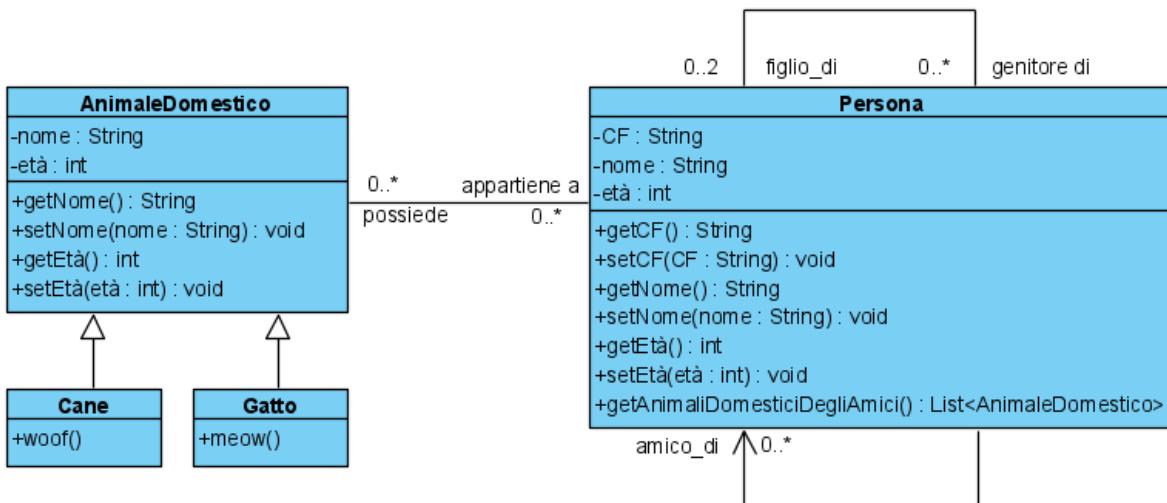


Esercizi di Object Orientation

Esercizio 1

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



Esercizio 2

Si realizzi un class diagram per il seguente frammento di codice.

```
import java.util.Date;
import java.util.List;

public class Post {
    String titolo;
    String contenuto;
    Date data;

    Utente autore;
    List<Commento> commenti;
    List<Reazione> reazioni;

    /* Costruttori, getter e setter omessi per brevità */

    void stampaCommentatori() {
        for(Commento c : commenti) {
            System.out.println(c.getAutore().toString());
        }
    }
}
```

ESERCIZIO 1

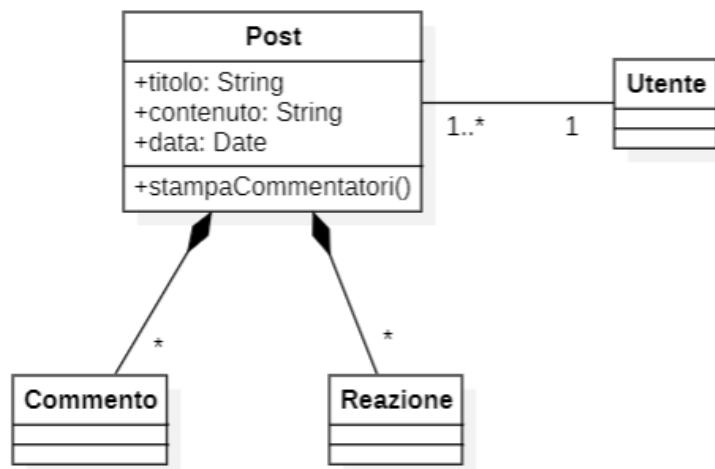
```
1 public class Persona {  
2     String CF;  
3     String nome;  
4     int eta;  
5  
6     ArrayList<AnimaleDomestico> listaAnimaliDomestici = new ArrayList<>();  
7     ArrayList<Persona> listaAmici = new ArrayList<>();  
8     ArrayList<Persona> listaFigli = new ArrayList<>();  
9     ArrayList<Persona> listaGenitori = new ArrayList<>();  
10  
11    public Persona(String CF, String nome, int eta) {  
12        this.CF = CF;  
13        this.nome = nome;  
14        this.eta = eta;  
15    }  
16  
17    public void addAmici(Persona amico) {  
18        this.listaAmici.add(amico);  
19    }  
20  
21    public ArrayList<Persona> getAmici(){  
22        return this.listaAmici;  
23    }  
24  
25    public void addFiglio(Persona figlio) {  
26        this.listaFigli.add(figlio);  
27    }  
28  
29    public ArrayList<Persona> getFigli() {  
30        return this.listaFigli;  
31    }  
32  
33    public void addGenitore(Persona genitore) {  
34        int dimensione = this.listaGenitori.size();  
35        if (dimensione ≥ 2) {  
36            System.out.println("errore");  
37        } else {  
38            this.listaGenitori.add(genitore);  
39        }  
40    }  
41  
42    public ArrayList<Persona> getGenitori() {  
43        return this.listaGenitori;  
44    }  
45  
46    public String getCF() {  
47        return this.CF;  
48    }  
49  
50    public void setCF(String cf) {  
51        this.CF = cf;  
52    }  
53  
54    public String getNome() {  
55        return this.nome;  
56    }  
57  
58    public void setNome(String n) {  
59        this.nome = n;  
60    }  
61  
62    public int getEta() {  
63        return this.eta;  
64    }  
65  
66    public void setEta(int e) {  
67        this.eta = e;  
68    }  
69  
70    public void addAnimaleDomestico(AnimaleDomestico ad) {  
71        listaAnimaliDomestici.add(ad);  
72    }  
73}
```

```
1 public class Cane {  
2     public Cane(String nome, int eta) {  
3         super(nome, eta);  
4     }  
5  
6     public void woof() {  
7         System.out.println("woooof!");  
8     }  
9 }
```

```
1 public class Gatto {  
2     public Gatto(String nome, int eta) {  
3         super(nome, eta);  
4     }  
5  
6     public void meow() {  
7         System.out.println("meow!");  
8     }  
9 }
```

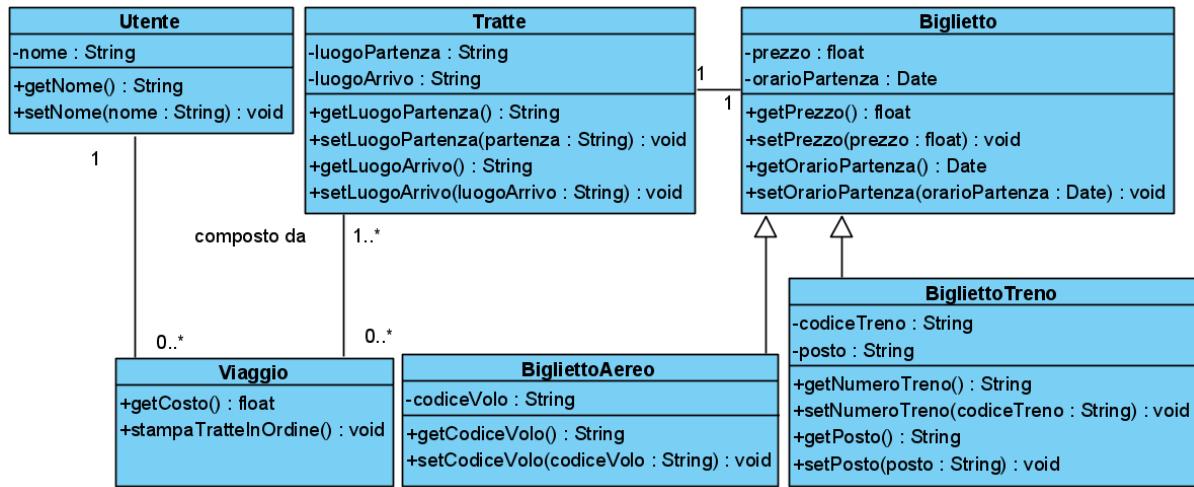
```
1 public class AnimaleDomestico {  
2     String nome;  
3     int eta;  
4  
5     ArrayList<Persona> listaPersone = new ArrayList<>();  
6  
7     public AnimaleDomestico(String nome, int eta) {  
8         this.nome = nome;  
9         this.eta = eta;  
10    }  
11  
12    public String getNome() {  
13        return this.nome;  
14    }  
15  
16    public void setNome(String n) {  
17        this.nome = n;  
18    }  
19  
20    public int getEta() {  
21        return this.eta;  
22    }  
23  
24    public void setEta(int e) {  
25        this.eta = e;  
26    }  
27  
28    public void addPersona(Persona p) {  
29        listaPersone.add(p);  
30    }  
31 }
```

ESERCIZIO 2



Esercizio 3

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



Esercizio 4

Si realizzi un class diagram per il seguente frammento di codice.

```
import java.util.List;

public class Automobile {
    String targa;
    String modello;
    Persona conducente;
    Persona[] passeggeri = new Persona[4];
    List<Revisione> revisioniEffettuate;

    /* Costruttore e getter e setter omessi */

    void stampaPasseggeri() {
        for(int i=0;i<passeggeri.length;i++) {
            if(passeggeri[i]!=null) {
                passeggeri[i].printName();
            }
        }
    }
}
```

ESERCIZIO 3

```
1 import java.util.ArrayList;
2
3 public class Utente {
4     String nome;
5
6     ArrayList<Viaggio> listaViaggi = new ArrayList<>();
7
8     public Utente(String nome) {
9         this.nome = nome;
10    }
11
12    public void addViaggio (Viaggio v) {
13        listaViaggi.add(v);
14    }
15
16    public String getNome() {
17        return this.nome;
18    }
19
20    public void setNome(String n) {
21        this.nome = n;
22    }
23}
24}
```

```
1 import java.util.ArrayList;
2
3 public class Viaggio {
4     Utente utente;
5     float costo;
6
7     ArrayList<Tratte> listaTratte = new ArrayList<>();
8
9     public Viaggio(String nome, float costo, Tratte tratta) {
10        utente = new Utente(nome);
11        this.costo = costo;
12        listaTratte.add(tratta);
13    }
14
15    public float getCosto() {
16        return this.costo;
17    }
18
19    public void addTratta(Tratte t) {
20        listaTratte.add(t);
21    }
22
23    public void stampaTratteInOrdine(ArrayList<Tratte> listaTratte) {
24        for (Tratte tratte : listaTratte) {
25            System.out.println(tratte);
26        }
27    }
28
29}
```

```
1 import java.util.ArrayList;
2
3 public class Tratte {
4     String luogoPartenza;
5     String luogoArrivo;
6
7     ArrayList<Viaggio> listaViaggi = new ArrayList<>();
8     Biglietto biglietto;
9
10    public Tratte(String luogoPartenza, String luogoArrivo, Biglietto biglietto) {
11        this.luogoPartenza = luogoPartenza;
12        this.luogoArrivo = luogoArrivo;
13        this.biglietto = biglietto;
14    }
15
16    public String getLuogoPartenza() {
17        return this.luogoPartenza;
18    }
19
20    public String getLuogoArrivo() {
21        return this.luogoArrivo;
22    }
23
24    public void setLuogoPartenza(String partenza) {
25        this.luogoPartenza = partenza;
26    }
27
28    public void setLuogoArrivo(String luogoArrivo) {
29        this.luogoArrivo = luogoArrivo;
30    }
31
32    public void addViaggio(Viaggio v) {
33        listaViaggi.add(v);
34    }
35}
```

```
1 import java.util.Date;
2
3 public class Biglietto {
4     float prezzo;
5     Date orarioPartenza;
6
7     Tratte tratta;
8
9     public Biglietto(float prezzo, Date orarioPartenza, Tratte tratta) {
10        this.prezzo = prezzo;
11        this.orarioPartenza = orarioPartenza;
12        this.tratta = tratta;
13    }
14
15    public float getPrezzo() {
16        return this.prezzo;
17    }
18
19    public Date getOrarioPartenza() {
20        return this.orarioPartenza;
21    }
22
23    public void setPrezzo (float prezzo) {
24        this.prezzo = prezzo;
25    }
26
27    public void setOrarioPartenza (Date orario) {
28        this.orarioPartenza = orario;
29    }
30
31 }
32
```

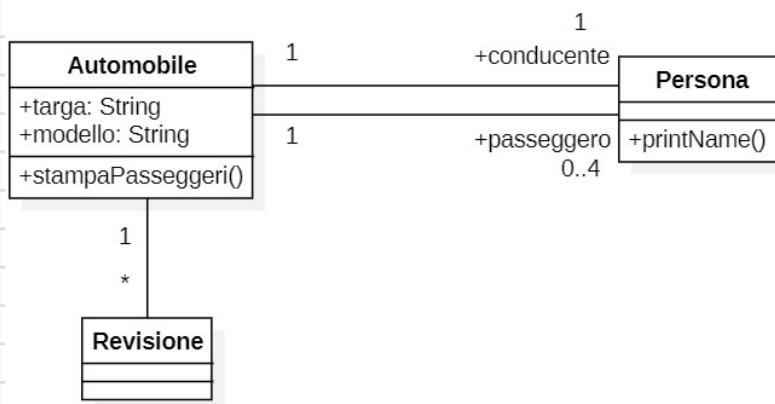
```
1 import java.util.Date;
2
3 public class BigliettoAereo extends Biglietto{
4     String codiceVolo;
5
6     public BigliettoAereo(String codiceVolo, Float prezzo, Date orarioPartenza, Tratte tratta) {
7         super(prezzo,orarioPartenza, tratta);
8         this.codiceVolo = codiceVolo;
9     }
10
11    public String getCodiceVolo() {
12        return this.codiceVolo;
13    }
14
15    public void setCodiceVolo(String cv) {
16        this.codiceVolo = cv;
17    }
18 }
```

```

1 import java.util.Date;
2
3 public class BigliettoTreno extends Biglietto{
4     String codiceTreno;
5     String posto;
6
7     public BigliettoTreno(String codiceTreno, String posto, Float prezzo, Date orarioPartenza, Tratte tratta) {
8         super(prezzo, orarioPartenza, tratta);
9         this.codiceTreno = codiceTreno;
10        this.posto = posto;
11    }
12
13    public String getCodiceTreno() {
14        return this.codiceTreno;
15    }
16
17    public String getPosto() {
18        return this.posto;
19    }
20
21    public void setCodiceTreno(String ct) {
22        this.codiceTreno = ct;
23    }
24
25    public void setPosto(String p) {
26        this.posto = p;
27    }
28 }
29

```

ESERCIZIO 4



Esercizio 5

```
class EccezioneA extends Exception {}
class EccezioneB extends EccezioneA {}
class EccezioneC extends EccezioneB {}

public class Test {
    public void foo() {
        try{
            throw new EccezioneB();
        }
        catch(EccezioneC e) {
            System.out.println("Catch C");
        }
        catch(EccezioneA e) {
            System.out.println("Catch A");
        }
        catch(Exception e) {
            System.out.println("Catch Exception");
        }
    }
}
```

Si consideri il codice Java in figura.

- Il codice compila correttamente? Se no, indicare come sarebbe possibile modificare il codice affinché la compilazione vada a buon fine. **Compila**
- Qual è l'output su stdout di un'invocazione del metodo foo() nella classe Test?
Output: Catch

Esercizio 6

```
class MiaEccezione extends Exception {}

public class Main {

    public void test() {
        try {
            System.out.println("Hello");
            randomMethod();
            System.out.println("World");
        } catch(MiaEccezione e) {
            System.out.println("MiaEccezione");
        }
    }

    public void randomMethod() throws Exception {
        //Math.random() ritorna un double pseudo-casuale tra 0.0 e 1.0
        if(Math.random() > 0.5) {
            throw new MiaEccezione();
        } else {
            throw new Exception();
        }
    }
}
```

Si consideri il codice riportato in figura.

- Il codice compila correttamente? Se no, indicare come sarebbe possibile modificare il codice affinché la compilazione vada a buon fine.
- Qual è l'output su stdout di un'invocazione del metodo `test()` nella classe `Main`? Nel caso siano possibili più output, indicarli tutti.

Non compila perché il metodo `RandomMethod()` manca della possibilità di lanciare un'eccezione.
Inoltre bisogna modificare il `catch` per catturare `Exception` e non `Mia Eccezione`.

Output:
Hello
Mia Eccezione

Esercizio 7

```
class EccezioneA extends Exception {
    String testo;
    public EccezioneA(String testo) {
        this.testo = testo;
    }
}

class EccezioneB extends EccezioneA {
    int n;
    public EccezioneB(int n) { → String testo
        this.n=n; ↘ Super(testo)
    }
}

public class Test {
    EccezioneA a;
    EccezioneB b;
    public void test() throws Exception {
        try { → " parametro"
            b = new EccezioneB(123);
            a = new EccezioneA("Ooops");
            throw a;
        }catch(EccezioneB b) {
            throw a;
        }
        catch(EccezioneA a) {
            throw b;
        }
    }
}
```

Si consideri il codice riportato in figura.

- Il codice compila correttamente? Se no, indicare come sarebbe possibile modificare il codice affinché la compilazione vada a buon fine.
- Qual è l'output su `stdout` di un'invocazione del metodo `test()` nella classe `Test`? Nel caso siano possibili più output, indicarli tutti.