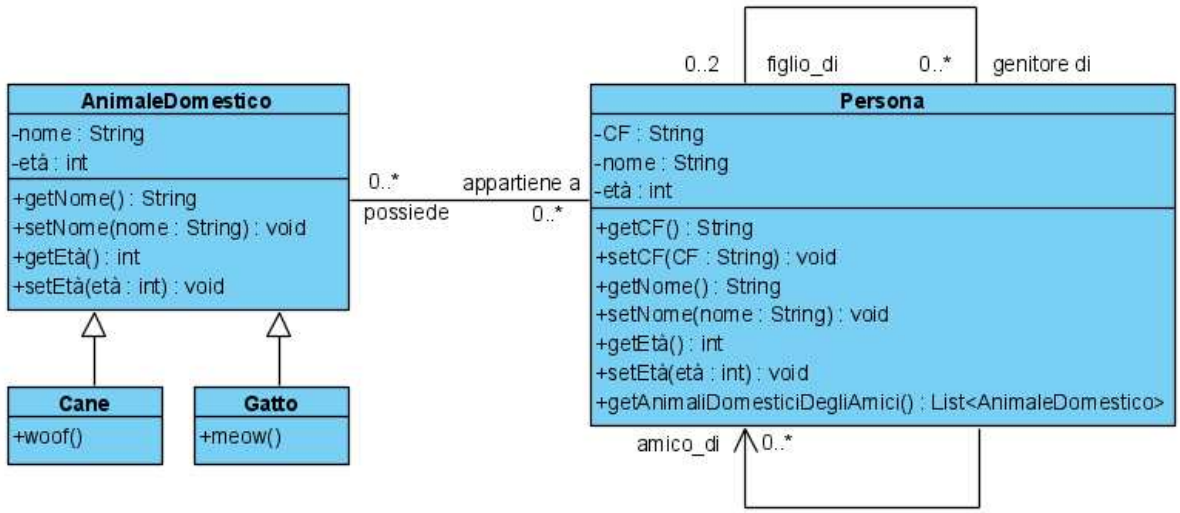
Esercizi di Object Orientation

Esercizio 1

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



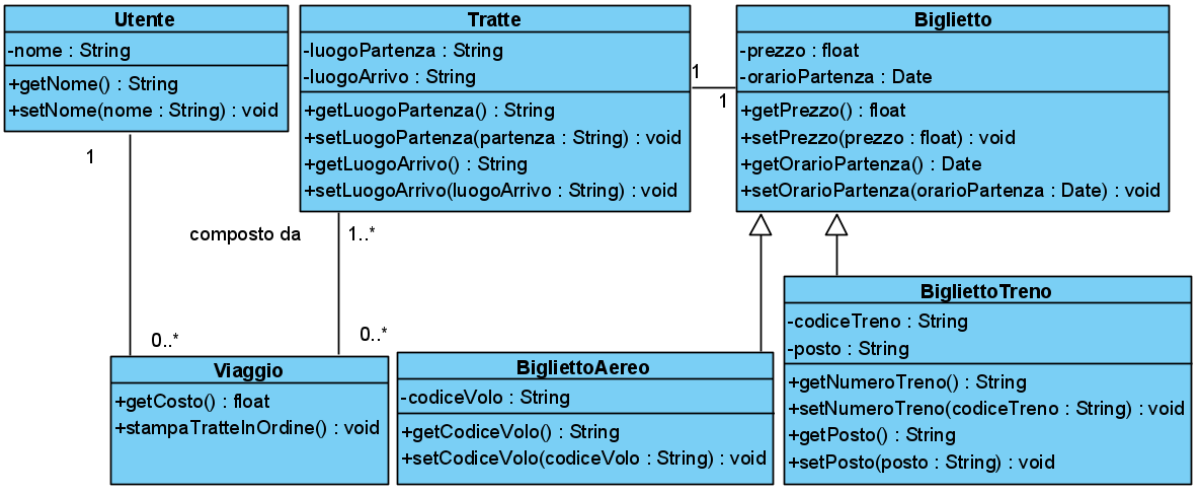
Esercizio 2

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 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 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.
- Qual è l'output su `stdout` di un'invocazione del metodo `foo()` nella classe `Test`?

Esercizio 6

```java
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() {
		//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.

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) {
		this.n=n;
	}
}

public class Test {
	EccezioneA a;
	EccezioneB b;
	public void test() throws Exception {
		try {
			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.

Esercizi su Sequence Diagram

Esercizio 1

Si realizzi un Sequence Diagram relativo a una invocazione del metodo start() della classe Robot riportata di seguito. Si assuma che tutti gli import necessari siano correttamente definiti.

```
public class Robot {
     Sensor s = new Sensor();
     Weapon w = new LaserTurret();
     void start() {
          while(!s.missionAccomplished()){
               if(s.enemyDetected()){
                    Enemy e = s.getEnemyDetails();
                    w.attack(e);
               } else {
                    moveRandomly();
               }
          }
     }
     /* Eventuali altri metodi omessi per brevità */
}
```

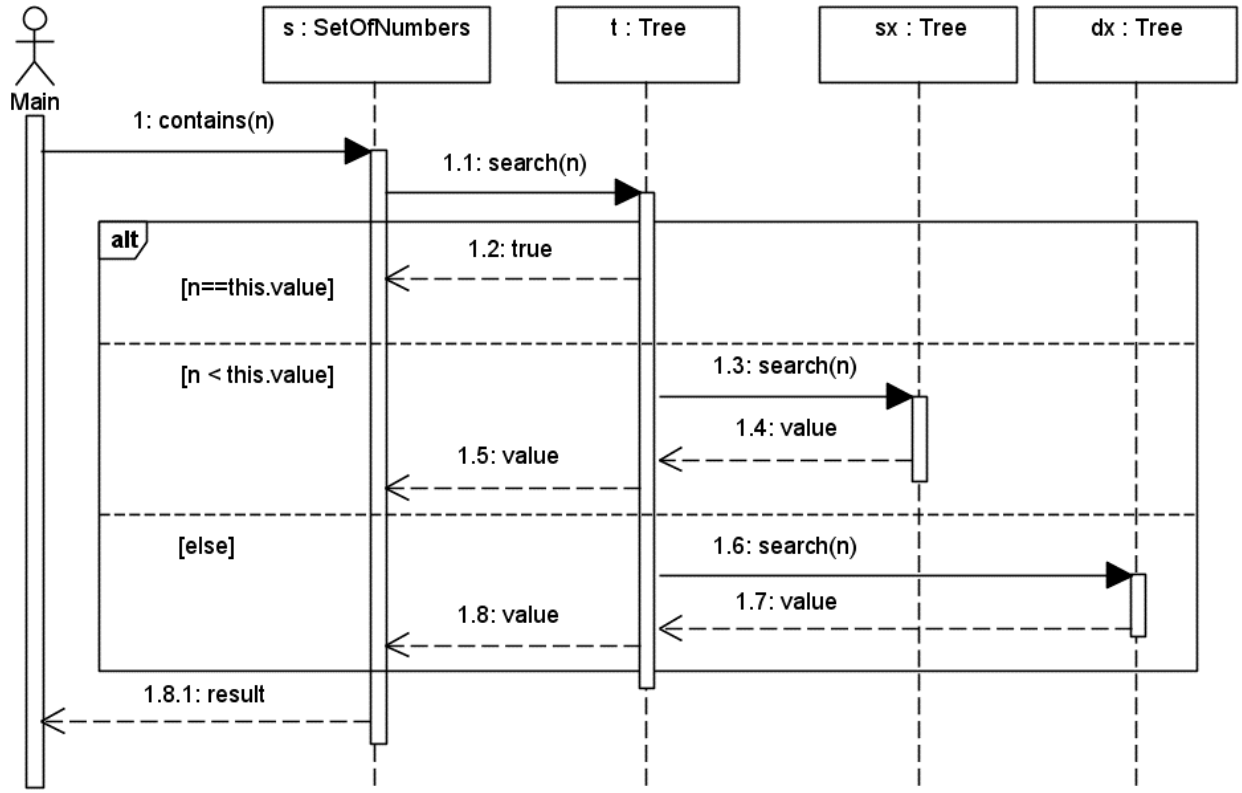
Esercizio 2

Si realizzi un Sequence Diagram relativo a una invocazione del metodo infect() della classe Virus riportata di seguito. Si assuma che tutti gli import necessari siano correttamente definiti

```
public class Virus {
     void infect() {
          NetworkScanner n = new NetworkScanner();
          List<Host> targets = n.getTargets();
          for(Host h : targets) {
               if(h.isWindows()) {
                    useCveExploit(h);
               }
               else if(h.isLinux()) {
                    kernelAttack(h);
               }
               else {
                    defaultAttack(h);
               }
          }
     }

     /* Eventuali altri metodi omessi per brevità */
}
```

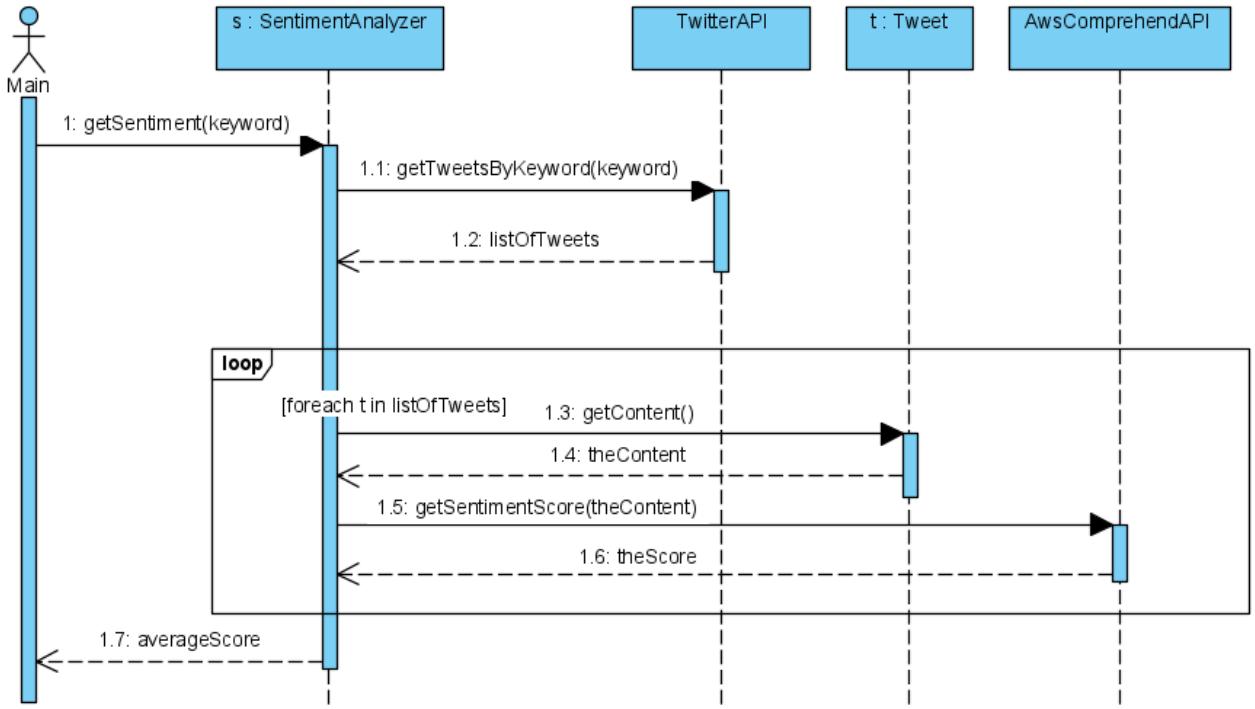
Esercizio 3

Si scriva tutto il codice Java che è possibile desumere dal seguente Sequence Diagram. Si assuma che la variabile n sia di tipo intero.



Esercizio 4

Si scriva tutto il codice Java che è possibile desumere dal seguente Sequence Diagram. Si assuma che la variabile keyword sia di tipo String.



```java
public class MusicalInstrument {
	public void play() {
		System.out.print("zzzzz");
	}
}

public class Drum extends MusicalInstrument {
	public void play () {
		System.out.print("ba-dum-bum-CHING");
	}
}

public class Guitar {
	public void solo() {
		System.out.print("Na nana na nanaaaa naaaaaaa");
	}
}

public class Band {
	public static void main(String[] args) {
		MusicalInstrument one = new Drum();
		MusicalInstrument two = new Guitar();
		two.play();
		one.play();
	}
}
```

Si consideri il codice Java sopra riportato.

- Il codice compila correttamente? Se no, indicare come modificare il codice affinché la compilazione abbia successo.
- Che modifica apportereste per introdurre la possibilità, per ogni strumento, di esibirsi in un assolo?
- 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.

Si realizzi un sequence diagram che rappresenti un'invocazione del metodo `cook()` della classe `RoboCook`.

```java
public class RoboCook {
	private IOComponent io;

	public void cook(Recipe r) {
		io.beep();
		List<Step> steps = r.getSteps();
		for(Step s : steps) {
			execute(s);
		}
		io.beep();
		String name = r.getName();
		io.sendSMS(name); //sends sms notification to the owner
	}

	private void execute(Step s) { /* execute step */ }
}
```

```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.