# Ingineria Sistemelor de Programare

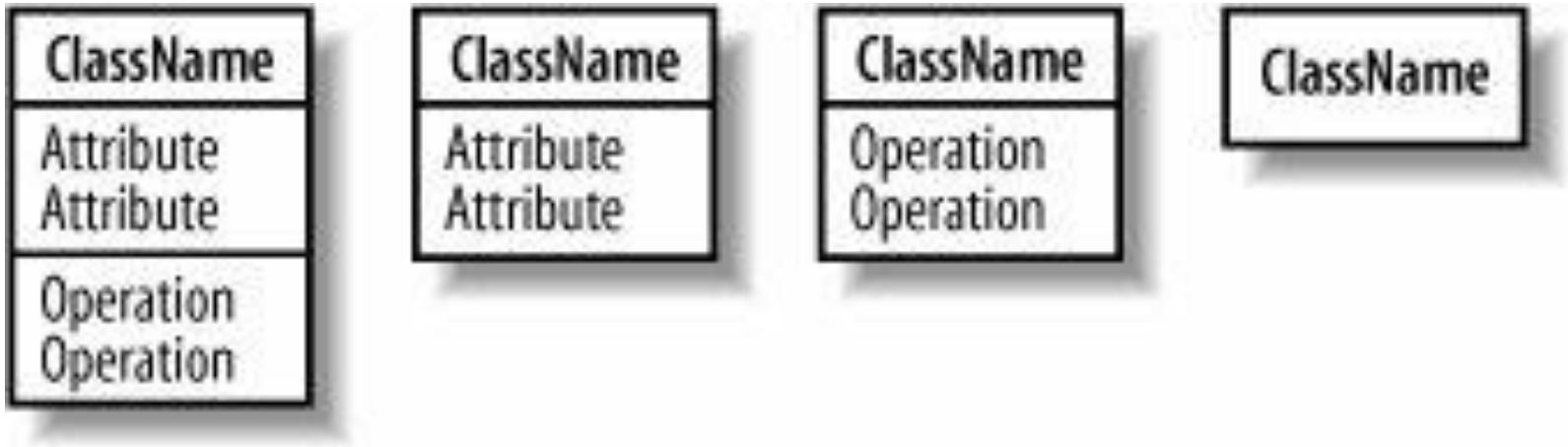
UML – Class Diagram

mihai.hulea@aut.utcluj.ro

2016

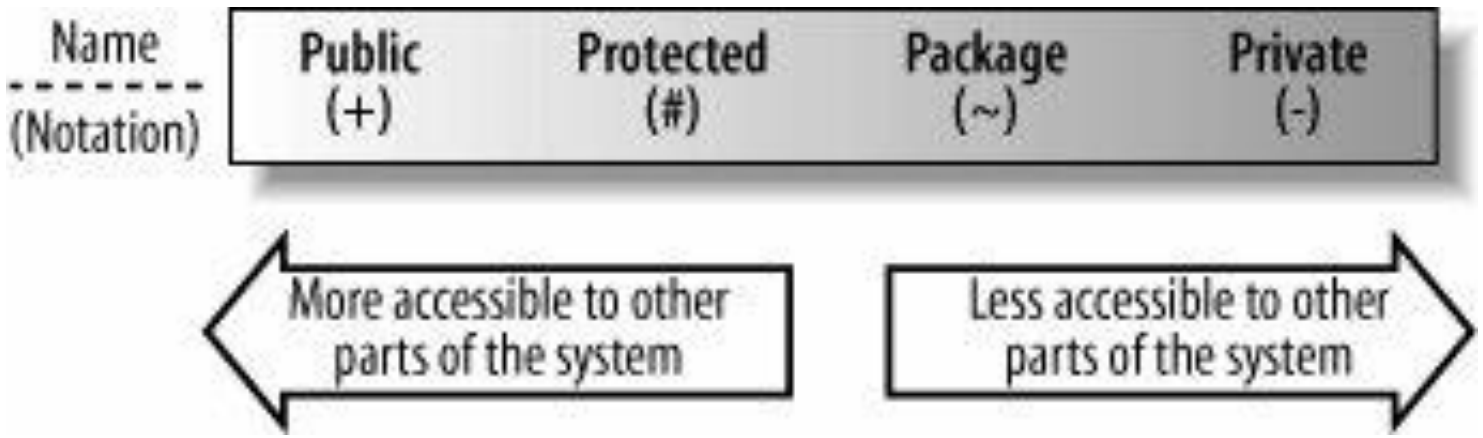
# Introducere

# Reprezentarea claselor



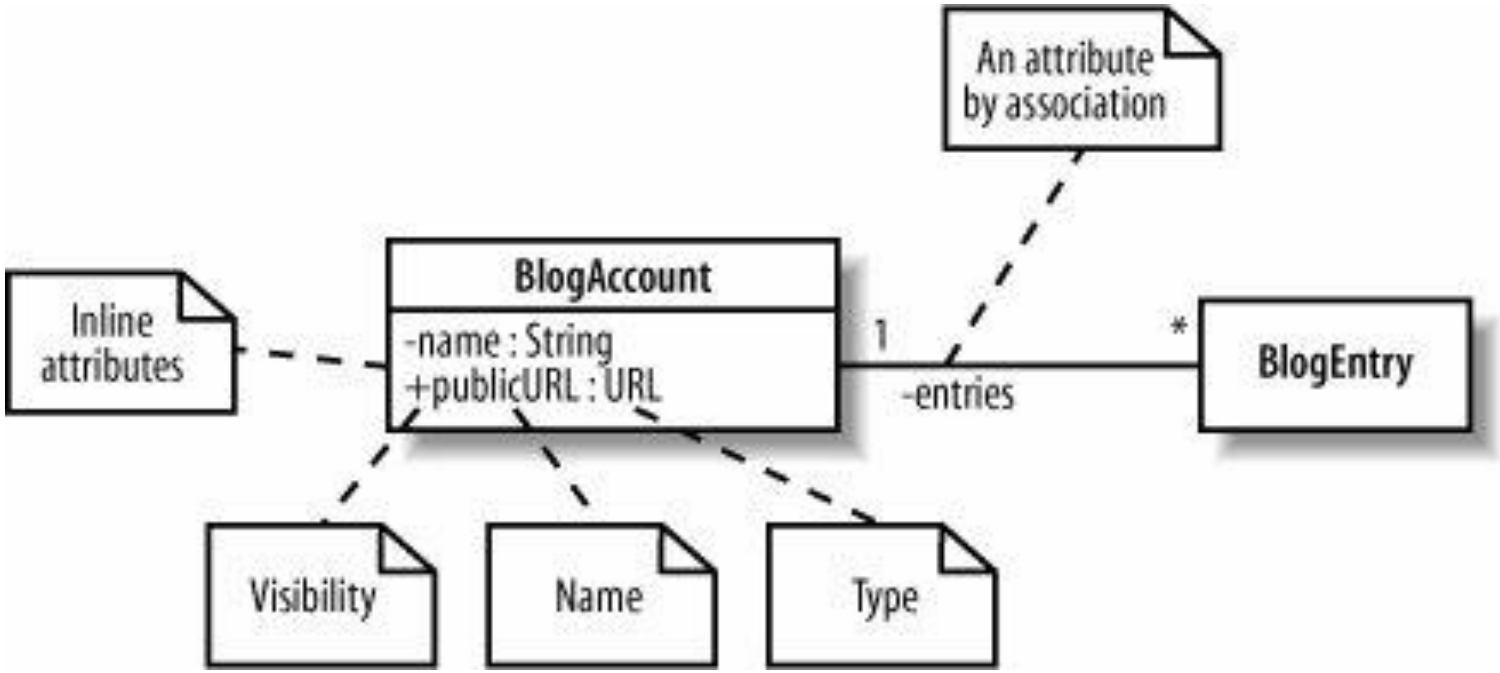
# Gradul de vizibilitate

- Public
- Protected
- Package
- Private



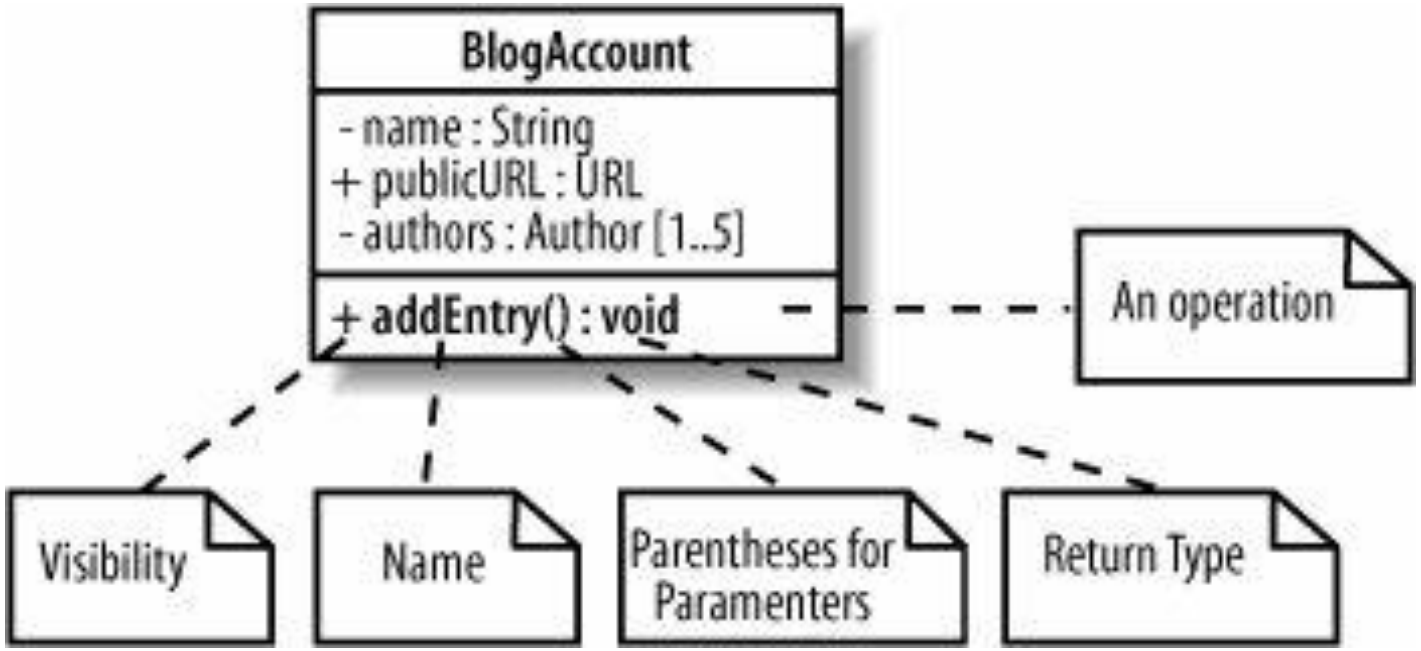
# Atribute

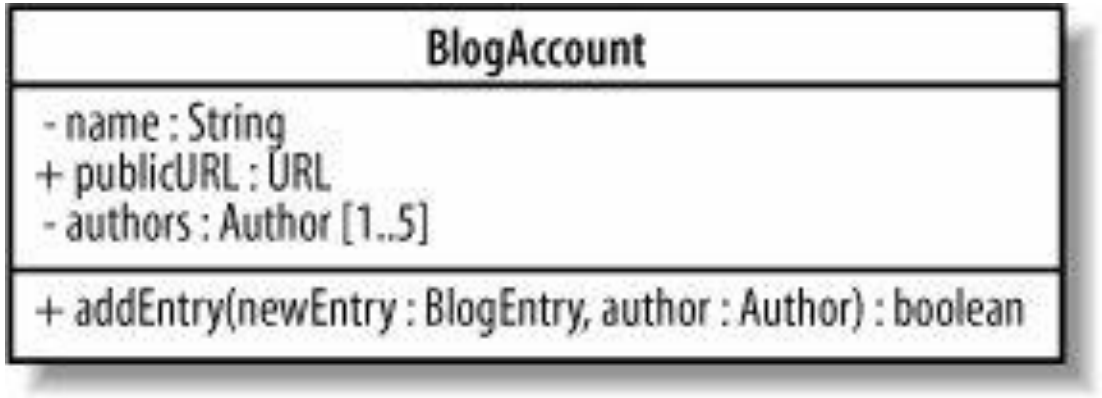
- Atribute de tip “inline” - definite in interorul clasei
- Atribute de asociere – definite pe linia de asociere dintre clase



# Operatii

- Structura operatie:
  - Grad de vizibilitate
  - Nume
  - Lista parametric
  - Tip de return



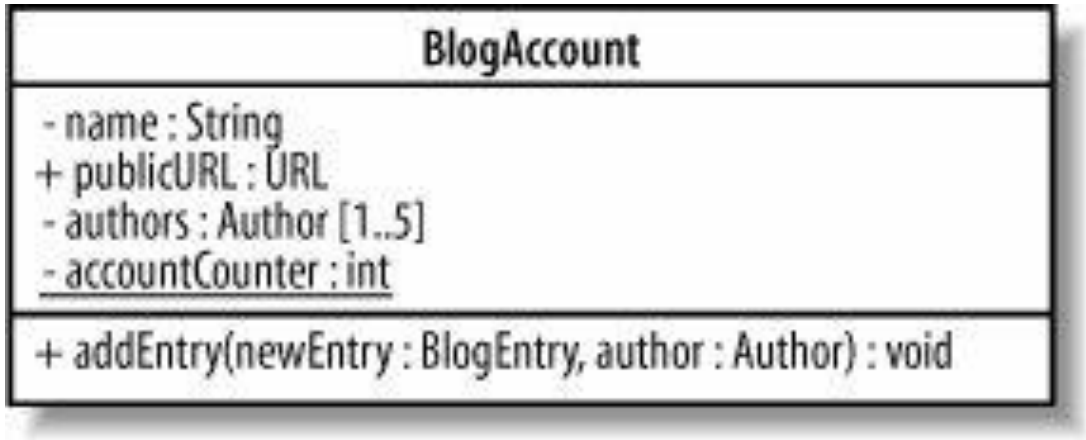



# Gradul de multiplicitate

- 1 – un obiect
- 0…* sau * - numar nedefinit de obiecte
- N – un numar definit de obiecte

# Atribute si operatii statice

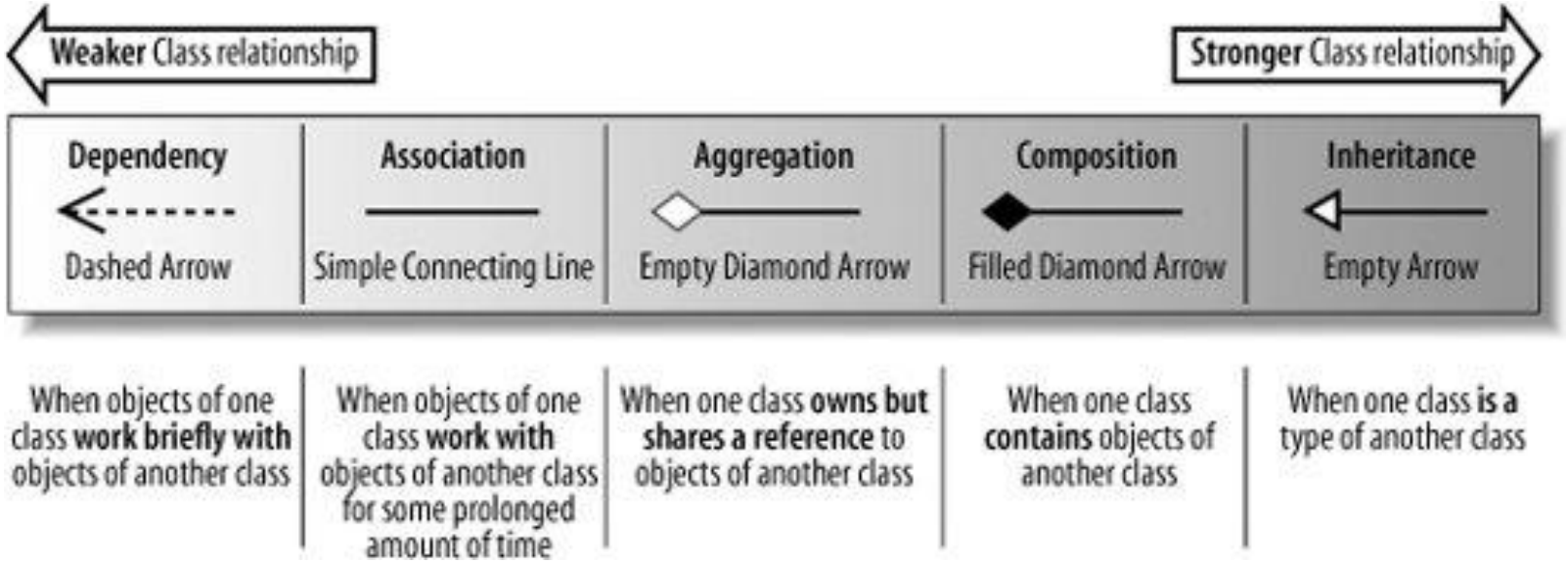
- Atributele si metodele subliniate sunt considerate statice



# Relatii intre clase

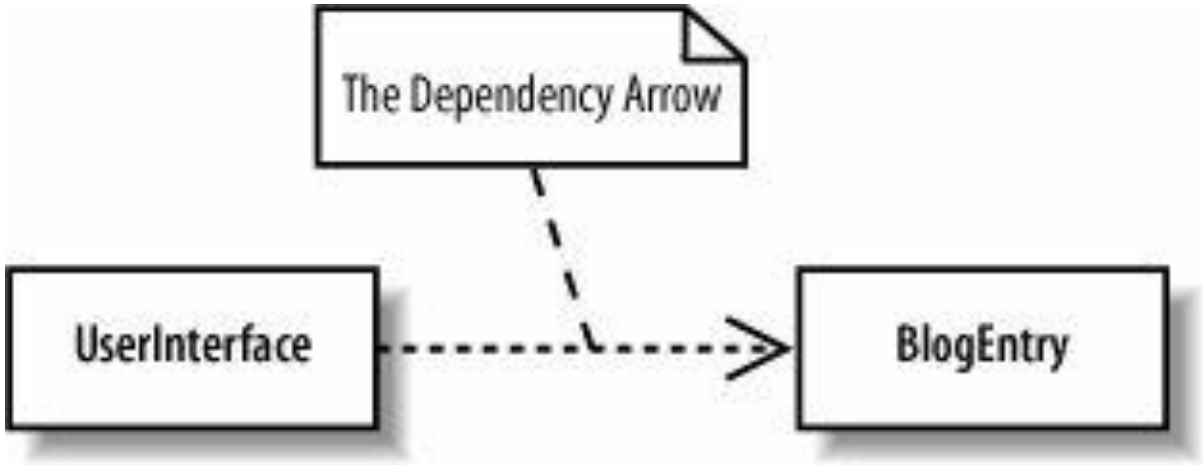
# Tipuri de relatii

- Relatiile expirma diferite tipuri de interdependenta intre clasa
- Tipuri de relatii:
  - Dependenta
  - Asociere
  - Agregare
  - Compozitie
  - Mostenire



# Relatia de dependenta

- Cel mai slab tip de interdependenta
- O clasa utilizeaza un obiect la un moment data (de exemplu la apelul unei metode)

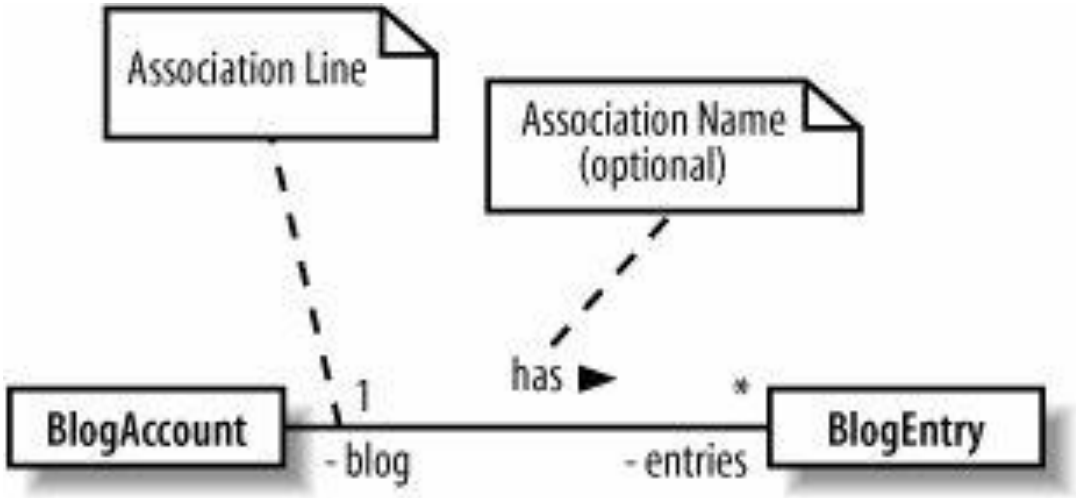


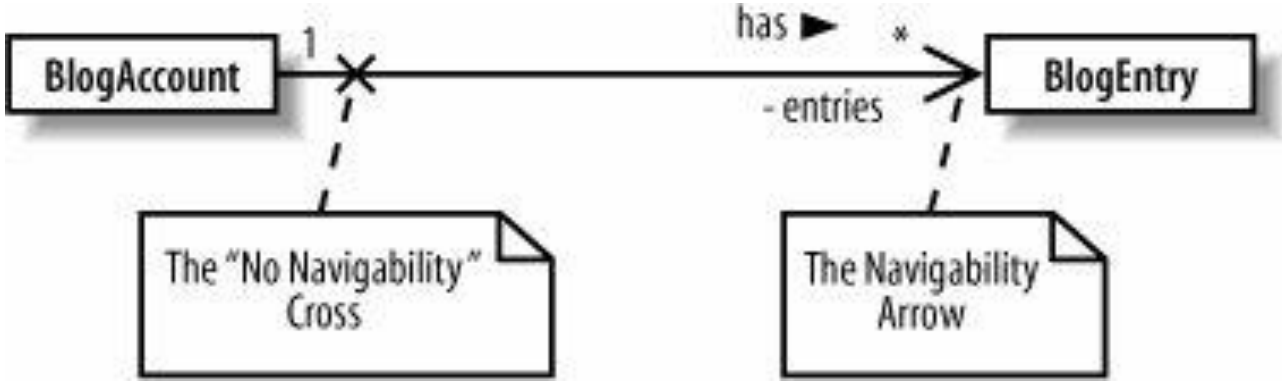
# Relatia de dependenta

```
class BlogEntry {
    private String text;
    String getText() {
        return text;
    }
}

public class UserInterface {
    void display(BlogEntry entry){
        System.out.println("Nlog entry:"+entry.getText());
    }
}
```

# Relatia de asociere

- Asociere unidirectionala
- Asociere bidirectionala
- O clasa contine un atribut al altei clase



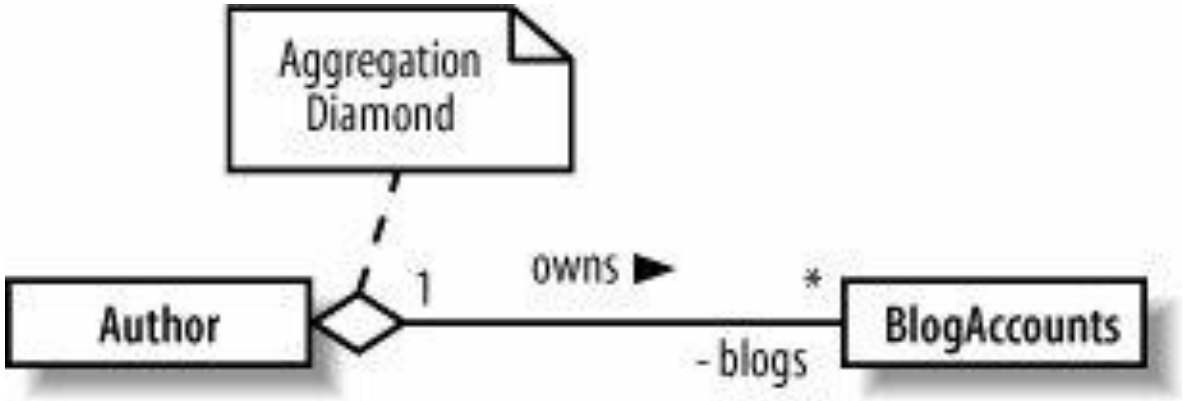
# Realatia de asociere

```java
import java.util.ArrayList;

public class BlogAccount {

    private ArrayList<BlogEntry> entries;
    //....
}


class BlogEntry{
    //.....
}
```

```java
import java.util.ArrayList;

public class BlogAccount {
    private ArrayList<BlogEntry> entries;
    //....
}


class BlogEntry{
    BlogAccount account;
    //.....
}
```

# Relatia de agregare

- O versiune a relatiei de asociere
- O clasa contine obiecte ale altei clase

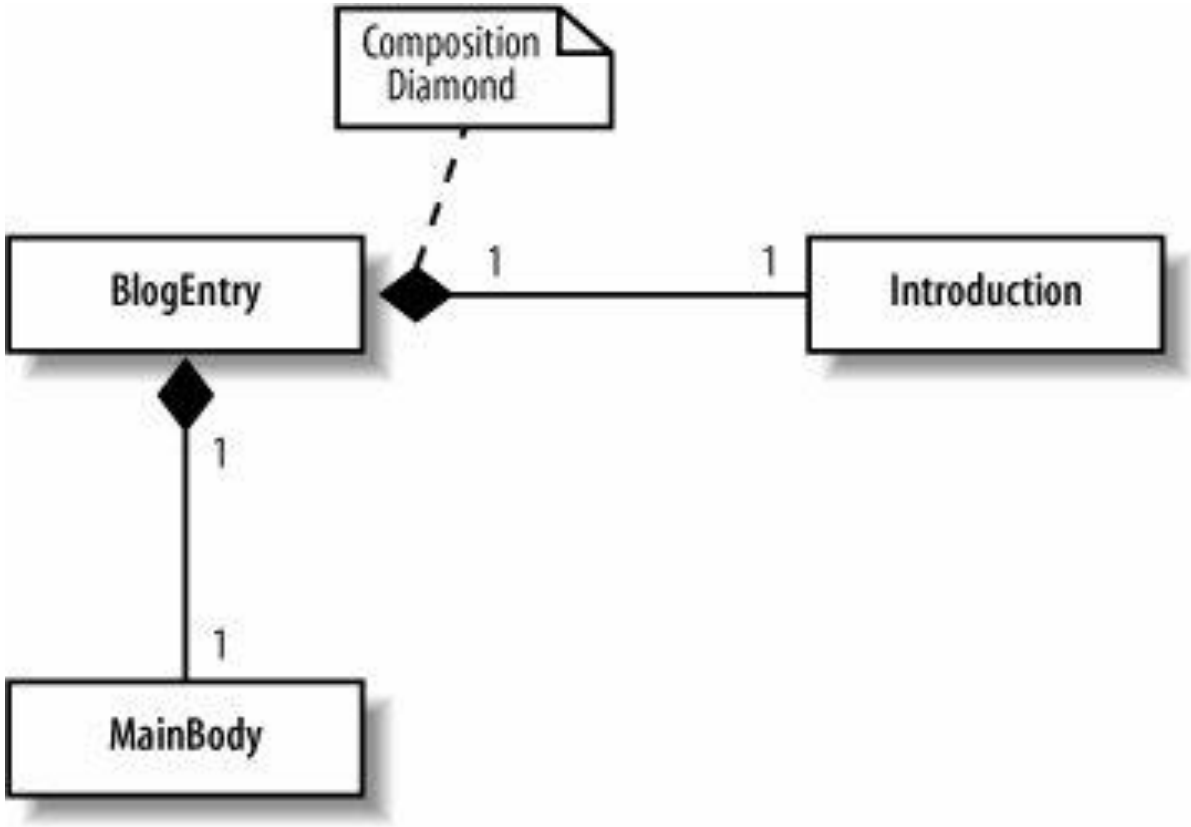


# Relatia de agregare

```
import java.util.ArrayList;


public class Author {
   ArrayList<BlogAccount> blogs =  new ArrayList<>();
}

class BlogAccount{

}
```

# Relatia de compozitie

- O varianta a agregarii in care ciclul de viata al obiectelor compuse este strans dependent de cel al clasei care le contine
- Reprezinta partile interne ale clase

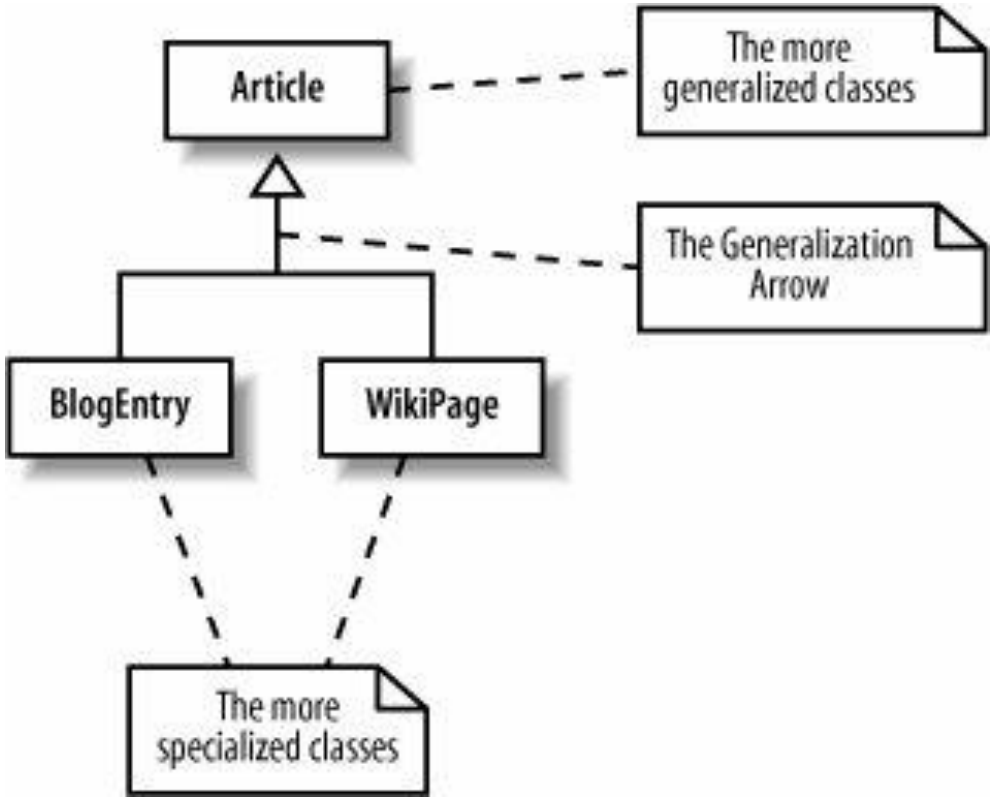


# Relatia de compozitie

```
public class BlogEntry {
    Introduction i;
    MainBody m;

    public BlogEntry() {
        i = new Introduction();
        m = new MainBody();
    }
}

class Introduction {
}

class MainBody {
}
```

# Relatia de generalizare

- O clasa mosteneste o alta clasa

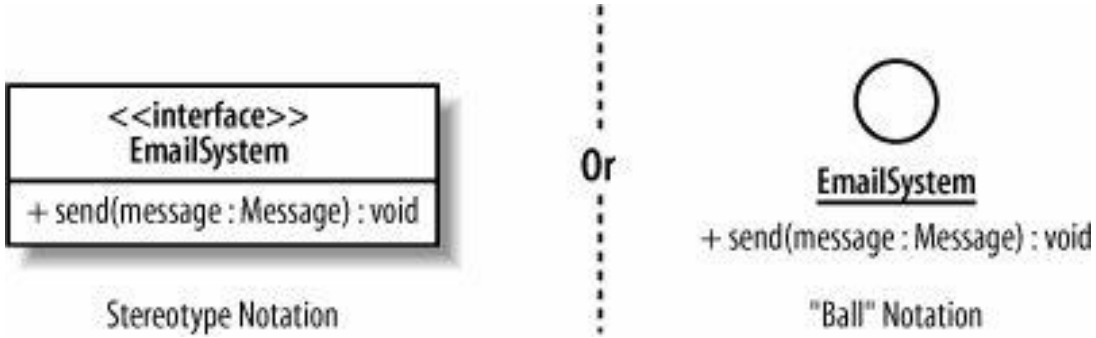


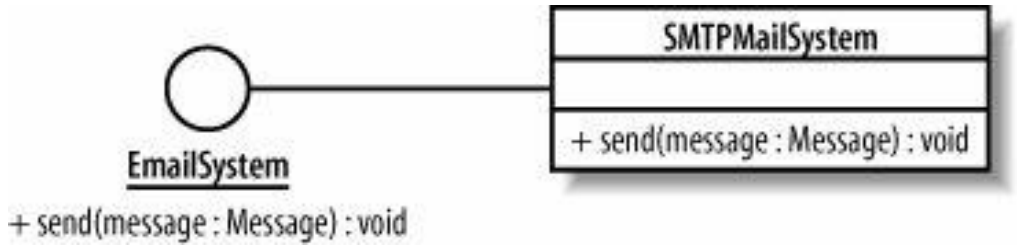
# Relatia de generalizare

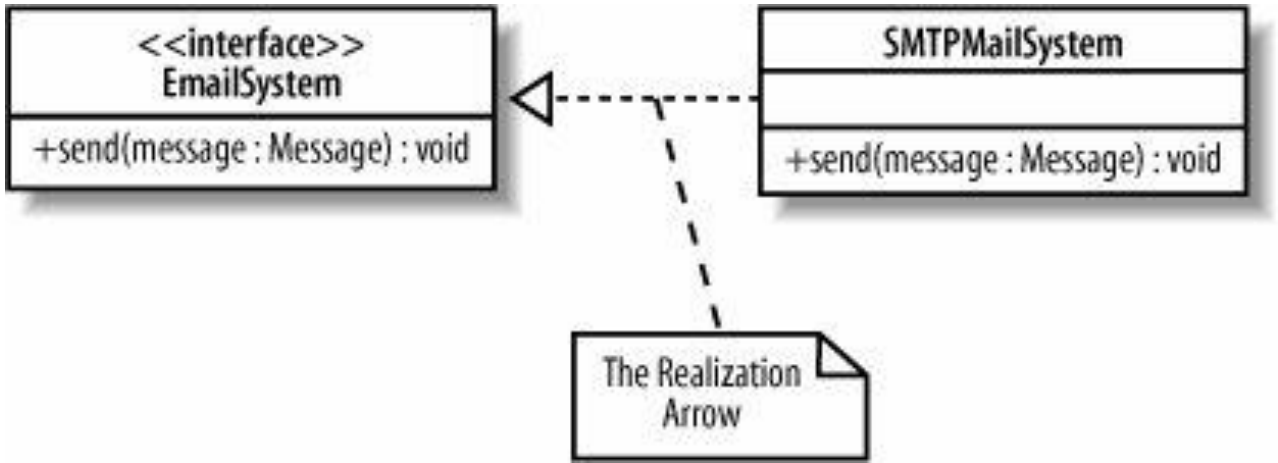
```
class Article{

}

class BlogEntry extends Article{

}

class WikiPage extends Article{

}
```

# Relatia de implementare

- Reprezentarea interfetelor:
  - Schematic (un cerc)
  - Similar cu clasele utilizand stereotipuri
- O clasa implementeaza o interfata



# Relatia de implementare

```
interface EmailSystem{

    public void sendMessage(Message message);
}

class SMTPEmailSystem implements EmailSystem{

    @Override
    public void sendMessage(Message message) {
        // ... some implementation
    }

}
```