DePaul University

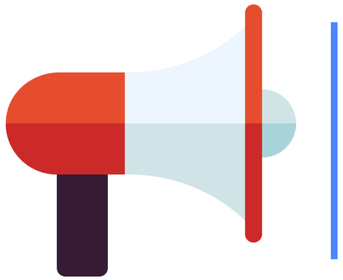
Software Documentation:

UML Class Diagrams

Object-oriented Software Development
SE 350– Spring 2021

Vahid Alizadeh

Week 5.1
April 27, 2021



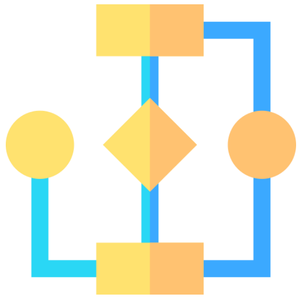
Announcements

Future Schedule

Assignment 1 is graded

Check the feedbacks on D2L

- ~~Assignment 1:~~
  - ~~Release: Week 3.1~~
  - ~~Due: Week 3.1~~
- Assignment 2:
  - ~~Release: Week 4.1 (Today)~~
  - Due: Week 5.1 > extended to April 28
- Mid Term Exam:
  - Week 5.2
  - Thursday – April 29, 2021
  - No Class
  - Take home exam
  - The Midterm questions and a video recording about it will be released 8 AM
  - You have the whole day to submit by 11:59 PM
  - No Limit on the amount of time you spend on the midterm

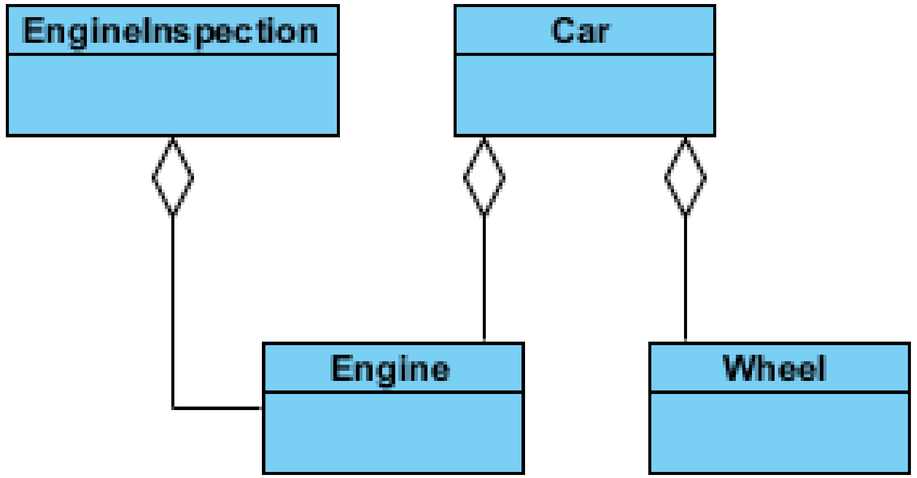


Design Documentation

UML

Aggregation

- Aggregation
  - Special kind of association
    - whole- part model
  - unidirectional (One-way) relationship
    - “Has-a”, “is part of”
  - Only one class is dependent on the other
  - Both the entries can survive individually
  - Illustrate composition with a hollow diamond
    - The diamond end points toward the "whole" class
- When to use Inheritance and Aggregation?
  - Use property/behavior without modification or add functionality > Aggregation
  - Use and modify property/behavior and add functionality > Inheritance

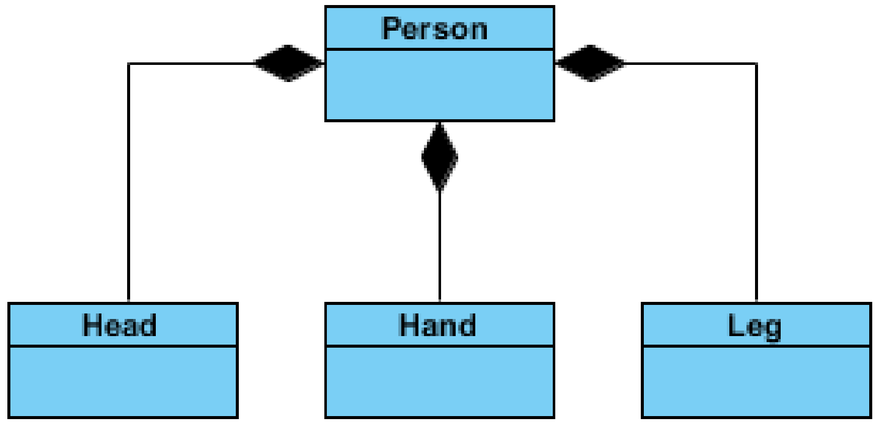


Composition

- **Composition**
  - “has-a”, “part of”, “belongs-to”
  - Stricter form of aggregation
  - Two classes are mutually dependent
  - Cannot exist without each other / life span are same
  - Illustrate composition with a filled diamond
    - The diamond end points toward the "whole" class

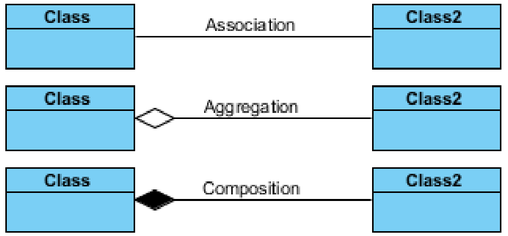
```
1 class Car
2 {
3   private Engine engine;
4   Car(Engine en)
5   {
6     engine = en;
7   }
8 }
```

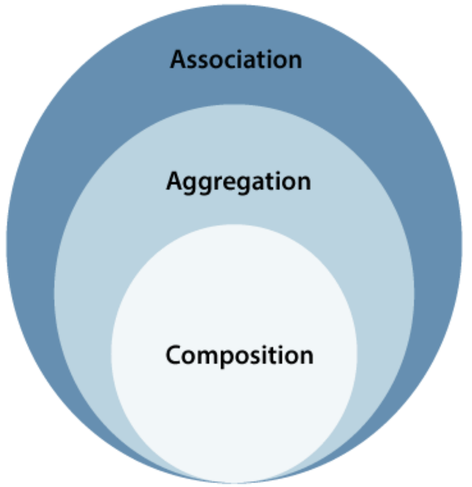
a class **Car** cannot exist without **Engine**,
it won't be functional anymore



Association vs. Aggregation vs. Composition

Attributes	Aggregation	Composition
Definition	Aggregation is denoted by a straight line with one end of the null-diamonded arrow mark.	The composition is denoted by a straight line with a full diamonded arrow mark at one end.
Dependency	It explains the section of the association relationship.	It explains the section of the aggregation relationship.
Type of Relationship	The relationship established is very delicate.	The relationship established is definitely strong.
Associate Objects	In aggregation, the related objects are present independently within the entire system.	In composition, the related objects are nor present within the entire system.
Linked Objects	The linked objects present in the system are independent to other objects.	The linked objects present in the system are dependent on other objects.
Deletion of Objects	Deletion of one element in the aggregation doesn't impact the other components present in the associative section.	The deletion of one object impacts the presence of the other related objects in the compression.
Examples	For the proper functioning of a car, the wheel is a mandatory element. But it doesn't require the same wheel for the operation of the car. Even other wheels can be substituted.	If an image is placed in a document and that document gets deleted, the image cannot be found again or used. The image residing in the document also gets deleted at the same time at the time of document deletion.
Representation		





Relationships Comparison with Example

- Dependency (references)

```java
public class EnrollmentService {
    public void enroll(Student s, Course c){}
}
```

- Association (has-a)

```java
public class Order {
    private Customer customer
}
```

- Aggregation (has-a + whole-part)

```java
public class PlayList {
    private List<Song> songs;
}
```

- Composition (has-a + whole-part + ownership)

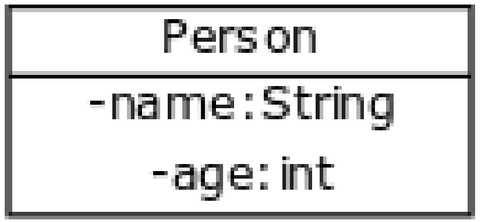
```java
public class Apartment{
    private Room bedroom;
    public Apartment() {
        bedroom = new Room();
    }
}
```

Describing class and attributes

- Class attribute syntax

```
attributeName: attributeType
```

```
public class Person {
    private String name;
    private int age;
}
```

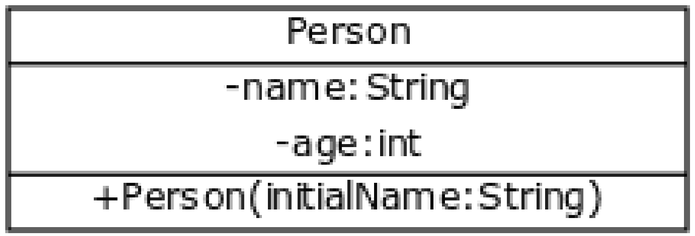


Describing class constructor

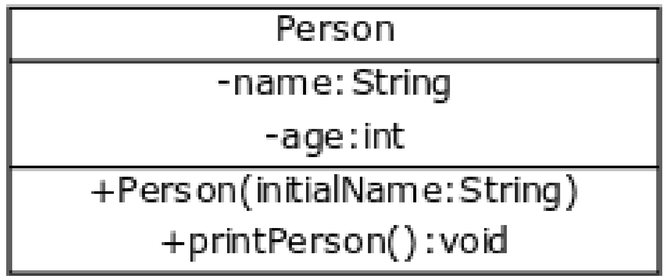
- Method parameter syntax

```
parameterName: parameterType
```

```java
public class Person {
    private String name;
    private int age;

    public Person(String initialName) {
        this.name = initialName;
        this.age = 0;
    }
}
```



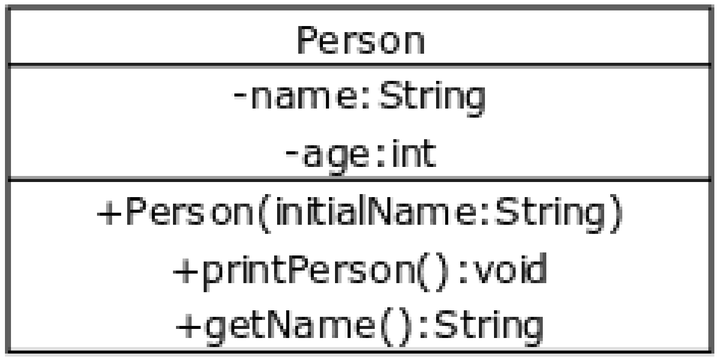
Describing class methods




```java
public class Person {
    private String name;
    private int age;

    public Person(String initialName) {
        this.name = initialName;
        this.age = 0;
    }

    public void printPerson() {
        System.out.println(this.name + ", age " +   this.age + " years");
    }
}
```

More Methods

```java
public class Person {
    private String name;
    private int age;

    public Person(String initialName) {
        this.name = initialName;
        this.age = 0;
    }

    public void printPerson() {
        System.out.println(this.name + ", age " +   this.age + " years");
    }

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



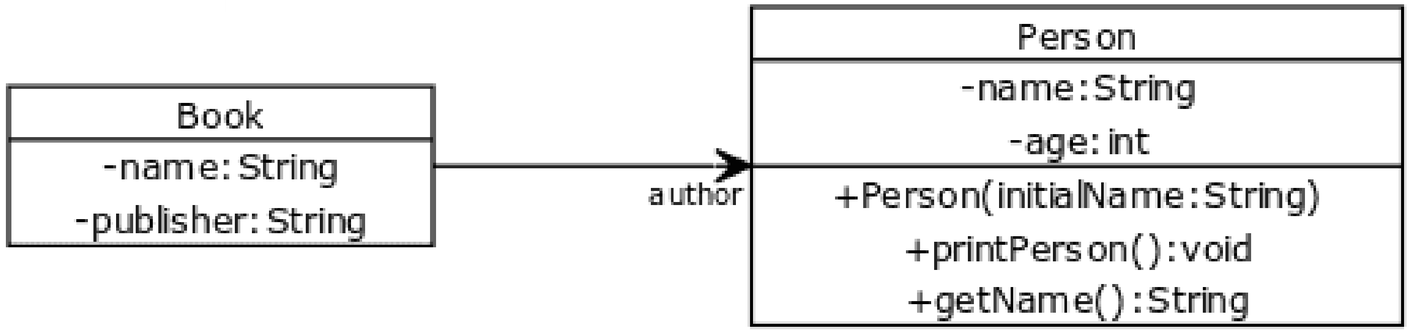
Visibility of Class Members

public	+	anywhere in the program and may be called by any object within the system
private	-	the class that defines it
protected	#	(a) the class that defines it or (b) a subclass of that class
package	~	instances of other classes within the same package

Connections between classes

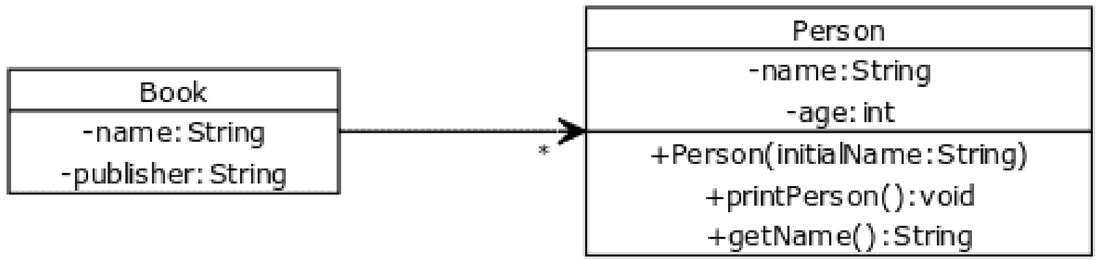
- **Connections**
  - **Shown as arrows**
  - **Show direction**

```java
public class Book {
    private String name;
    private String publisher;
    private Person author;

    // constructors and methods
}
```



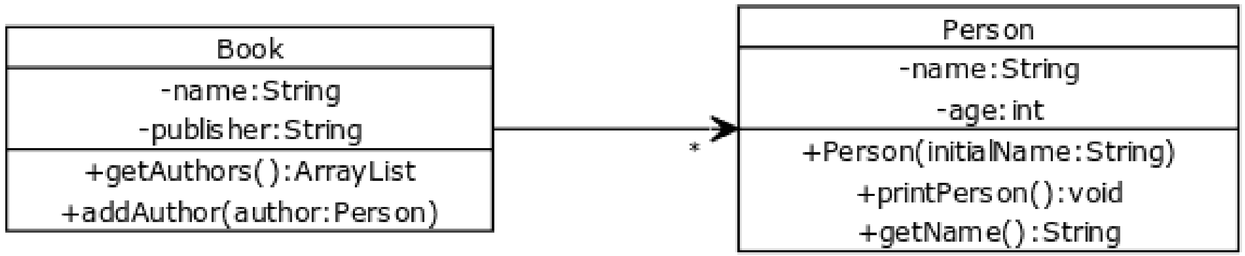
Connections between classes

```java
public class Book {
    private String name;
    private String publisher;
    private ArrayList<Person> authors;

    // constructors and methods
}
```



Connections between classes

```java
public class Book {
    private String name;
    private String publisher;
    private ArrayList<Person> authors;

    // constructor

    public ArrayList<Person> getAuthors() {
        return this.authors;
    }

    public void addAuthor(Person author) {
        this.authors.add(author);
    }
}
```



Exercise

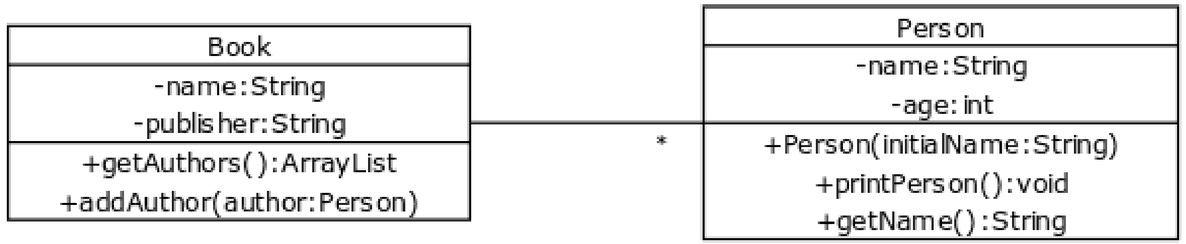
- Tickets and Show Class Diagram
- Implement the classes in this diagram

Exercise: Solution


```java
public class Show {
    private String movie;
    private String time;
}
```

```java
public class Ticket {
    private int seat;
    private int code;
    private Show show;
}
```

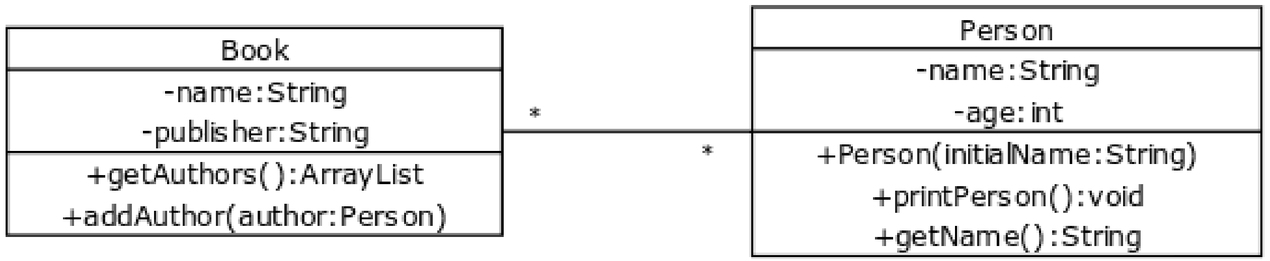
Connection with no arrows




```java
public class Book {
    private String name;
    private String publisher;
    private ArrayList<Person> authors;

    // ..
}
```

```java
public class Person {
    private String name;
    private int age;
    private Book book;

    // ...
}
```

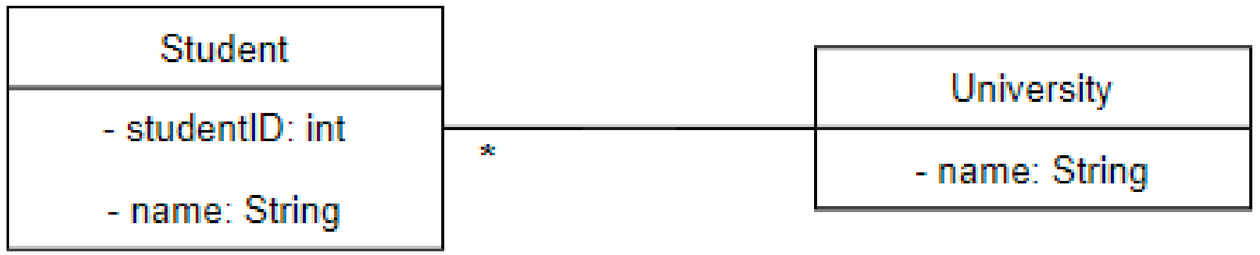
Connection with no arrows



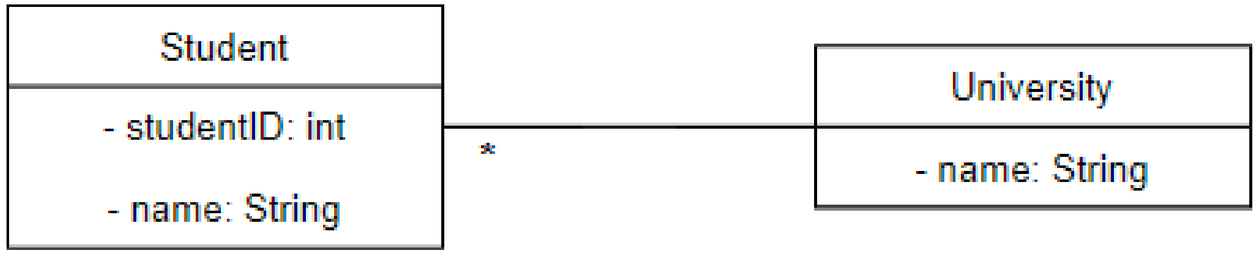

```java
import java.util.ArrayList;

public class Person {
    private String name;
    private int age;
    private ArrayList<Book> books;

    // ...
}
```

Exercise

- Student and university diagram
- Implement the classes in this diagram



Exercise: Solution



Student.java

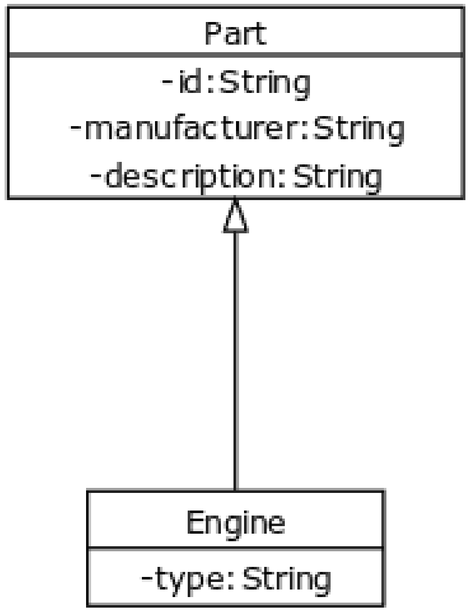
```java
public class Student {
    private int studentID;
    private String name;
    private University university;
}
```

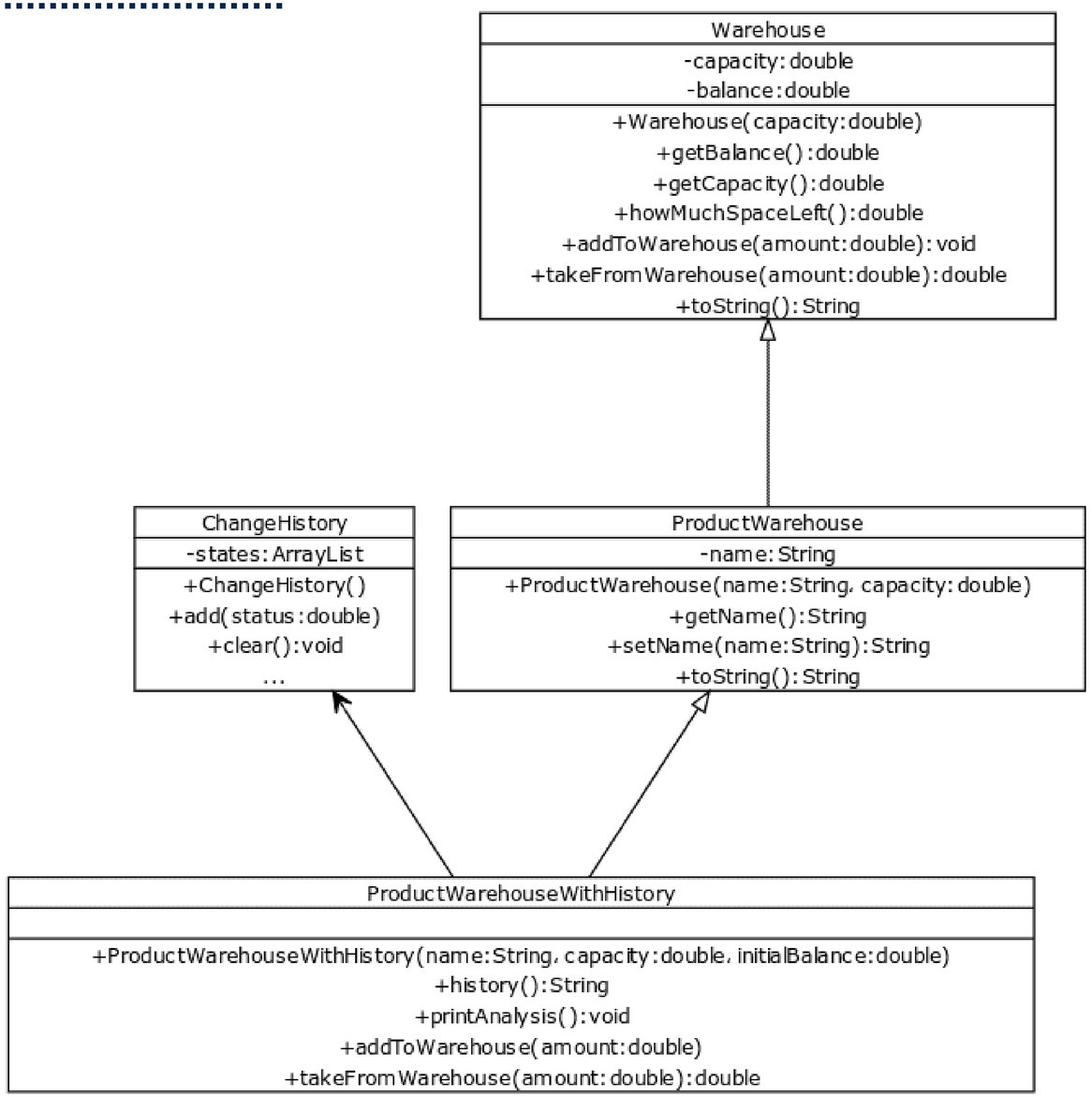
University.java

```java
import java.util.ArrayList;

public class University {
    private String name;
    private ArrayList<Student> students;
}
```

Describing Inheritance with UML

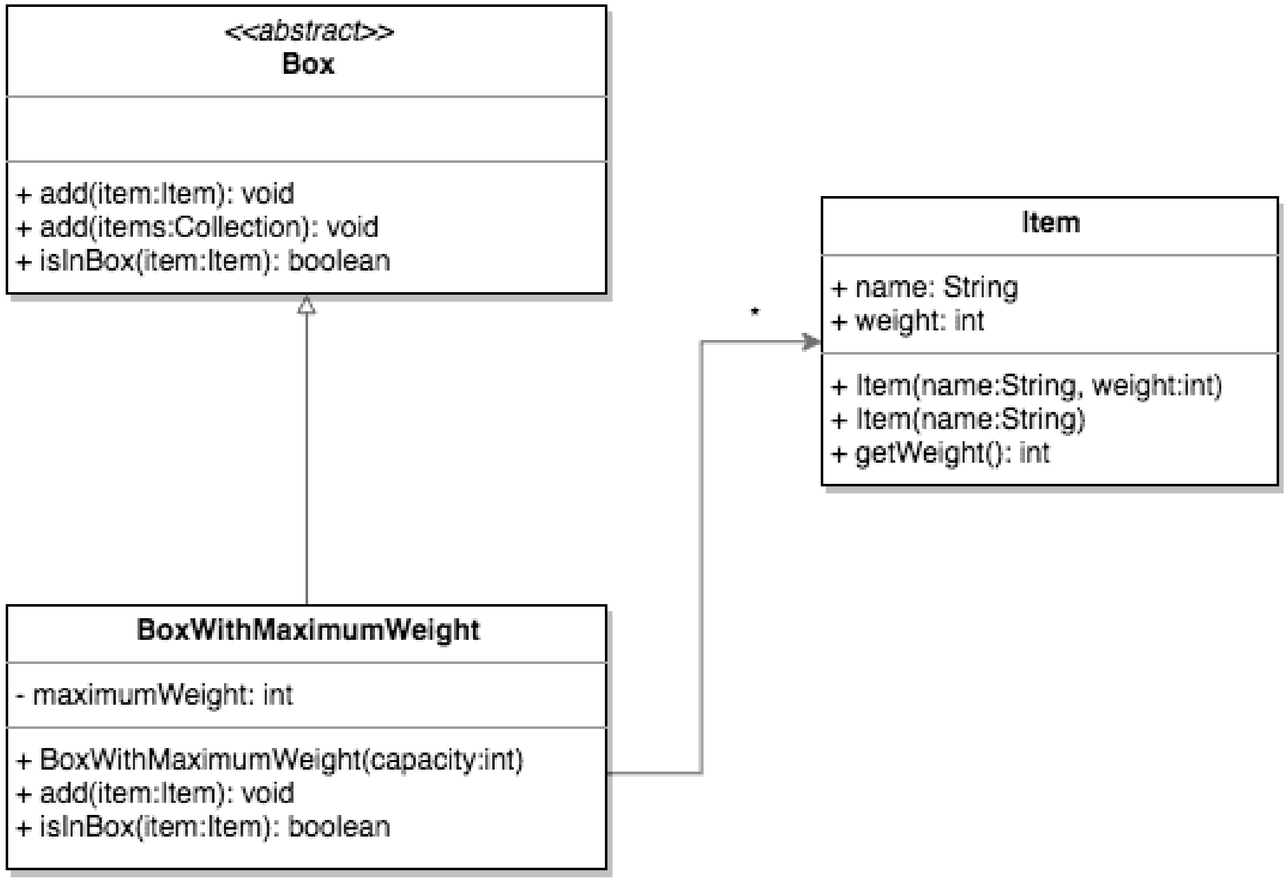
- Arrow with a triangle head
- Points to the class being inherited from





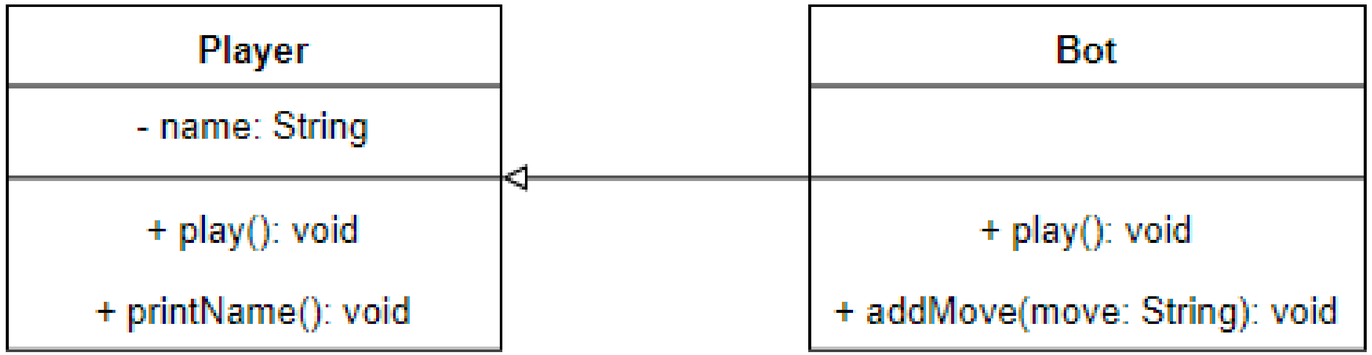
Inheritance of Abstract Classes

- Add the description <<abstract>> above the name of the class.

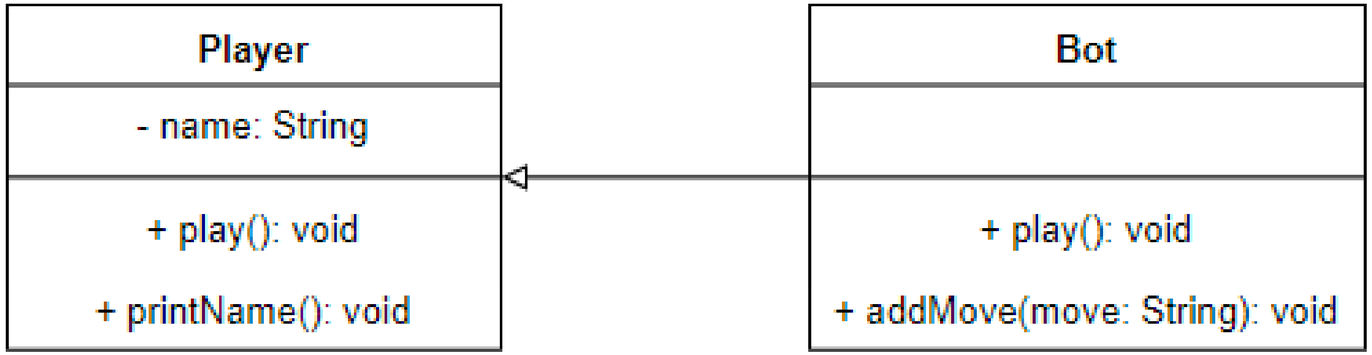


Exercise

- Player and Bot class diagram.
- Implement the classes in this diagram



Exercise: Solution



Player.java

```java
public class Player {
    private String name;

    public void play() {

    }

    public void printName() {

    }
}
```

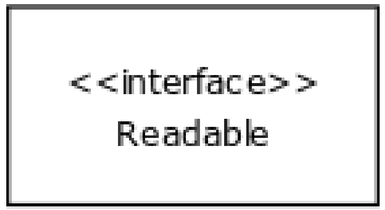
Bot.java

```java
public class Bot extends Player {

    @Override
    public void play() {

    }

    public void addMove(String move) {

    }
}
```

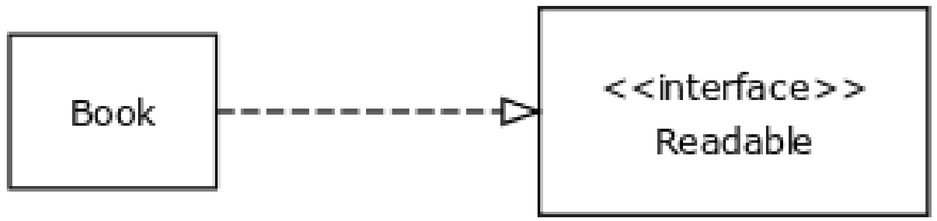
Describing Interfaces with UML

- **<<interface>> NameOfTheInterface**

```
1 public interface Readable {
2
3 }
```

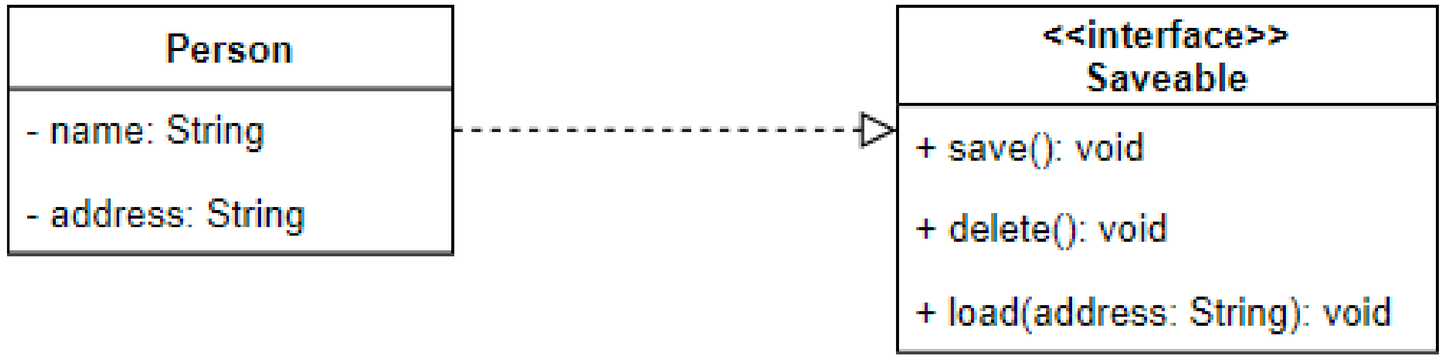


- **Implement an interface with dashed arrows and a triangle arrowhead.**

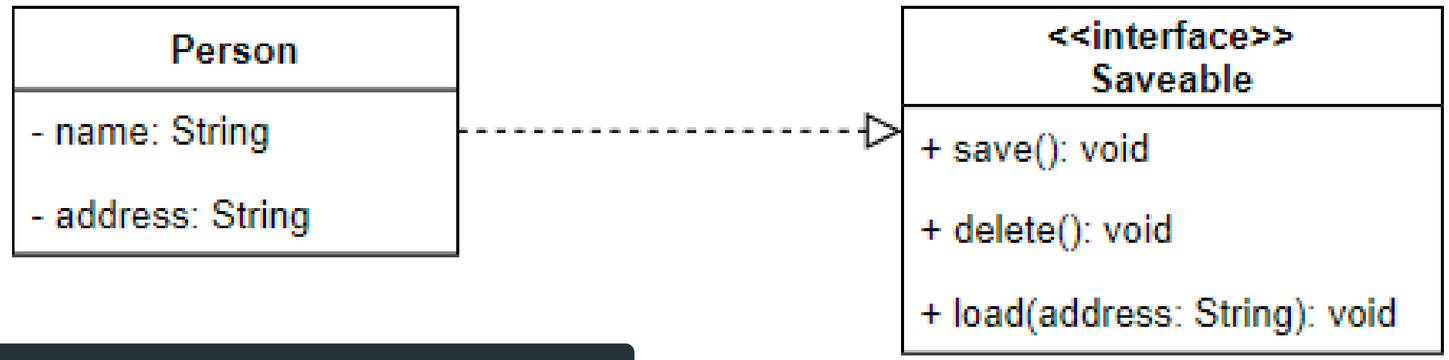


Exercise

- Interface Saveable and Class Person.
- Implement this class diagram.



Exercise: Solution

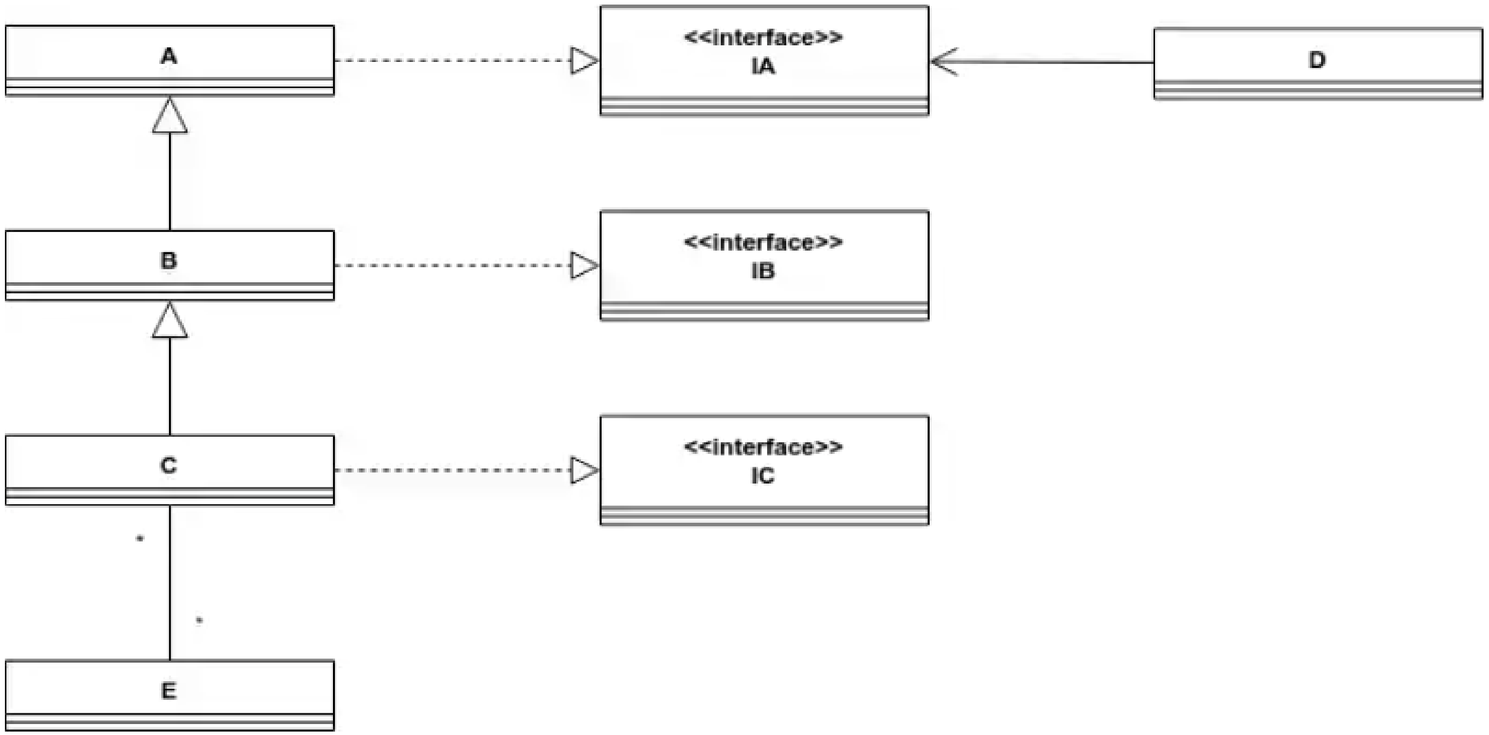



```java
public class Person implements Saveable {
    private String name;
    private String address;

    @Override
    public void save() {

    }

    @Override
    public void delete() {

    }

    @Override
    public void load(String address) {

    }
}
```

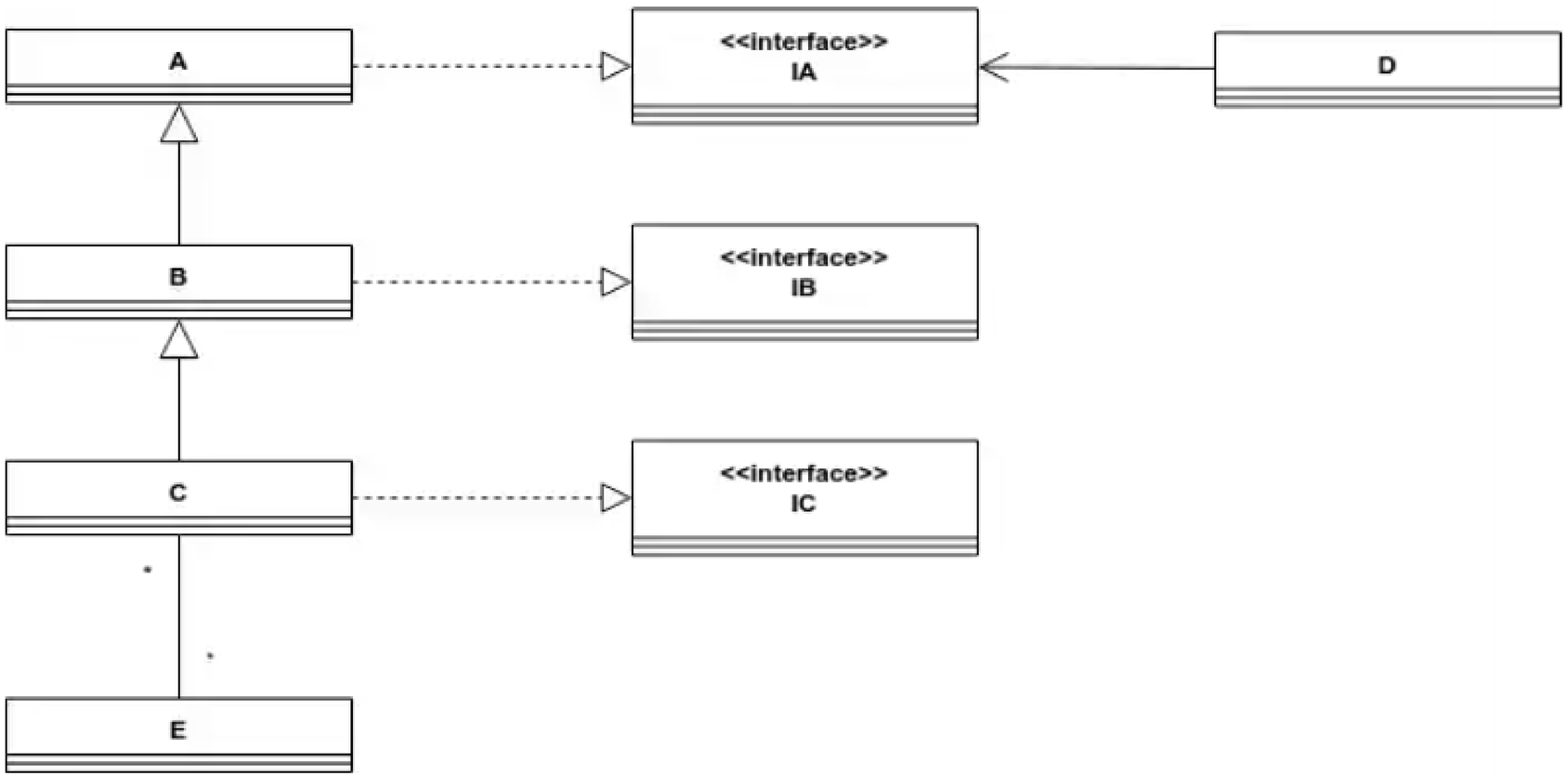
```java
public interface Saveable {

    void save();
    void delete();
    void load(String address);

}
```

Exercise

- A larger diagram example.
- Implement this class diagram.



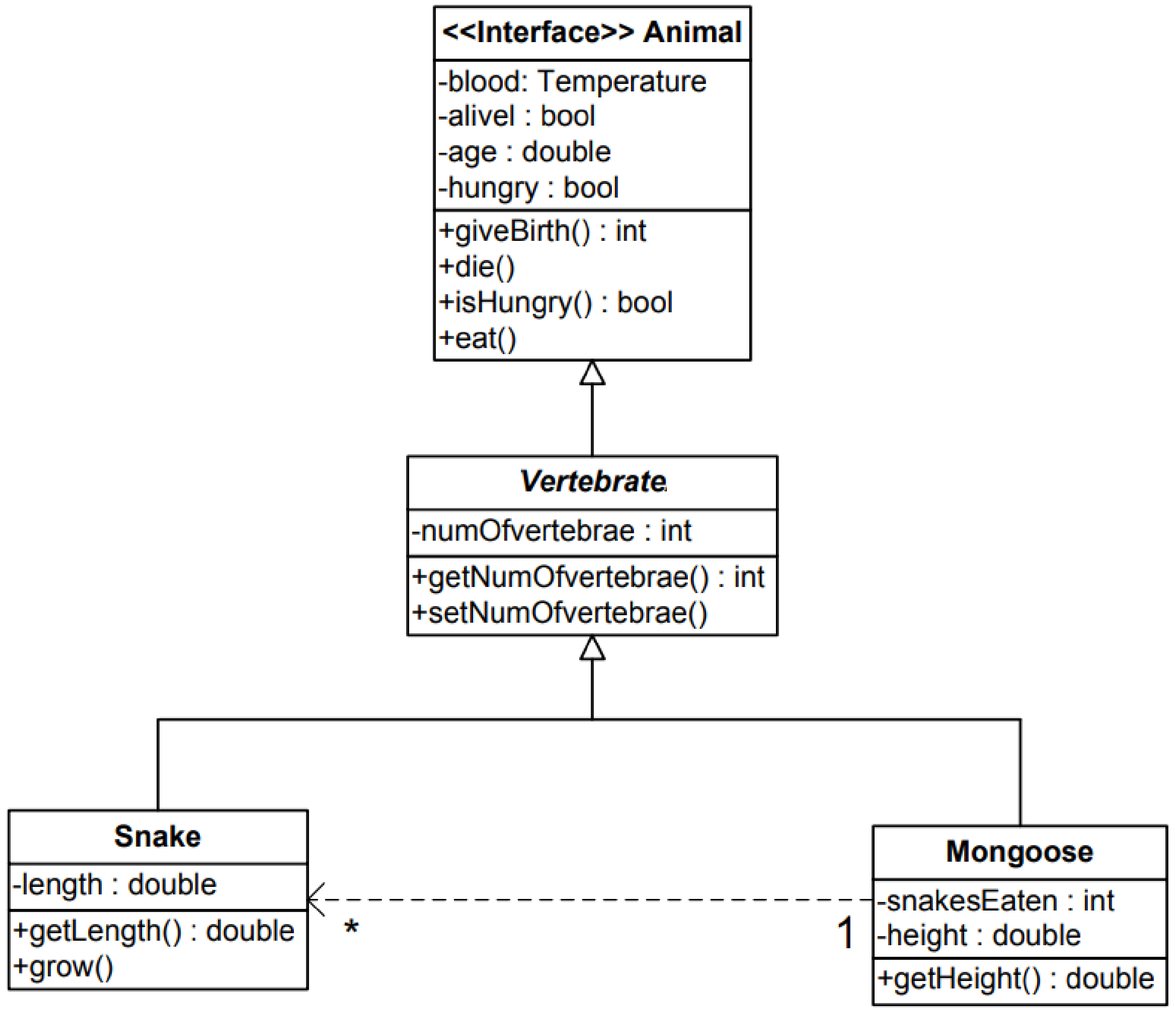
Exercise: Solution




```java
public class A implements IA  {

}

public class B extends A implements IB {

}

public class C extends B implements IC {
    private ArrayList<E> e;
}

public class D {
    private IA ia;
}

public class E {
    private ArrayList<C> c;
}

public interface IA {

}

public interface IB {

}

public interface IC {

}
```

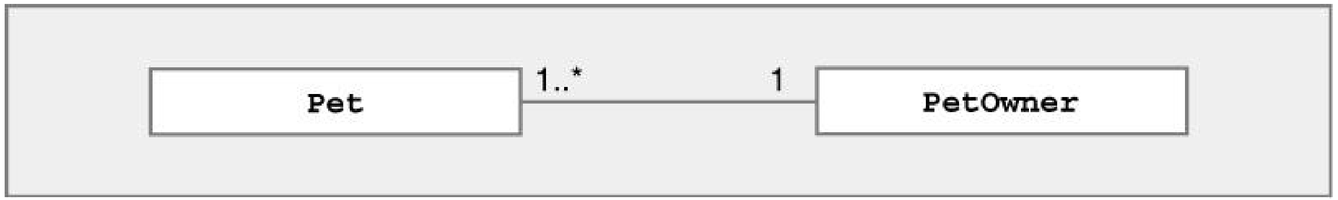
Exercise

- What is the relationship between Vertebrate and Animal?
  - Vertebrate is a subclass of Animal.
- What is the relationship between Snake and Animal?
  - Snake is a subclass of Animal.
- What is the association type between Mongoose and Snake?
  - Dependency association.
- According to the class diagram, can 2 mongooses eat 1 snake?
  - No. The relationship between Mongoose and Snake is one-to-zero-or-more.
- Identify one thing wrong with this diagram.
  - Because Animal is an interface, the relationship from Vertebrate should be dashed.



UML Examples: Veterinary System

- Read UML class diagram
- Try to understand relationships

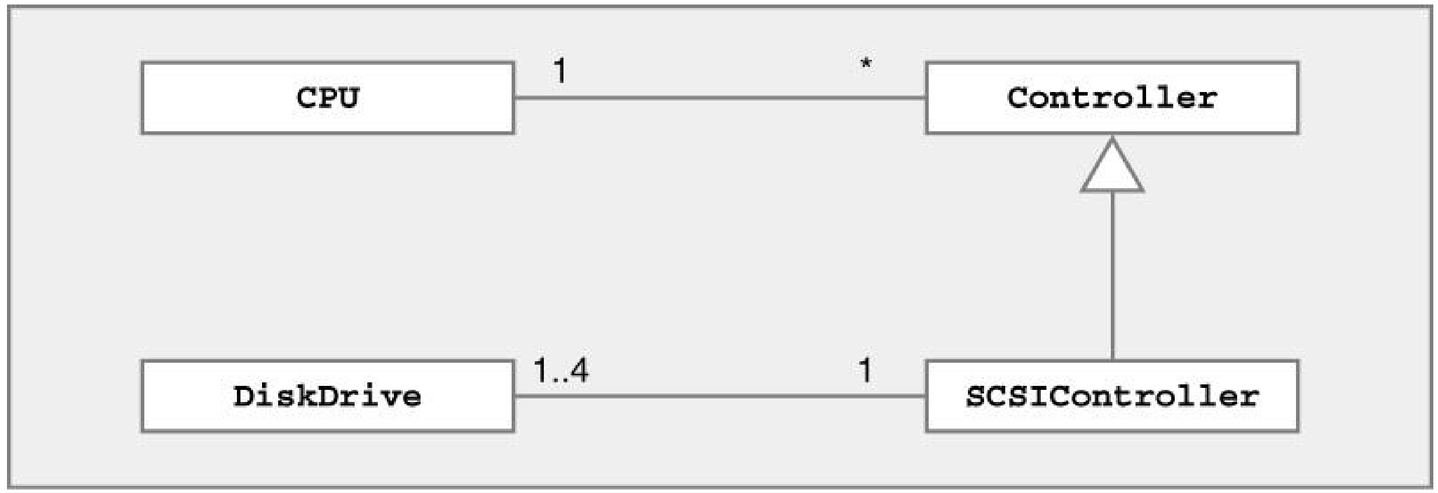


1 or more Pets associated with 1 PetOwner

```
1 class Pet {
2     PetOwner myOwner;   // 1 owner for each pet
3 }
4 class PetOwner {
5     Pet [ ] myPets; // multiple pets for each owner
6 }
7
```

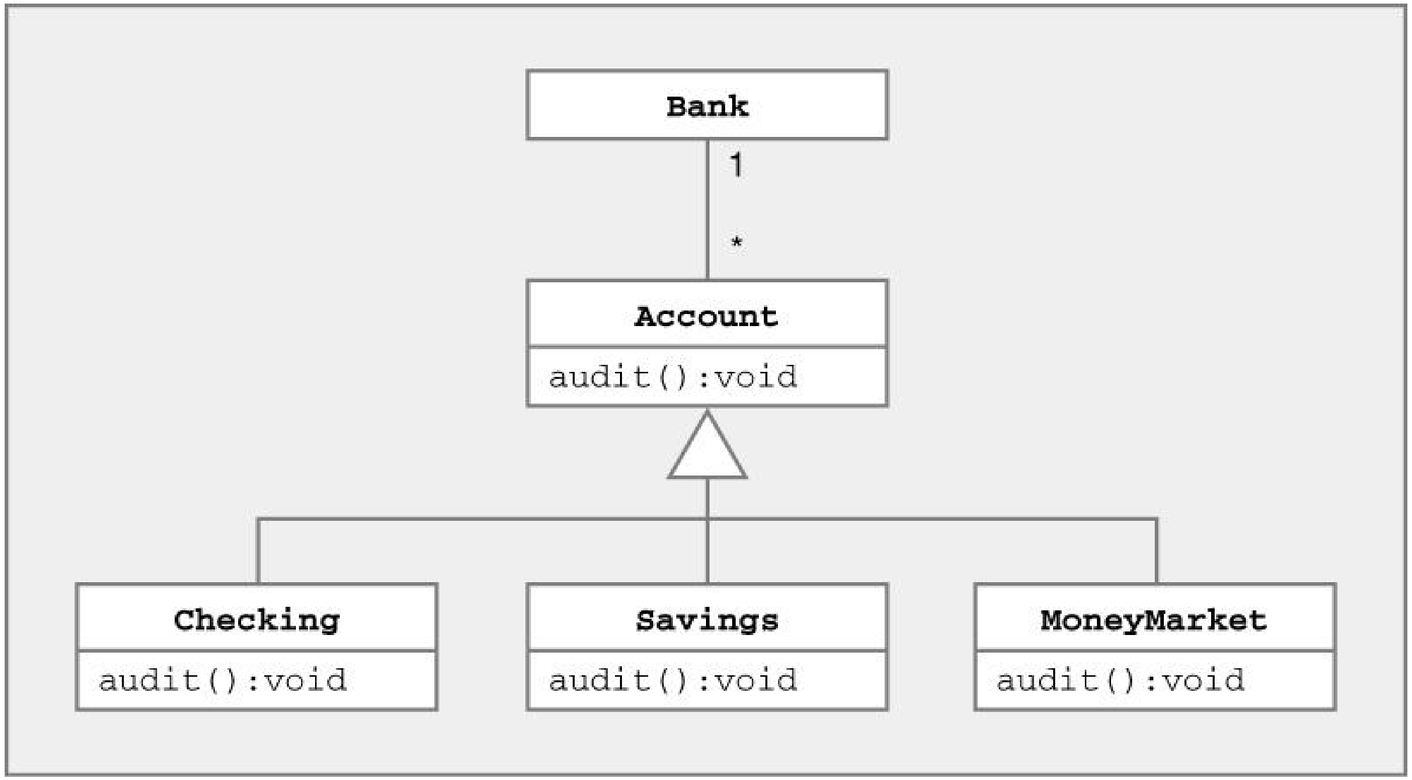
UML Examples: Computer System

```
1  class CPU {
2      Controller [ ] myCtlrs;
3  }
4  class Controller {
5      CPU myCPU;
6  }
7  class SCSIController extends Controller {
8      DiskDrive [ ] myDrives = new DiskDrive[4];
9  }
10 Class DiskDrive {
11      SCSIController mySCSI;
12 }
13
```



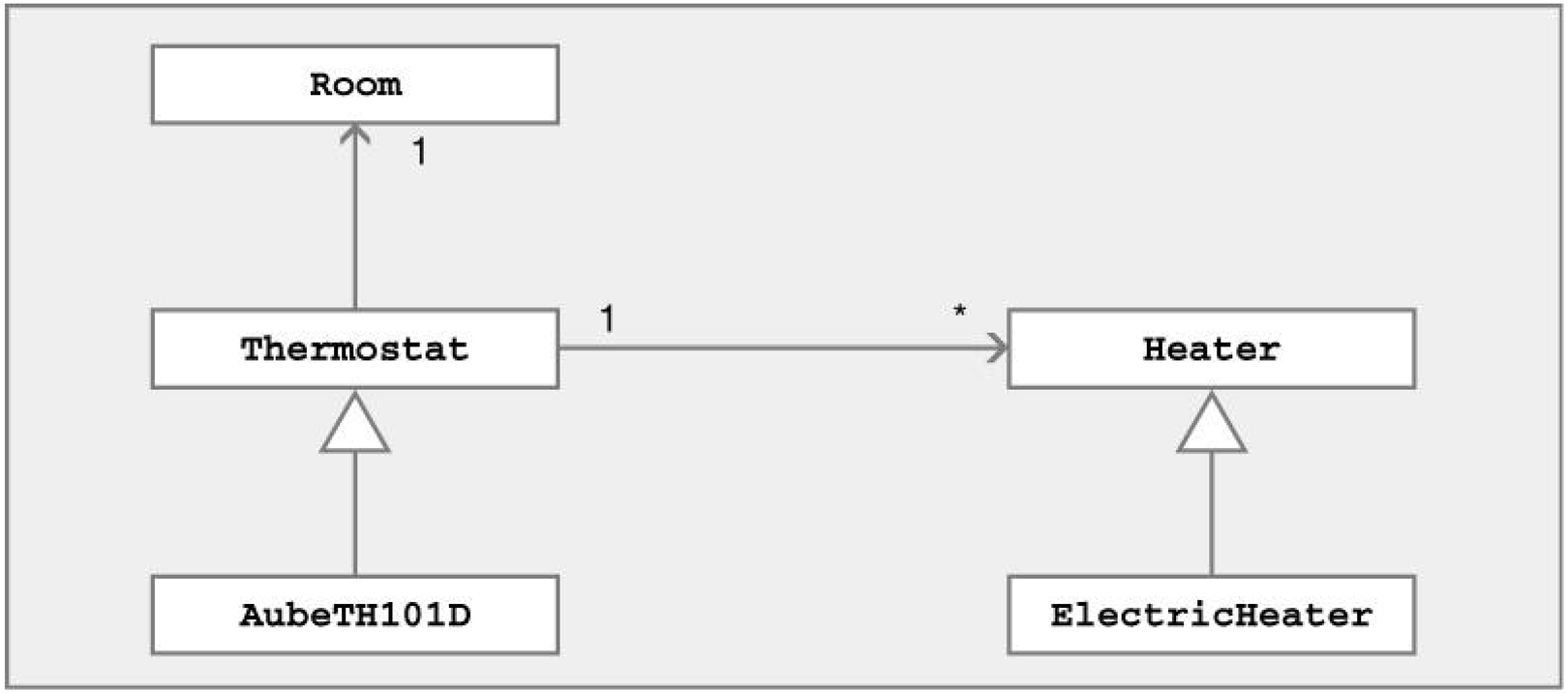
- 1 **CPU** associated with 0 or more **Controllers**
- 1-4 **DiskDrives** associated with 1 **SCSIController**
- **SCSIController** is a (specialized) **Controller**

UML Examples: Banking System



- 1 **Bank** associated with 0 or more **Accounts**
- **Checking**, **Savings**, **MoneyMarket** are **Accounts**

UML Examples: Home Heating System

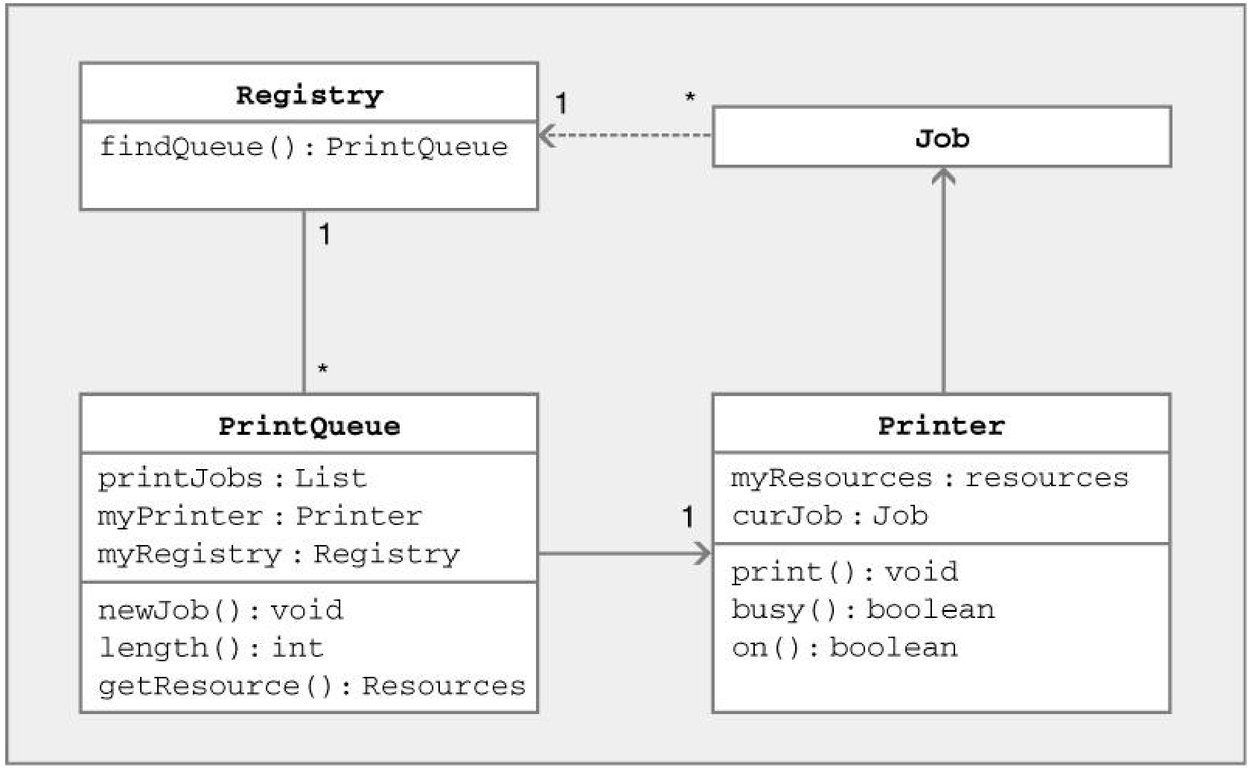


- Each **Thermostat** has 1 **Room**
- Each **Thermostat** associated with 0 or more **Heaters**
- **ElectricHeater** is a specialized **Heater**
- **AubeTH101D** is a specialized **Thermostat**

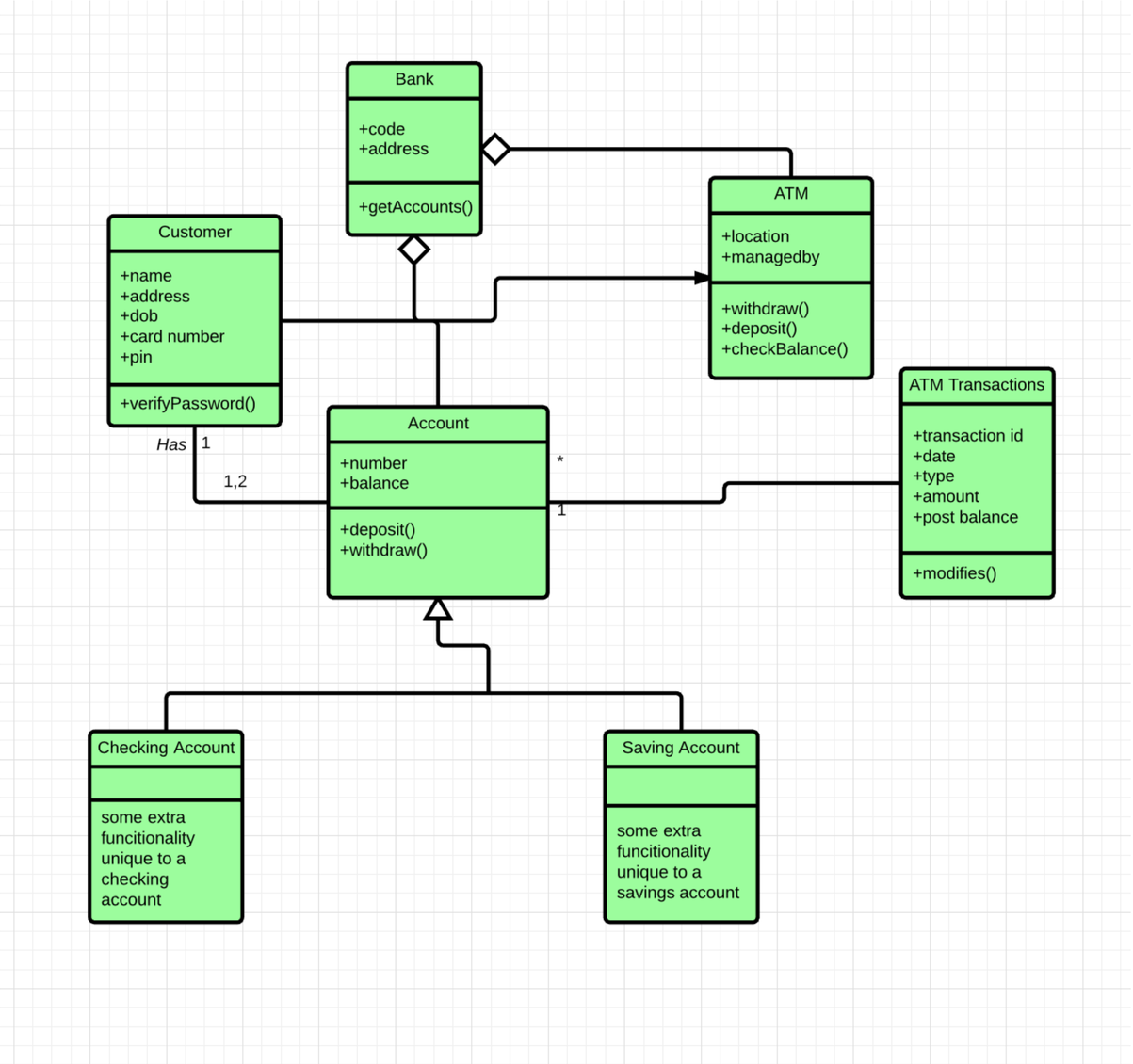
UML Examples: Printing System

```
1  class Registry {
2      PrintQueue findQueue();
3  }
4
5  class PrintQueue {
6      List printJobs;
7      Printer myPrinter;
8      Registry myRegistry;
9      void newJob();
10     int length();
11     Resources getResource();
12 }
13
```

```
1  Class Printer {
2      Resources myResources;
3      Job curJob;
4      void print();
5      boolean busy();
6      boolean on();
7  }
8
9  class Job {
10     Job(Registry r) {
11         …
12     }
13 }
```



UML Examples: Large Banking System



Resources

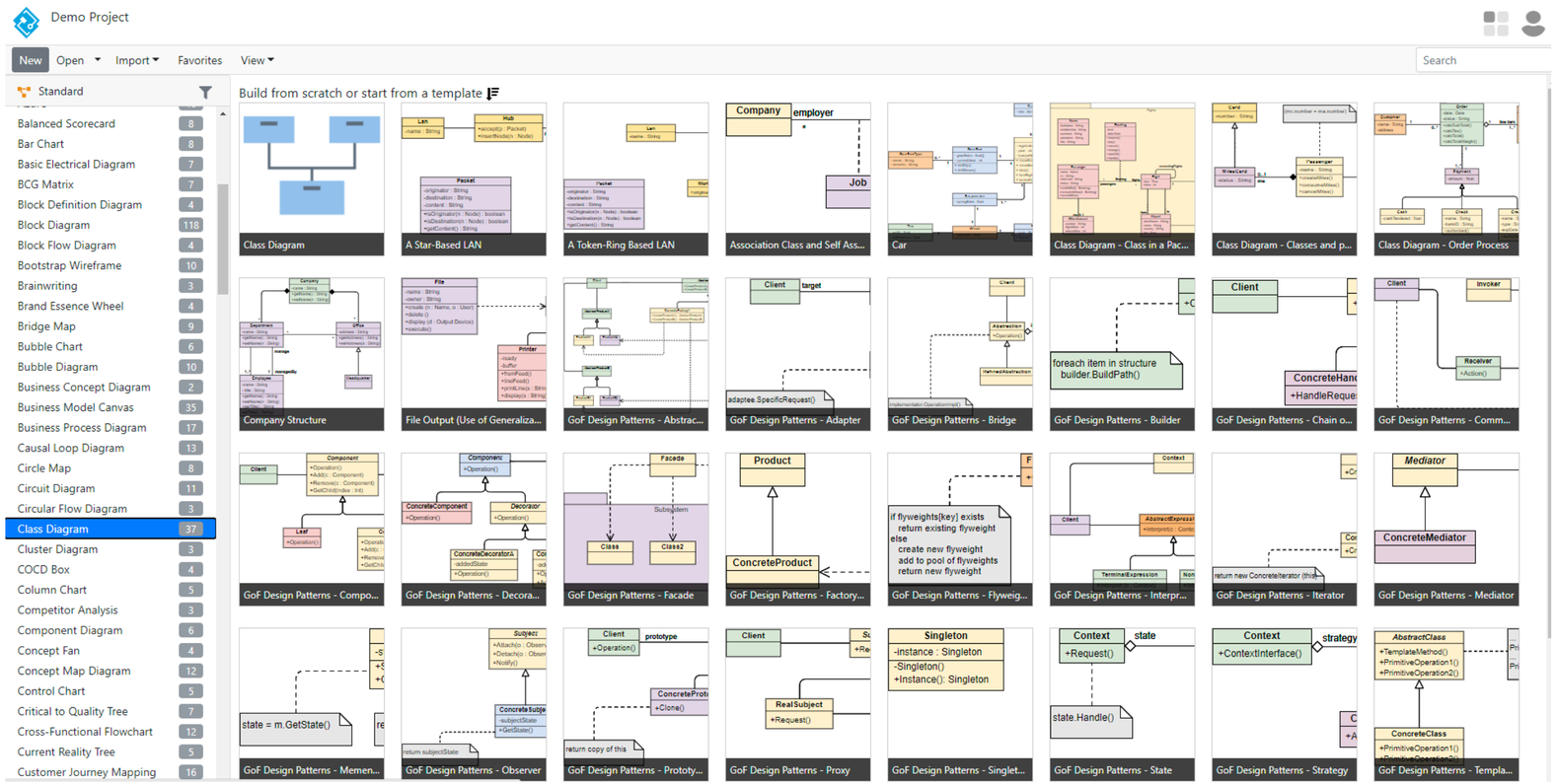
UML

UML - Tools & Utilities

- UML Tools List @ Wiki

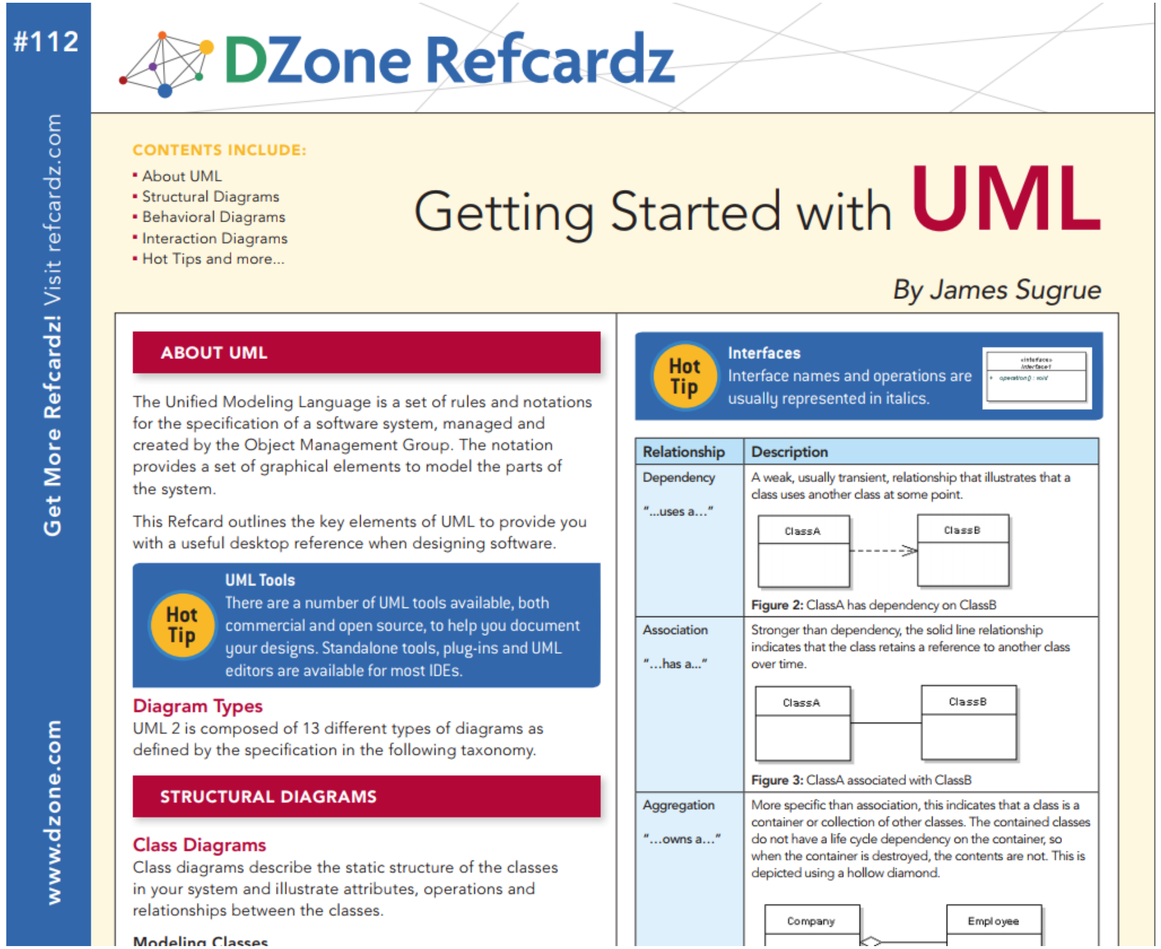
- Desktop
  - Your IDE (https://www.jetbrains.com/help/idea/class-diagram.html)
    - Code Iris Plugin
    - Sketch It Plugin
    - PlantUML Plugin
  - Modelio
  - StarUML
  - ArgoUML
  - Umbrello UML Modeller
  - Acceleo
  - UMLdesigner
  - PlantUML

- Online
  - yuml.me
  - GenMyModel
  - visual-paradigm
  - MS Visio
  - Creately
  - Draw.io



UML Cheat Sheets

- [UML RefCard from Dzone]
  - reference card, covers all major diagrams
  - Uploaded on D2L
- [Microsoft Model References and Guidelines]
- Allen Holub's Cheat sheet
  - https://holub.com/uml/
  - PDF version on D2L



#112

Get More Refcardz! Visit refcardz.com

www.dzone.com

DZone Refcardz

CONTENTS INCLUDE:

- About UML
- Structural Diagrams
- Behavioral Diagrams
- Interaction Diagrams
- Hot Tips and more...

Getting Started with UML

By James Sugrue

ABOUT UML

The Unified Modeling Language is a set of rules and notations for the specification of a software system, managed and created by the Object Management Group. The notation provides a set of graphical elements to model the parts of the system.

This Refcard outlines the key elements of UML to provide you with a useful desktop reference when designing software.

Hot Tip

UML Tools

There are a number of UML tools available, both commercial and open source, to help you document your designs. Standalone tools, plug-ins and UML editors are available for most IDEs.

Diagram Types

UML 2 is composed of 13 different types of diagrams as defined by the specification in the following taxonomy.

STRUCTURAL DIAGRAMS

Class Diagrams

Class diagrams describe the static structure of the classes in your system and illustrate attributes, operations and relationships between the classes.

Modeling Classes

Hot Tip

Interfaces

Interface names and operations are usually represented in italics.

Relationship	Description
Dependency "...uses a..."	A weak, usually transient, relationship that illustrates that a class uses another class at some point.
Association "...has a..."	Stronger than dependency, the solid line relationship indicates that the class retains a reference to another class over time.
Aggregation "...owns a..."	More specific than association, this indicates that a class is a container or collection of other classes. The contained classes do not have a life cycle dependency on the container, so when the container is destroyed, the contents are not. This is depicted using a hollow diamond.

ClassA ClassB

Figure 2: ClassA has dependency on ClassB

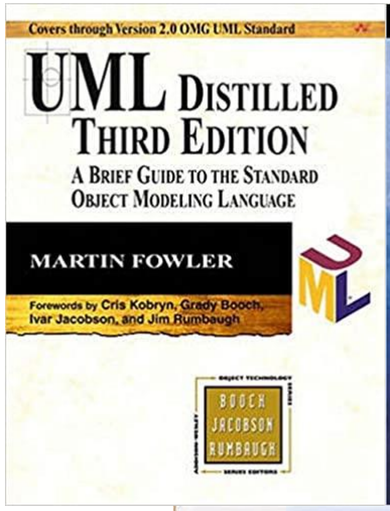
ClassA ClassB

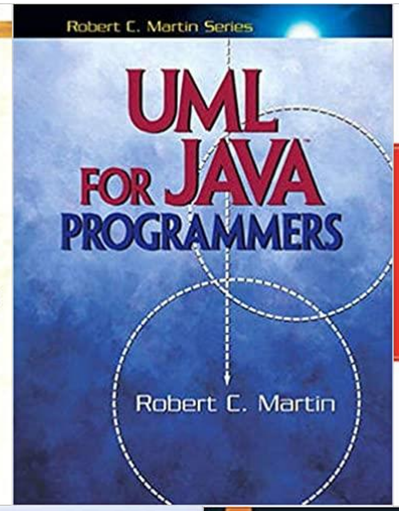
Figure 3: ClassA associated with ClassB

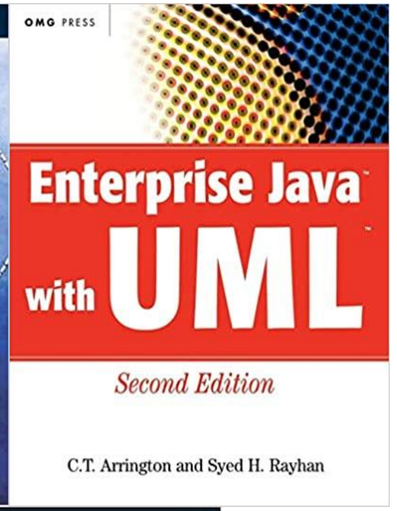
Company Employee

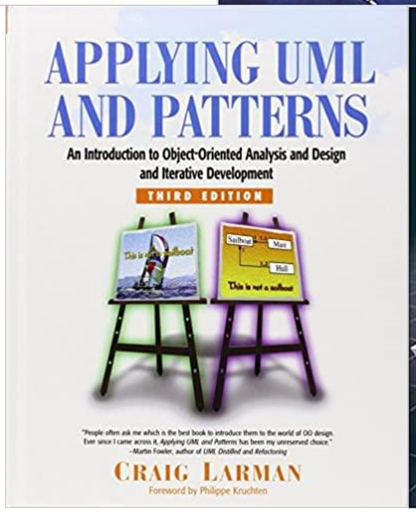
UML Books

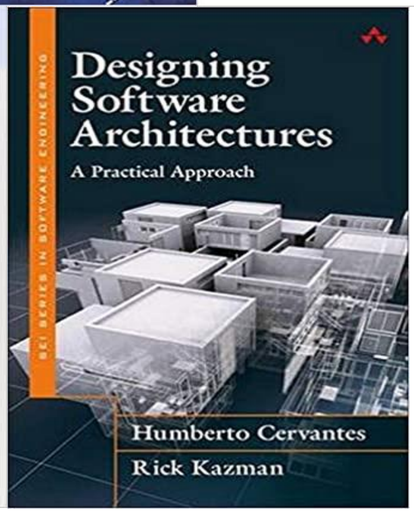
- UML Distilled: A Brief Guide to the Standard Object Modeling Language 3rd Edition (Link)
- UML for Java Programmers (Link)
- Enterprise Java with UML (Link)
- Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (Link)
- Designing Software Architectures: A Practical Approach (Link)

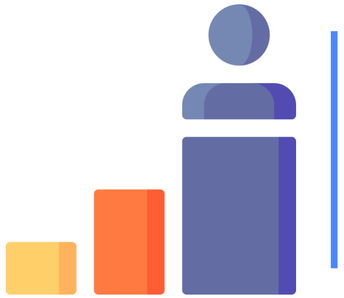












Class Activity

Get Ready!

Class Activity - Week 5.2

Get ready to compete!

Which of the following is FALSE about abstract classes in Java

Which of the following is FALSE about abstract classes in Java

If we derive an abstract class and do not implement all the abstract methods, then the derived class should also be marked as abstract using 'abstract' keyword

Abstract classes can have constructors

A class can be made abstract without any abstract method

A class can inherit from multiple abstract classes.

Total Results: 0

Which of the following is FALSE about abstract classes in Java

If we derive an abstract class and do not implement all the abstract methods, then the derived class should also be marked as abstract using 'abstract' keyword

Abstract classes can have constructors

A class can be made abstract without any abstract method

A class can inherit from multiple abstract classes.

Leaderboard

Which statements are True?

Which statements are True?

```
SE350 Quiz Week 5.1

1) An interface can contain following type of members.
....public, static, final fields (i.e., constants)
....default and static methods with bodies

2) An instance of interface can be created.

3) A class can implement multiple interfaces.

4) Many classes can implement the same interface.
```

1, 2, 3 and 4

1, 3 and 4

1, 2 and 4

2, 3 and 4

Total Results: 0

Which statements are True?

```
SE350 Quiz Week 5.1

1) An interface can contain following type of members.
....public, static, final fields (i.e., constants)
....default and static methods with bodies

2) An instance of interface can be created.

3) A class can implement multiple interfaces.

4) Many classes can implement the same interface.
```

1, 2, 3 and 4

1, 3 and 4

1, 2 and 4

2, 3 and 4

Leaderboard

Predict the output of the following program.

Predict the output of the following program.

```
abstract class demo {
    public int a;
    demo() {
        a = 10;
    }
    abstract public void set();
    abstract final public void get();
}

class Test extends demo
{
    public void set(int a) {
        this.a = a;
    }

    final public void get() {
        System.out.println("a = " + a);
    }

    public static void main(String[] args){
        Test obj = new Test();
        obj.set(20);
        obj.get();
    }
}
```

a = 10

a = 20

Compilation error

Total Results: 0

Predict the output of the following program.

```
abstract class demo {
    public int a;
    demo() {
        a = 10;
    }
    abstract public void set();
    abstract final public void get();
}

class Test extends demo
{
    public void set(int a) {
        this.a = a;
    }

    final public void get() {
        System.out.println("a = " + a);
    }

    public static void main(String[] args){
        Test obj = new Test();
        obj.set(20);
        obj.get();
    }
}
```

a = 10

a = 20

Compilation error

Leaderboard

Which of the following statement(s) with regard to an abstract class in JAVA is/are TRUE ?

Which of the following statement(s) with regard to an abstract class in JAVA is/are TRUE ?

```
[1] An abstract class is one that is not used to create objects.
[2] An abstract class is designed only to act as a base class to be inherited
by other classes.
```

- Both [1] and [2]
- Only [1]
- Only [2]
- Neither [1] nor [2]

Total Results: 0

Which of the following statement(s) with regard to an abstract class in JAVA is/are TRUE ?

```
[1] An abstract class is one that is not used to create objects.
[2] An abstract class is designed only to act as a base class to be inherited
by other classes.
```

Both [1] and [2]

Only [1]

Only [2]

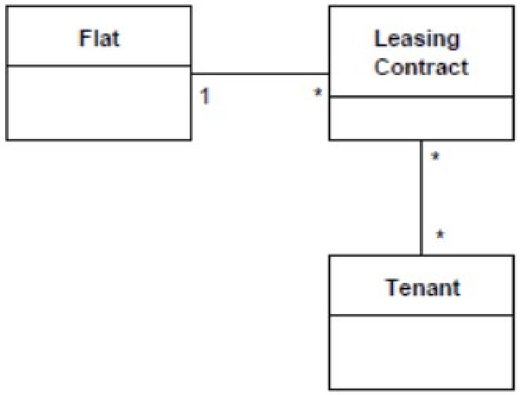
Neither [1] nor [2]

Leaderboard

You are given the following clipping of a UML class diagram. Which of the following statements are true?

You are given the following clipping of a UML class diagram.

Which of the following statements are true?



- ☐ One flat can be rented by multiple tenants with different leasing contracts.
- ☐ One flat can be rented by multiple tenants with the same leasing contract.
- ☐ If a tenant dies (i.e. the tenant-object is deleted), all leasing contracts of the tenant are deleted.
- ☐ One tenant can rent the same flat multiple times with different leasing contracts.

1 and 2 and 4

1 and 2 and 3

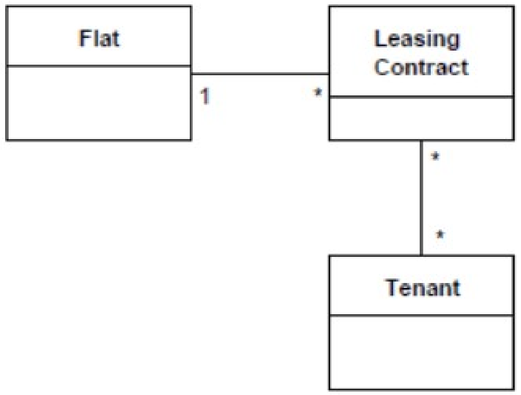
2 and 3

2 and 3 and 4

Total Results: 0

You are given the following clipping of a UML class diagram.

Which of the following statements are true?



- ☐ One flat can be rented by multiple tenants with different leasing contracts.
- ☐ One flat can be rented by multiple tenants with the same leasing contract.
- ☐ If a tenant dies (i.e. the tenant-object is deleted), all leasing contracts of the tenant are deleted.
- ☐ One tenant can rent the same flat multiple times with different leasing contracts.

1 and 2 and 4

1 and 2 and 3

2 and 3

2 and 3 and 4

Leaderboard

A generalization relationship between a subclass and a superclass has the following properties:

A generalization relationship between a subclass and a superclass has the following properties:

A superclass must not be abstract.

A subclass can only inherit from one superclass.

The subclass inherits the properties of the superclass.

The subclass must not have more attributes than the superclass.

Total Results: 0

A generalization relationship between a subclass and a superclass has the following properties:

A superclass must not be abstract.

A subclass can only inherit from one superclass.

The subclass inherits the properties of the superclass.

The subclass must not have more attributes than the superclass.

Leaderboard

Associations:

Associations:

- [ ] 1- may have navigable and non-navigable association directions.
- [ ] 2- have to be identified by a unique association name.
- [ ] 3- show possible relations between instances of classes.
- [ ] 4- may have multiplicities to indicate to how many instances of a class the object is associated to.

1 and 2 and 3

1 and 2 and 4

1 and 3 and 4

2 and 3 and 4

Total Results: 0

Associations:

- [ ] 1- may have navigable and non-navigable association directions.
- [ ] 2- have to be identified by a unique association name.
- [ ] 3- show possible relations between instances of classes.
- [ ] 4- may have multiplicities to indicate to how many instances of a class the object is associated to.

1 and 2 and 3

1 and 2 and 4

1 and 3 and 4

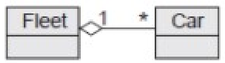
2 and 3 and 4

Leaderboard

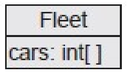
How do you model the following situation with a UML class diagram:

How do you model the following situation with a UML class diagram:

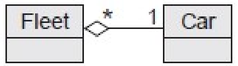
The car fleet of a car rental contains multiple cars, one car belongs to exactly one car fleet.



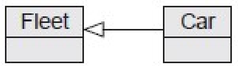
1



2



3



4

1

2

3

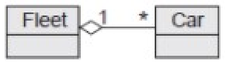
4

Total Results: 0

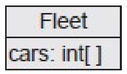
How do you model the following situation with a UML class diagram:

The car fleet of a car rental contains multiple cars, one car belongs to exactly one car fleet.

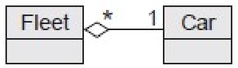
1



2



3



4

1

2

3

4

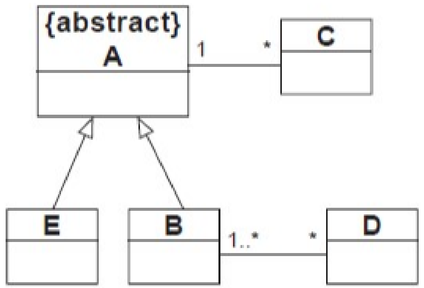
Powered by Poll Everywhere

Start the presentation to see live content. For screen share software, share the entire screen. Get help at **pollev.com/app**

Leaderboard

You are given the following clipping of a UML class diagram. Which of the following statements are true?

You are given the following clipping of a UML class diagram. Which of the following statements are true?



- ☐ Direct instances of A exist.
- ☐ One object of D is associated with at least one object of B.
- ☐ One object of B is associated with 1..* objects of D.
- ☐ One object of a subclass of A is associated with * objects of C.

1 and 2

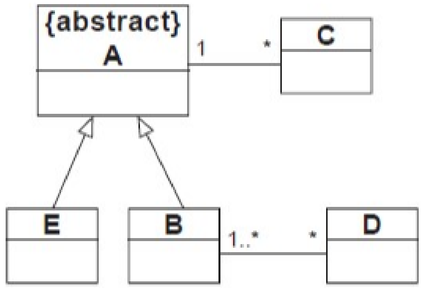
2 and 3

3 and 4

2 and 4

Total Results: 0

You are given the following clipping of a UML class diagram. Which of the following statements are true?



- ☐ Direct instances of A exist.
- ☐ One object of D is associated with at least one object of B.
- ☐ One object of B is associated with 1..* objects of D.
- ☐ One object of a subclass of A is associated with * objects of C.

1 and 2

2 and 3

3 and 4

2 and 4

Leaderboard

Any Question

??????????????

How do you feel about the course?

Please Send Your Question or Feedback...

Top **New**