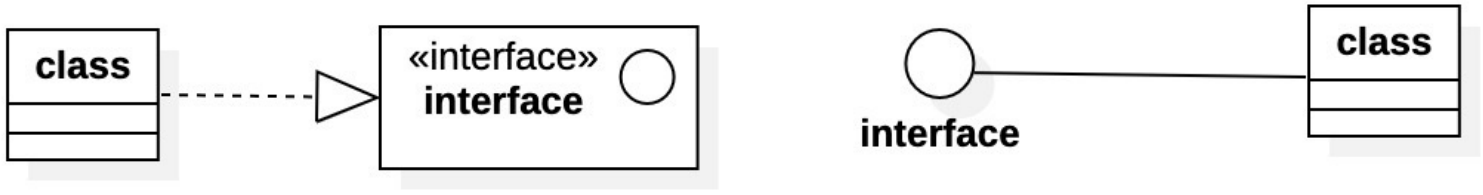
What is Realization?

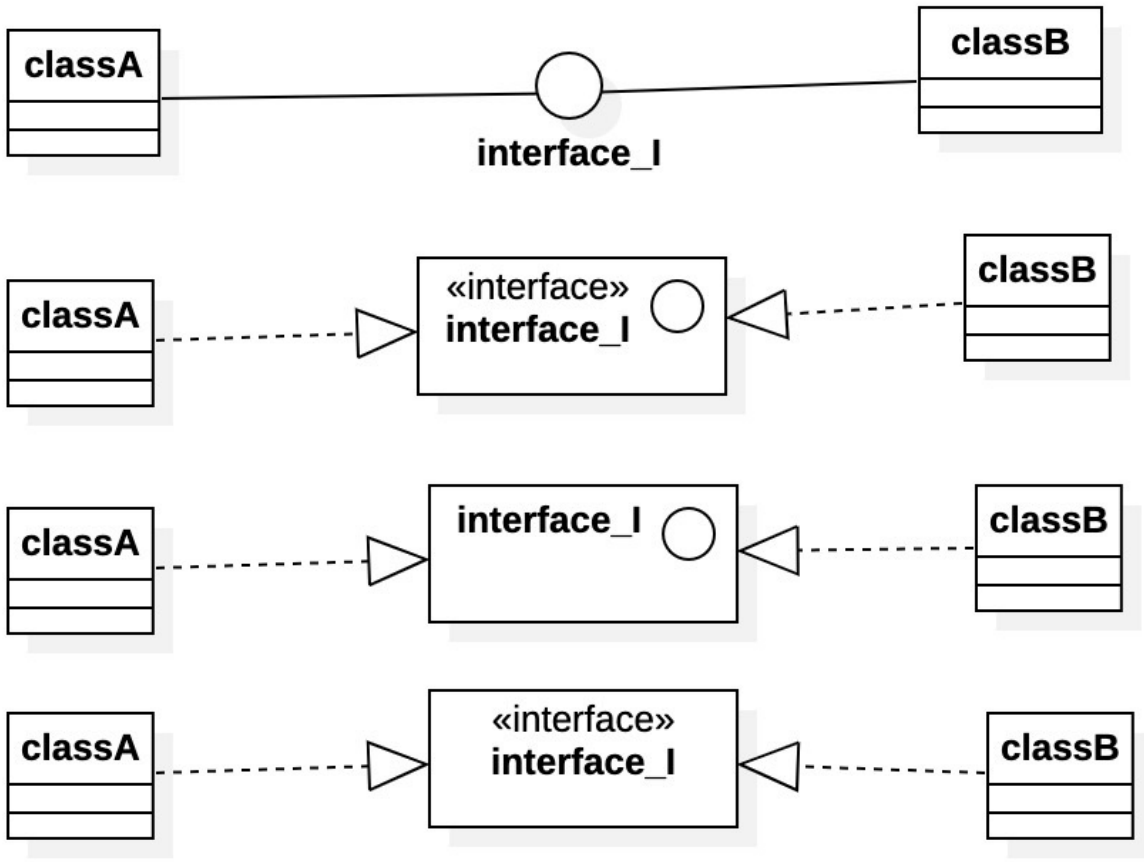
Realization

- In UML modeling, a realization relationship is a relationship between two model elements, in which one model element (the client) realizes (accepts to implement or execute) the behavior that the other model element (the supplier) specifies.
- The supplier presents the outside view of its requirements ,and the client will implement the internal details (inside view).



StarUML Notation Examples

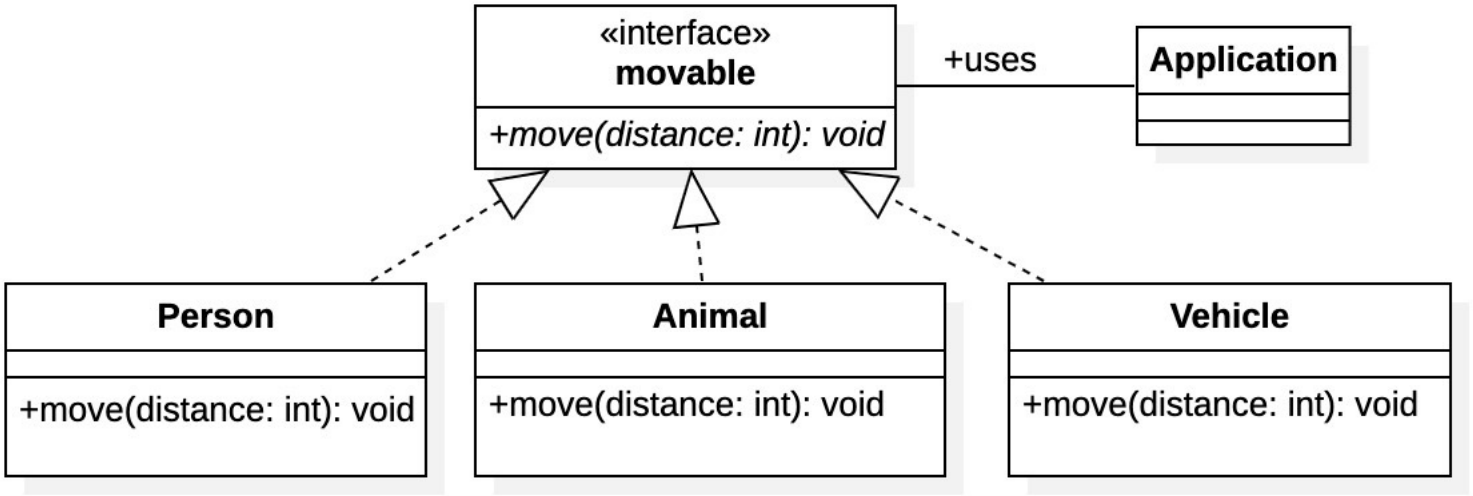
- Different venders may suggest several examples of Realization notation. Here is a few examples suggested by StarUML, the tool that we will use in this course:



Implementation of Realization in Java

Realization and Polymorphism

- Realization helps implementation of polymorphism among classes (not necessarily classes that belong to the same hierarchy). Example:



- If the definition of the interface classes are properly implemented, The following code in Java works perfectly:

Java Code

```
interface movable { …}
class Person implements movable{ …}
class Animal implements movable{ …}
class Vehicle implements movable{ …}
public class Application {
    public void run() {
    Movable p = new Person();
    Movable a = new Animal();
    Movable v = new Vehicle();
    move(p);
    move(a);
    move(v);
    }
    public void move(Movable m) {m.move(100);}
}
```

Notice how Movable represents object of classes Person, Animal, and Vehicle

Notice how polymorphism is implemented in method move.

Java Realization

- Interfaces give Java some of the power of multiple inheritance, however, there is no code reuse, since each class must re-implement the methods.
- A reference of an interface type can refer to the instances of any classes that implement that interface.

Example of Realization in Java

```
interface A
{
   public void fun();
}
```

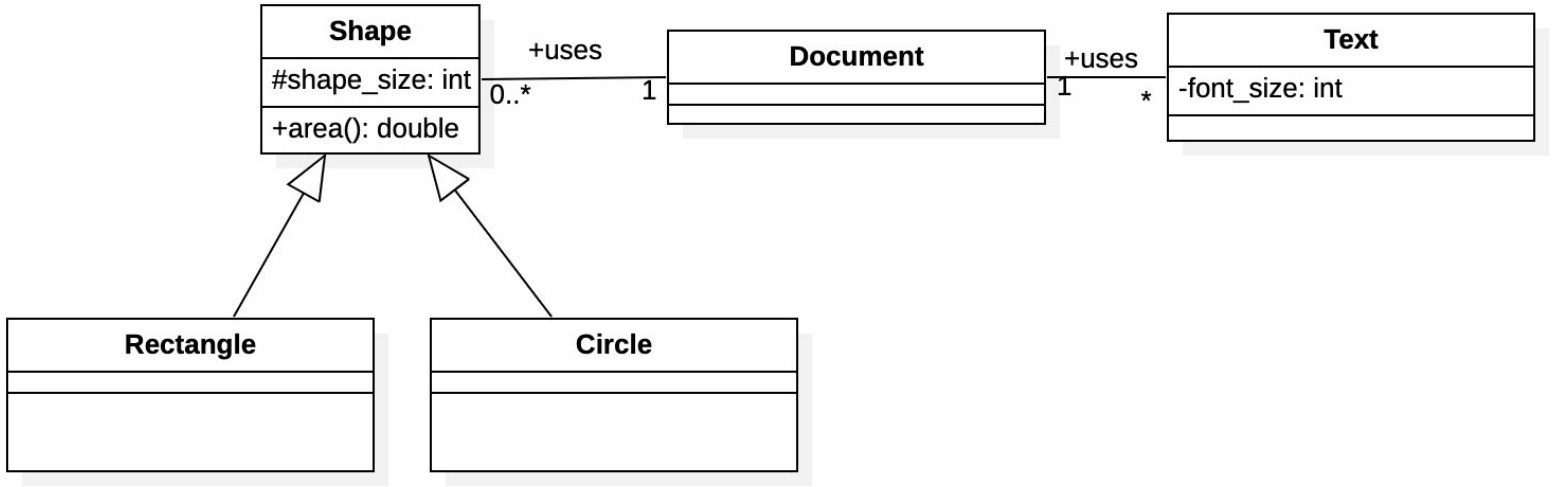
```
interface B
{
   public void bar();
}
```

```
class Shape implements A, B {
   …
}
```

Class Exercise

Class Exercise

- Assume a word processing software application, needs a few class such as **Text, Shape, Document**, **etc**. Where, class **Document** uses the other two classes.
- Obviously, there are some common behaviors among objects of class Shape and Test.
- **Class Discussion!**
- **Questions to be asked:**
  - **How should we design this application.**
    - **How can we make the future maintenance easier**
    - **How can we make the implementation of polymorphism easier/possible**



The answer will be discussed during the lecture sessions

Another Class Discussion:

- What about Implementation of realization in C++?
  - How can realization be implemented, while C++ doesn't have a feature such as 'Java interface'?
- Answer will be discussed during the lectures