Introduction to Systems Programming

Inheritance, abstract classes, and inheritance-based polymorphism

2025 - Simón Gutiérrez Brida *(Based on material by Dr. Nazareno Aguirre)*

Extending classes

In Java, as in most OOP languages, any class will be a subclass of another. In Java we have a root class which is the exception of this rule. Object is not a subclass of any other class, but all other classes are a subclass of Object.

Any class A that is a subclass of B will have access to:

- Public fields, methods, and constructors.
- Package fields, methods, and constructors.
- Protected fields, methods, and constructors.

Any class A that is a subclass of B can redefine/overwrite:

- Methods, but access permissions must remain equal or greater.
  - E.g.: a protected method in B can be made protected or public in A.

Extending classes

If class B is a subclass of class A, then B can be used in place of A. The opposite is not possible.

Continuing from the previous example, any constructor of B must call a constructor of A as its first statement. This rule might not always be visible because of empty constructors, if A has an empty constructor, then it will be used by default if a constructor in B doesn't call it explicitly.

Superclass constructor call

- Subclass constructors must always contain a 'super' call.

- If none is written, the compiler inserts one (without parameters).
  - works only, if the superclass has a constructor without parameters

- Must be the first statement in the subclass constructor

Extending classes

“B extends A” can be seen as “B is an A”.

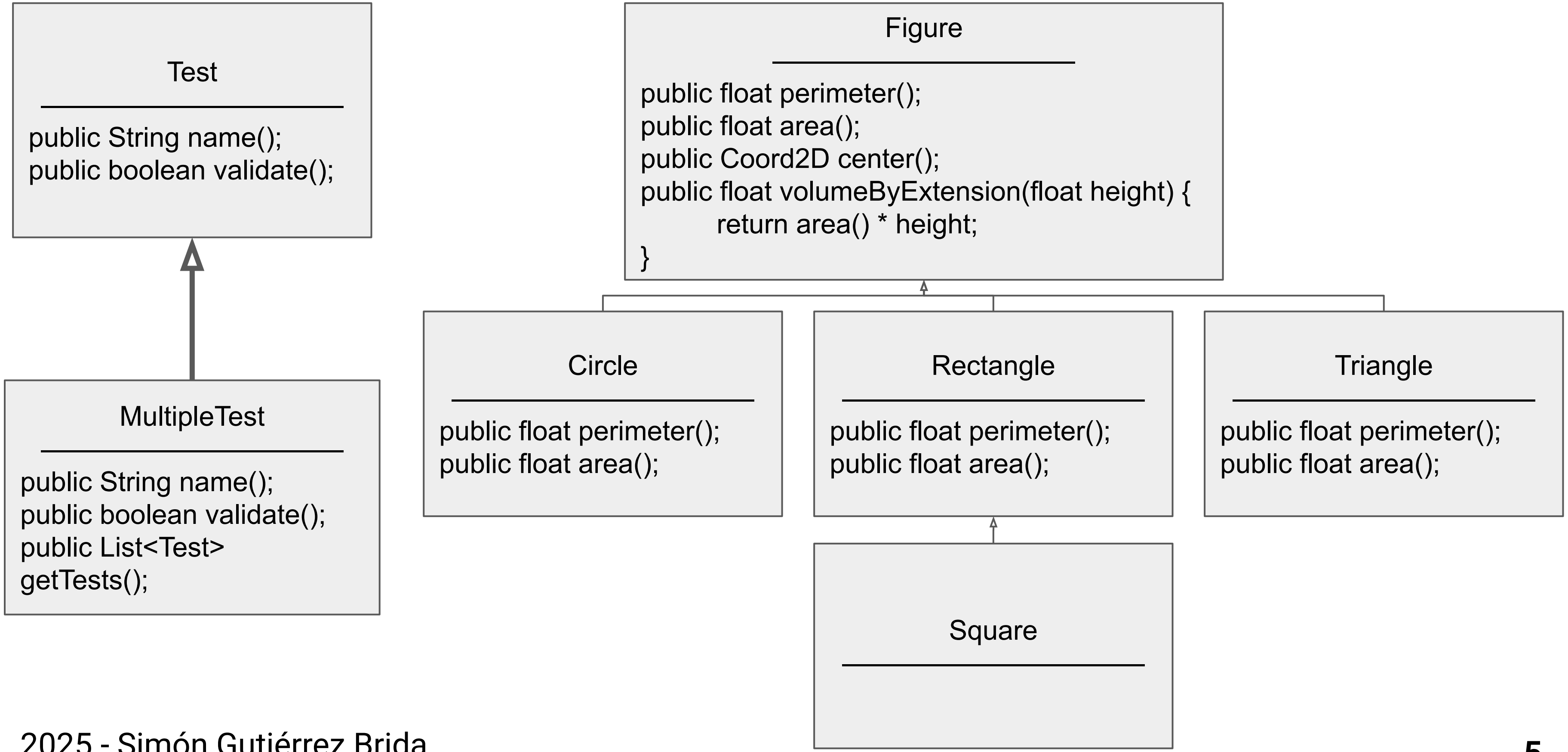
Inheritance also applies to interfaces:

```
public class LinkedList<T> implements List<T>
```

Also means “LinkedList is a List”

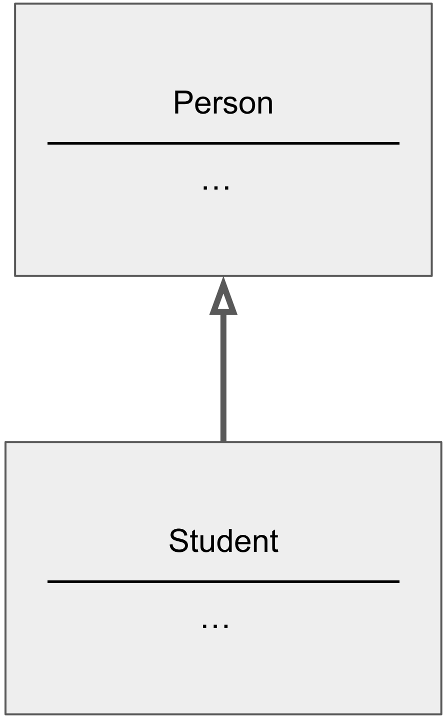
Extending classes in Java

The good, the bad, and the ugly


Test
public String name();
public boolean validate();
MultipleTest
public String name();
public boolean validate();
public List<Test> getTests();
Figure
public float perimeter();
public float area();
public Coord2D center();
public float volumeByExtension(float height) {
return area() * height;
}
Circle
public float perimeter();
public float area();
Rectangle
public float perimeter();
public float area();
Triangle
public float perimeter();
public float area();
Square

Extending classes in Java

The good, <u>the bad</u>, and the ugly

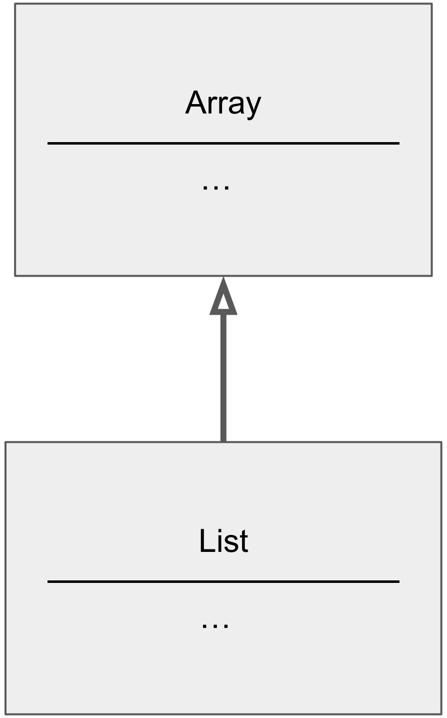


Although not technically bad or incorrect, it may be better to have a Student class with a Person field inside; even better if Person would be an interface so Student doesn't depend at all on its implementation.

Nevertheless, this is a very simple example.

Extending classes in Java

The good, the bad, and the ugly



Considering an Array class, and a List class (not to be confused with any specific class in Java).

Although there are similar behaviours, they are very different conceptually.

Extending classes in Java

What can we extend

- Interfaces (extending an interface is not the same as implementing one!).
- Classes.
- Abstract classes.

Extending classes in Java

What can we extend

- Interfaces (extending an interface is not the same as implementing one!).
- Classes.
- Abstract classes.

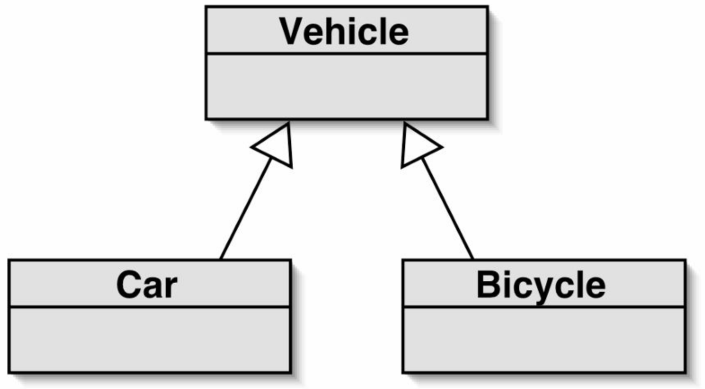
An abstract class in a partially implemented class, it cannot have instances, we have seen an example before in class Figure.

Subclasses and subtyping

- Classes define types.
- Subclasses define subtypes.
- Objects of subclasses can be used where objects of supertypes are required.
  - This is called substitution.

Subclasses and assignment

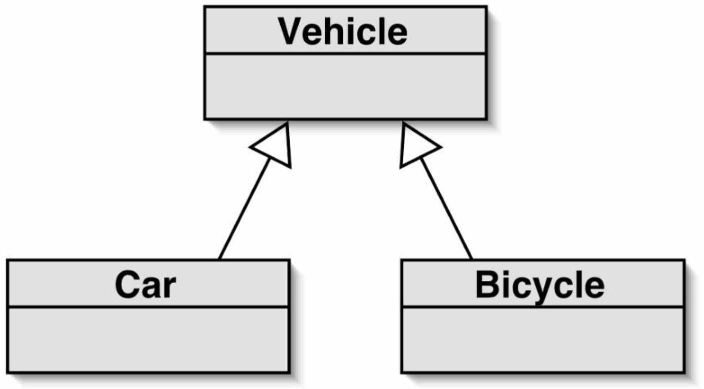
Static vs Dynamic Types



```
Vehicle v1 = new Vehicle();
Vehicle v2 = new Car();
Vehicle v3 = new Bicycle();
```

Subclasses and assignment

Static vs Dynamic Types

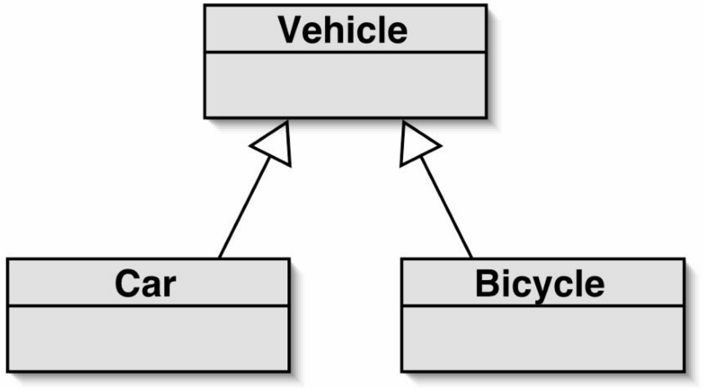


Static types, checked during compilation.

```
Vehicle v1 = new Vehicle();
Vehicle v2 = new Car();
Vehicle v3 = new Bicycle();
```

Subclasses and assignment

Static vs Dynamic Types



Dynamic types, given during runtime.

```
Vehicle v1 = new Vehicle();
Vehicle v2 = new Car();
Vehicle v3 = new Bicycle();
```

Subclasses and parameter passing

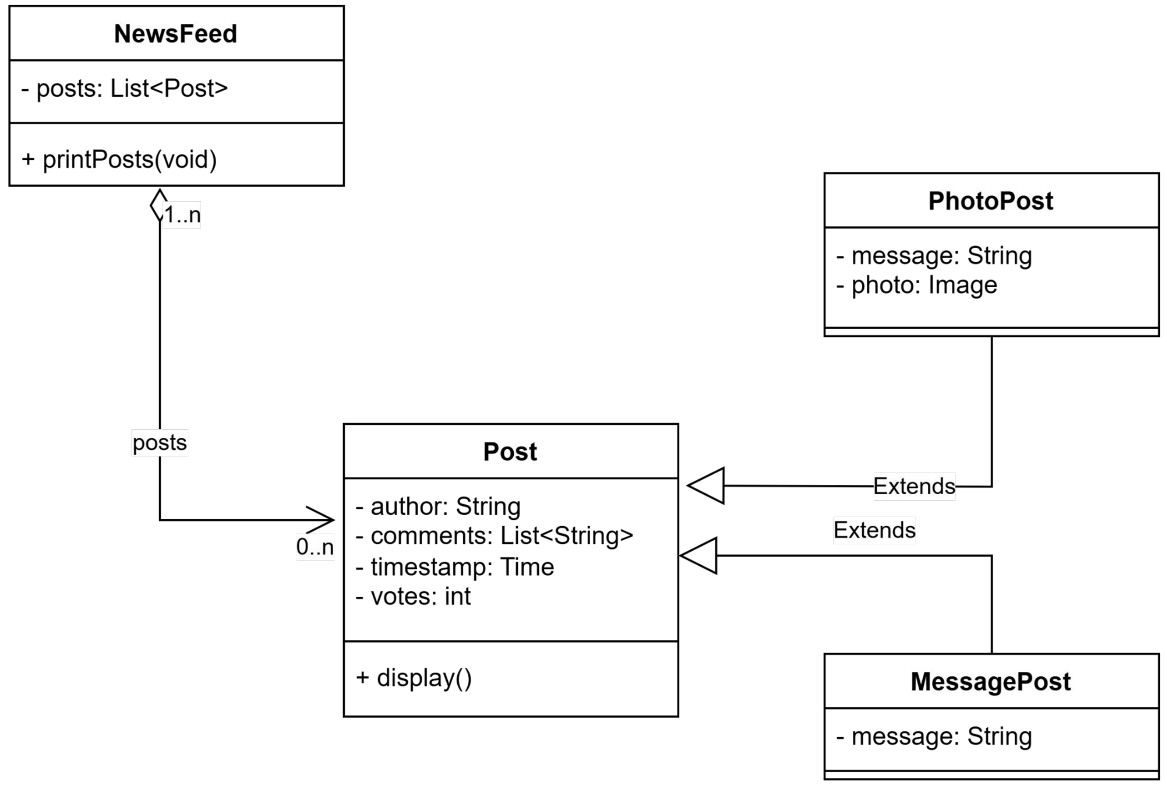
```
public void addArea(Figure figure)
{
  this.area += figure.area();
}

Figure circle = new Circle(3.5f);
Figure rectangle = new Rectangle(4.0f, 2.0f);
Square square = new Square(2.0f);
addArea(circle);
addArea(rectangle);
addArea(square);
```

A question

If a variable of class/type A can be assigned any value of type A or a subclass of A; what types (non-primitive) can be assigned to an Object variable?

Inheritance hierarchy



Inheritance hierarchy

```
Leonardo da Vinci
Had a great idea this morning.
But now I forgot what it was. Something to do with flying ...
40 seconds ago - 2 people like this.
  No comments.

Alexander Graham Bell
[experiment.jpg]
I think I might call this thing 'telephone'.
12 minutes ago  -  4 people like this.
  No comments.
```

What we want

```
Leonardo da Vinci
40 seconds ago  -  2 people like this.
  No comments.

Alexander Graham Bell
12 minutes ago  -  4 people like this.
  No comments.
```

What we have

Possible solutions

- Redefine method display on each Post subtype/subclass.
- Make display abstract in Post, so it must be defined in each subclass.
- Define method display as:

```
public void display()
{
    //show author
    //call displayBody()
    //show timestamp and votes
}
```

  - And make displayBody an abstract method

To discuss

Given class A with methods m1, m2, and m3; classes B and C as subclasses of A overriding methods m1, and m2; and class D as a subclass of B overriding method m2.

Given the statements

```
A x = randomInstanceOfA();
x.m2();
x.m3();
```

With method “randomInstanceOfA” giving an instance of A, B, C, or D. How do we know which method “m2” and “m3” is called?

Abstract Data Types

Lists

A List is a linearly organized collection of values, its main operations are:

- Creation
- Insertion/Deletion/Retrieval
- Properties about a list: empty, size, contains.

Abstract Data Types

Sets

A Set is an unordered collection of different elements, its main operations are:

- Creation
- Insertion/Deletion
- Union/Intersection
- Properties about a set: empty, size, contains, is a sub set.

Abstract Data Types

Stacks

A Stack is a linearly organized collection of values (similar to a list), it’s a FILO collection (First In, Last Out), its main operations are:

- Creation
- Push/Pop
- Properties about a stack: empty, size

Abstract Data Types

Queues

A Queue is a linearly organized collection of values (similar to a list), it’s a FIFO collection (First In, First Out), its main operations are:

- Creation
- Enqueue/Dequeue
- Properties about a queue: empty, size

Demo and discussion

Making our own ADT and implementation

The implementation of an ADT might not be different than the implementation of another. A LinkedList can be used to implement Lists, Sets, Collections (*), Queues, Stacks, Deques, etc.

Is this a good approach? What about class invariants?

** Consider methods map, all, and any.*