Java's "Levels" for class relations / code reuse

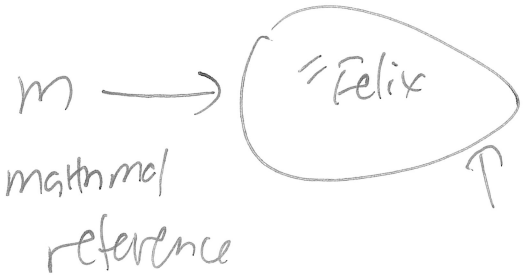
1) Interfaces [guarantee of methods in an implementing class]

2) Abstract Class [groupings of potentially different types of things that all share some commonalities]
   - define method signature (just like interface)
   - ~~wri~~ specify instance variables
   - write some methods

3) Regular Class...

I can have references of types that are Interfaces and Abstract Classes, but all actual objects must be of regular classes.

```
Mammal m = new Cat("Felix");
```



Requirement: Object is a more "specific" type than the reference

"is-a" relationship.

Inheritance: Villain extends Character

When class B inherits from class A:

(1) We get all of class A's code for free.

(2) We can ADD new methods and instance variable special to us.

(3) Refine a method from class A, if we don't like it. (OVERRIDING)

Class A - Superclass

vars	methods
a	p
b	q
c	r
	s

Class B extends class A (subclass)

var	methods
d	q
	t
	v

A class can only inherit/extend one other class.

class D extends class C
class C extends class B
class B extends class A