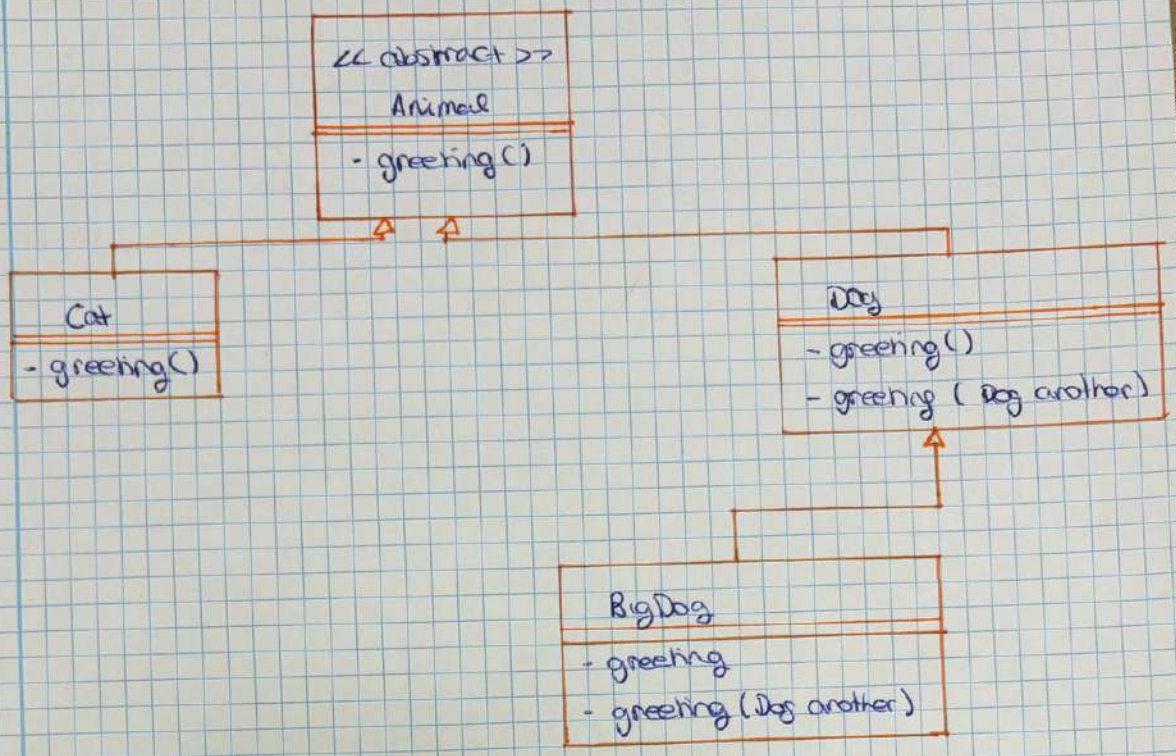


```
Cat cat1 = new Cat();
cat1.greeting();
Dog dog1 = new Dog();
dog1.greeting();
BigDog bigDog1 = new BigDog
bigDog1.greeting();
```

Meow! / Woof! / Woow!

Se crea un objeto y se llama al método greeting que imprime el sonido del animal.

```
Animal animal1 = new Cat();
animal1.greeting();
Animal animal2 = new Dog();
animal2.greeting();
Animal animal3 = new BigDog();
animal3.greeting();
```

Meow! / Woof! / Woow!

animal 1,2,3 es de tipo Animal, apunta a una clase. Como llama a greeting() e imprimente cada llamada a su version greeting().

```
Animal animal4 = new Animal();
```

Animal es abstracto no se instancia → ERROR

```
Dog dog2 = (Dog)animal2;
BigDog bigDog2 = (BigDog)animal3;
Dog dog3 = (Dog)animal3;
Cat cat2 = (Cat)animal2;
```

animal 2 y 3 eran Dog y BigDog por lo que el "downcasting" funciona

Animal2 apunta a Dog y un Dog no puede convertirse en Cat → ERROR

```
dog2.greeting(dog3);  }   dog y dog ⇒ forma Dog(anotherDog) ⇒ "Woof"
dog3.greeting(dog2);  }   dog y dog ⇒ forma Dog(anotherDog) ⇒ "Woof"
dog2.greeting(bigDog2);}  dog y bigDog (es un dog) ⇒ Dog(anotherDog) ⇒ "Woof"
bigDog2.greeting(dog2);}  BigDog.sobreescribe greeting(anotherDog) y da "Wooouuu"
bigDog2.greeting(bigDog1);} BigDog y BigDog ⇒ BigDog.greeting(dog anotherDog) ⇒
                              "Wooouuu"
```