MAKE SCHOOL

Polymorphism

What we're going to learn

- What polymorphism means and how it relates to inheritance and overriding
- How to implement and use polymorphism

Check In

MAKE SCHOOL

Share your favorite easy 30 min recipe!

Students, write your response!

Recap of OOP Part 2: Inheritance

MAKE SCHOOL

- **Inheritance** is a mechanism of objects that allows one to derive classes based off another class and share properties and methods among class hierarchies.
- Objects can inherit methods and properties from other objects, allowing for modular and D.R.Y. object characterization.
- A **superclass** is a higher-hierarchy base class that has basic attributes to derive from.
- A **subclass** is a lower-hierarchy class that derives attributes from another base class.

MAKE SCHOOL

Polymorphism
What does this word mean to you?

Students, write your response!

Polymorphism

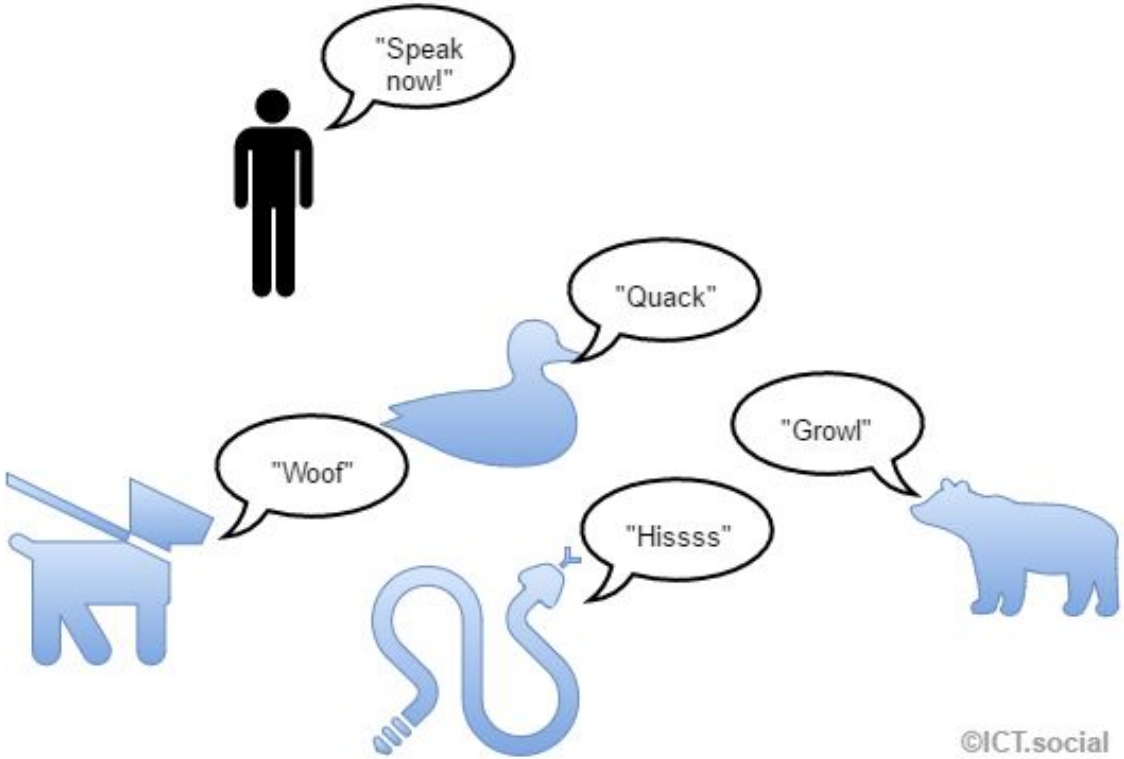
MAKE SCHOOL

- **Polymorphism** is a mechanism by which a single object can take on many different forms.

- Objects demonstrating *polymorphism* often have multiple different instances with different methods/properties sharing the same name.

- Polymorphism occurs through the mechanic of *inheritance*.

Polymorphism



Pac-Man and Polymorphism

MAKE SCHOOL

Consider our **Pac-Man** class structure.

Let's extend our objects to include an example of polymorphism.

The `Eatable()` class has an `.eat()` method, but its implementation can differ between the superclass and each of its subclasses (e.g., Pac-Man can always eat pellets and fruits, but can only eat ghosts when they're scared).

This allows us to access different behaviors (implementations of the method), **even though the method name is the same**.

MAKE SCHOOL

Let’s look at an example!

Students browse: repl.it/@MakeSchool/animalpolymorphismexample?lite=true

MAKE SCHOOL

Now it's your turn!

Create another subclass of Animal with at least two methods one of them being an inherited method!

Be prepared to share with the class!

Students browse: repl.it/@MakeSchool/animalpolymorphismexample?lite=true

Consider...

We're calling the same function 3 times in the previous code snippet. How can we improve this code to be less redundant?

```
John = Person("John")
Jane = Student("Jane")
Jamie = Teacher("Jamie")

John.say_hello()
Jane.say_hello()
Jamie.say_hello()
```

Students, write your response!

Functionalize Our Classes

MAKE SCHOOL

We can modify and clean up our code to showcase how **polymorphism** and **inheritance** allows for more *D.R.Y.* and less redundant code snippets.

When we have objects of different classes that each inherit from the same superclass, we can call a method on each object and see different behavior.

```
John = Person("John")
Jane = Student("Jane")
Jamie = Teacher("Jamie")

people = [John, Jane, Jamie]

for person in people:
    person.say_hello()
```

Polymorphism in Functions

We can write functions that also take advantage of polymorphism!

In our function we don’t need to care about which object is being passed as a parameter, all we assume is that it has some sort of speak() method

Students browse: repl.it/@MakeSchool/animalpolymorphismexample?lite=true

Summary

MAKE SCHOOL

- **Objects** are structured collections of data in the form of variables (**properties**) and functions that act on those variables (**methods**)
- **Classes** are blueprint-like structures used to **create object instances**.
- To use a class, access its properties and call its methods, one must **instantiate an object** that represents an instance of that class.
- **Subclasses** will **inherit** all properties and methods from its **superclass**, which allows code using OOP to avoid duplication and remain DRY.
- Classes can have **polymorphic** methods that have the same name but different behaviors (implementations) based on the object's actual class.

MAKE SCHOOL

Shout outs

Students, write your response!

Assignments

MAKE SCHOOL

- Flower Garden (Sept 4th)
- Quiz 1 (Sept 8th)
- Superhero Team Dueler Tutorial (Sept 11th)