Methods are the behaviors of an object that we have glanced over before. Think of them as the actions an object can take or do. They are separated into instance methods and static methods. With instance methods there can also be mutators and accessors or getters and setters. So there are different ways that an object can have a method. I will describe and show these four methods in this video so you can have an idea of what they are about. As always, have fun with what I show and don't feel you need to completely follow my examples. Just make sure you get the basics down for what you will need to use.

I will start with static methods, as I only want to get a basic understanding of them for you. Static methods are not used much in object-oriented programming. They are procedural-style instead and are made by using this design of the method, public static void example. The public means that the method can be used by the entire program. Void is used because the method will not return anything. Now if you were confused by procedural-style, it is only a top-down approach going straight from the start of the program to the end of it.

Now lets get into instance methods, as they are the methods that we will be using for object-oriented programs. These are methods that must have an object created for it to work and will work on that object. These methods will not use static though, as if they do, the method would have to be copied in other objects and will create undo redundancy. So, when we take out the static of the method we will then call the method using what is known as dot notation. This will take the form of using the object then putting a dot and then writing the method that we wish to use.

Lets put this in practice, shall we. Lets make a new object and call it Tom. Tom will be our new cat. Now lets head over to our cat class. We still have the fields that make up what a cat has. I will put in a couple of methods so that we can start using these fields and Tom will be able to do something. I will make a method and call it pet. In it I will write a print statement asking the question "What happens when I pet Tom?", add a plus symbol, and write with an escape sequence for new line forward slash n "He purrs", putting another plus symbol and adding the variable purr at the end. Now I will go to my object named Tom and under it type tom.pet and run the program. As you see it prints out that he will purr when petted.

This is an example of an accessor or getter method of Java. It will access an object's field and give what the field states. An accessor will not change the state of an object as what a mutator will do. A mutator will modify the state of an object. So it would be having the cat as happy in one moment, then using a method that was made before, change that happy state to one of anger or scared. We can make a couple of String variables here, write another method here, and in the Tom object, we write what we change the state to with the method.

With these methods, we can do a lot more with our objects and make them do things that we wish them to do. These methods can now make Tom purr with happiness, hiss in anger, and screech in fright. You can add some more methods to make Tom, or whatever name you wish to give your cat, and have the cat do more things than what I have in here. Get your imagination going and make the cat your own. Have it call for food, say hi to people, or eat fish and drink water. In fact, I might even do that to my program. Well good luck, and experiment. I believe in you.