When we talk about creating a class in Java, we're essentially defining a blueprint or a template for objects. It's like creating a recipe for making something. A class contains variables (also known as attributes) and methods that define the behavior and characteristics of the objects that will be created from it. Here's an example of a simple class called "Person":

```java

public class Person {

String name;

int age;

public void sayHello() {

System.out.println("Hello, my name is " + name + " and I am " + age + " years old.");

}

}

```

In this example, the "Person" class has two attributes: "name" and "age". It also has a method called "sayHello()" that prints out a greeting with the person's name and age. Now, let's move on to creating an object. When we create an object, we're essentially creating an instance of a class. It's like making a specific object based on the blueprint we defined with the class. Here's an example of creating an object from the "Person" class:

```java

Person person1 = new Person();

person1.name = "John";

person1.age = 25;

person1.sayHello();

```

In this example, we create a new object called "person1" from the "Person" class using the "new" keyword. We then assign values to the attributes "name" and "age" of the object. Finally, we call the "sayHello()" method on the object, which will print out the greeting with the assigned name and age. That's the basic idea behind creating a class and creating an object in Java. The class serves as a blueprint, while the object is an instance created from that blueprint.

In Java, attributes are also known as instance variables or fields. They represent the state or characteristics of an object. To access an attribute, we use dot notation. Let's say we have a class called "Car" with an attribute called "color". Here's an example:

```java

public class Car {

String color;

public void setColor(String newColor) {

color = newColor;

}

public String getColor() {

return color;

}

}

```

In this example, we have a class called "Car" with a "color" attribute. We also have two methods: "setColor()" and "getColor()". The "setColor()" method takes a parameter called "newColor" and sets the value of the "color" attribute to the provided value. The "getColor()" method simply returns the current value of the "color" attribute.

Now, let's create an object of the "Car" class and access and modify its attributes:

```java

Car myCar = new Car();

myCar.setColor("blue");

System.out.println("My car's color is " + myCar.getColor());

```

In this code snippet, we create a new object called "myCar" from the "Car" class. We then use the "setColor()" method to set the color of the car to "blue". Finally, we use the "getColor()" method to retrieve the current color of the car and print it out. bSo, by using methods like "setColor()" and "getColor()", we can access and modify the attributes of an object in Java.