Classes and objects are fundamental concepts in Java. Think of a **class** as a blueprint or a template for creating objects. It defines the properties (variables) and behaviors (methods) that an object of that class will have. An **object** is a specific instance of a class, created from that blueprint.

### Creating a Class

Let's use an example of a Car class. In Eclipse, you can create a new class by right-clicking on your project's src folder, then selecting New -> Class. Name the class Car.

Java

// Car.java  
public class Car {  
 // Properties (fields)  
 String brand;  
 String model;  
 int year;  
   
 // Methods (behaviors)  
 public void startEngine() {  
 System.out.println(brand + " " + model + " engine started.");  
 }  
   
 public void drive() {  
 System.out.println("The " + brand + " is driving.");  
 }  
}

In this Car class, we've defined three properties: brand, model, and year. These are the data that each Car object will store. We've also defined two methods: startEngine() and drive(). These are the actions that a Car object can perform.

### Creating Objects

To use the Car class, you need to create objects from it. You can do this in a separate class with a main method. In Eclipse, create a new class, maybe named Dealership, and add the following code:

Java

// Dealership.java  
public class Dealership {  
 public static void main(String[] args) {  
 // Creating the first Car object  
 Car myCar = new Car();  
 myCar.brand = "Toyota";  
 myCar.model = "Camry";  
 myCar.year = 2022;  
   
 System.out.println("My first car is a " + myCar.brand + " " + myCar.model);  
 myCar.startEngine();  
   
 // Creating another Car object  
 Car newCar = new Car();  
 newCar.brand = "Honda";  
 newCar.model = "Civic";  
 newCar.year = 2024;  
   
 System.out.println("My new car is a " + newCar.brand + " " + newCar.model);  
 newCar.drive();  
 }  
}

Here's a breakdown of what's happening:

1. **Car myCar = new Car();**: This line creates an instance of the Car class and assigns it to a variable named myCar. The new keyword is used to allocate memory for the new object.
2. **myCar.brand = "Toyota";**: We're accessing the brand property of the myCar object and setting its value to "Toyota".
3. **myCar.startEngine();**: We're calling the startEngine() method on the myCar object. This executes the code inside that method.

This process is analogous to building a house from a blueprint. The blueprint (Car class) defines the structure and features, while the actual house (myCar object) is a tangible realization of that blueprint. Each object (myCar, newCar) has its own unique set of property values, but they all share the same methods and overall structure defined by the class.

### Key Takeaways

* **Class**: A blueprint or template. It defines the structure and behavior.
* **Object**: An instance of a class. It's a real-world entity created from the class blueprint.
* **Encapsulation**: Classes bundle data (fields) and methods that operate on that data into a single unit.
* **Instantiation**: The process of creating an object from a class using the new keyword.

This is a basic introduction to classes and objects. As you progress, you'll learn about constructors, inheritance, and other object-oriented programming (OOP) principles that build on these foundational ideas.