# Object Attributes

*by Sophia*

WHAT'S COVERED

In this lesson, you will explore how to set and change values of object attributes and create default values. Specifically, this lesson covers:

1. Setting Values
2. Setting Default Attribute Values

## 1. Setting Values

In the prior tutorial, you created a `User` class with two attributes. It's important to remember that the attributes were declared as private in accordance with the principle of data encapsulation (the values in the attributes should not be directly read or modified from outside of the class; access is provided via appropriate public methods). Once you used the constructor to set the values, you did not make any changes to the attributes. Let's revisit that class:

```
public class User {
  private String userName;
  private String password;

  public User(String userName, String password) {
    this.userName = userName;
    this.password = password;
  }

  // Allow read-only access to user name
  public String getUserName() {
    return userName;
  }

  // There is no direct access to the password.
```

```
  // In real code there would be a method check
  // for a match with the stored password.
}
```

TRY IT

**Directions:** If you haven't already done so, go ahead and add the `User` class to the IDE:

Continuing to use the example from the last tutorial, you created "account" as the instance of `User` class. Here is that full code again from the previous tutorial. Recall that there were two files: User.java (containing the `User` class) and UserExample.java (containing the UserExample class—the **driver class**—with the application's main() method).

Here is the code for the `UserExample` class:

```
class UserExample {
  public static void main(String[] args) {
    User account = new User("sophia", "mypass");
    System.out.println("Account created for " + account.getUserName());
  }
}
```

TRY IT

**Directions:** Run the application (via the driver class).

The output screen would display like this:

```
Account created for Sophia
```

REFLECT

While the `User` class is quite simple, the way that it is compiled as a separate class provides a bit of perspective on how a Java class can be compiled and used in different programs. Just as the driver class, `UserExample`, makes use of the `User` class, other classes could do the same (with appropriate access to the .class file, but that aspect will not be addressed here). Think about how such classes allow building a large program out of many pieces.

Remember, with the new instance `account`, the call to the constructor is passing the arguments of "sophia" as the username and "mypass" as the password. This instance assigns these values to the parameters set in the attributes of the `User` class.

TERM TO KNOW

**Driver Class**
The class in an application that contains the main() method.

# 2. Setting Default Attribute Values

Up to this point, the focus has been on setting values via constructor parameters. However, it is not necessary to pass in the value for every attribute when a new instance of an object is created—for example, if the value is always a set value when an attribute is declared, as in the PeopleCounter class that you have seen previously. In this class, the count attribute is always initialized to 0:

```
public class PeopleCounter {
  private long count = 0;

  public void anotherOne() {
    count++;
    System.out.println("So far " + count);
  }
}
```

Let's expand on the `User` class to see how you can use attributes whose values are given default values. Perhaps now you would want to:

- Determine if the user's account is active.
- Determine when the user joined.

This provides three more attributes that you want to set. However, in all of these cases, there's no need to manually pass these values in, since:

- The account should automatically have the activity flag set to True.
- When a user is created, the created date should simply be set to the current date.

The last attribute deals with time. Anytime that you will do anything with dates and times, you will have to import one of the classes for working with dates provided by Java's standard libraries.


KEY CONCEPT

In a number of previous examples, you imported classes such as `java.util.Scanner` for use in our code. Here you will need to import the datetime module to obtain this attribute's value. Importing classes from the library will be discussed in a later tutorial. Just know for now that importing of library classes allows you to incorporate additional functionality outside of what's available by default in Java. To import the library class, you will use the import command at the top of the program before the class is defined.

The class that needs to be imported here is `java.time.LocalDate` using this import statement as the first line of code:

```
import java.time.LocalDate;
```

```
public class UserAccount {
  private String userName;
  private String password;
  private LocalDate dateJoined = LocalDate.now();
  private boolean activeUser = true;

  public UserAccount(String userName, String password) {
    this.userName = userName;
    this.password = password;
  }
}
```

HINT

Note that in addition to setting these values when the attributes in the class are declared, you can set their initial values in the parameterized constructor, even though no values for these were passed:

```
public class UserAccount {
  private String userName;
  private String password;
  private LocalDate dateJoined;
  private boolean activeUser;

  public UserAccount(String userName, String password) {
    this.userName = userName;
    this.password = password;
    //this.dateJoined = LocalDate.now();
    //activeUser = true;
  }
```

TRY IT

**Directions:** Add the new attributes and default values to your `User` class. Setting the values for the new attribute `activeUser` should be familiar. You would simply set the `activeUser` variable to true for every instance of the `UserAccount` class that is created.

The `dateJoined` attribute is using the `java.time.LocalDate` to get today's date. It is set using the `now()` method of the `LocalDate` class. Again, these are an object and method provided by the standard Java libraries. Once the class is imported, you would have access to use anything that class has prebuilt in it.

TRY IT

**Directions:** Let's add an output for each of these new attributes to our program and use them.

The output of each attribute of the account instance would look like this:

```
User name: Sophia
Is active user: true
Date joined: 2024-04-29
```

REFLECT

Notice that when creating our instance, nothing has changed, as you would just be passing in the username and password. All of the other attributes are automatically set.

## SUMMARY

In this lesson, you learned how to **set values to attributes** and change those values after the instances have been created. You also learned how to **set default values for attributes**. Finally, you briefly looked at importing the java.time.LocalDate class to set the default value using the current date.

Source: This content and supplemental material has been adapted from Java, Java, Java: Object-Oriented Problem Solving. Source cs.trincoll.edu/~ram/jjj/jjj-os-20170625.pdf

It has also been adapted from “Python for Everybody” By Dr. Charles R. Severance. Source py4e.com/html3/

## TERMS TO KNOW

**Driver Class**

The class in an application that contains the main() method.