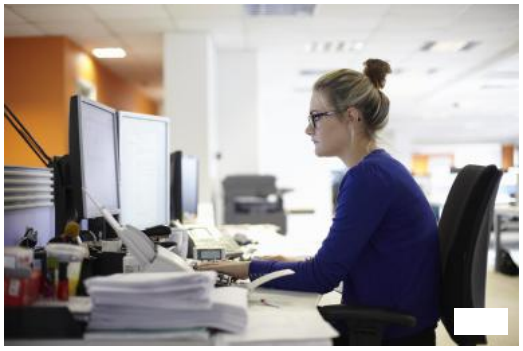
By Paul Leahy
Java Expert



Richard Drury/Iconica/Getty Images

Updated November 24, 2014.

One of the ways we can enforce data encapsulation is through the use of accessors and mutators. The role of accessors and mutators are to return and set the values of an object's state. This article is a practical guide on how to program them in Java.

As an example I'm going to use a Person class with the following state and constructor already defined:

```
public class Person {

  //Private fields
  private String firstName;
  private String middleNames;
  private String lastName;
  private String address;
  private String username;

  //Constructor method
  public Person(String firstName, String middleNames, String
lastName, String address)
  {
    this.firstName = firstName;
    this.middleNames = middleNames;
    this.lastName = lastName;
    this.address = address;
  }
```

Accessor Methods

An accessor method is used to return the value of a private field. It follows a naming scheme prefixing the word "get" to the start of the method name. For example let's add accessor methods for firstname, middleNames and lastname:

```
  //Accessor for firstName
  public String getFirstName()
  {
    return firstName;
  }
```

```
//Accessor for middleNames
public String getMiddlesNames()
{
  return middleNames;
}

//Accessor for lastName
public String getLastName()
{
  return lastName;
}
```


These methods always return the same data type as their corresponding private field (e.g., String) and then simply return the value of that private field.

We can now access their values through the methods of a Person object:

```
public class PersonExample {

  public static void main(String[] args) {

    Person dave = new Person("Dave", "Bob Bill", "Davidson", "12
Pall Mall");
    System.out.println(dave.getFirstName() + " " +
dave.getMiddlesNames() + " " + dave.getLastName());
  }
}
```


A mutator method is used to set a value of a private field. It follows a naming scheme prefixing the word "set" to the start of the method name. For example, let's add mutator fields for address and username:

```
//Mutator for address
public void setAddress(String address)
{
  this.address = address;
}

//Mutator for username
public void setUsername(String username)
{
  this.username = username;
}
```

These methods do not have a return type and accept a parameter that is the same data type as their corresponding private field.

The parameter is then used to set the value of that private field.

It's now possible to modify the values for the address and username inside the Person object:

```
public class PersonExample {

  public static void main(String[] args) {

    Person dave = new Person("Dave", "Bob Bill", "Davidson", "12
Pall Mall");
    dave.setAddress("256 Bow Street");
    dave.setUsername("DDavidson");

  }
}
```

Why Use Accessors and Mutators?

It's easy to come to the conclusion that we could just change the private fields of the class definition to be public and achieve the same results. It's important to remember that we want to hide the data of the object as much as possible. The extra buffer provided by these methods allows us to:

- change how the data is handled behind the scenes
- impose validation on the values that the fields are being set to.

Let's say we decide to modify how we store middle names. Instead of just one String we now use an array of Strings:

```
  private String firstName;
  //Now using an array of Strings
  private String[] middleNames;
  private String lastName;
  private String address;
  private String username;

  public Person(String firstName, String middleNames, String
lastName, String address)
  {
    this.firstName = firstName;

    //create an array of Strings
    this.middleNames = middleNames.split(" ");
    this.lastName = lastName;
    this.address = address;
    this.username = "";
  }

  //Accessor for middleNames
  public String getMiddlesNames()
  {
    //return a String by appending all the Strings of middleNames
together
    StringBuilder names = new StringBuilder();

    for(int j=0;j < (middleNames.length-1);j++)
    {
      names.append(middleNames[j] + " ");
    }
```

```
        names.append(middleNames[middleNames.length-1]);
        return names.toString();
    }
```

The implementation inside the object has changed but the outside world is not affected. The way the methods are called remains exactly the same:

```
public class PersonExample {

  public static void main(String[] args) {

    Person dave = new Person("Dave", "Bob Bill", "Davidson", "12
Pall Mall");
    System.out.println(dave.getFirstName() + " " +
dave.getMiddlesNames() + " " + dave.getLastName());
  }
}
```

Or, let's say the application that is using the Person object can only accept usernames that have a maximum of ten characters. We can add validation in the setUsername mutator to make sure the username conforms to this requirement:

```
public void setUsername(String username)
{
  if (username.length() > 10)
  {
    this.username = username.substring(0,10);
  }
  else
  {
    this.username = username;
  }
}
```

Now if the username passed to the setUsername mutator is longer than ten characters it is automatically truncated.

Related Articles

- **Data Encapsulation**
- **Using Java Constructors: An Easy How-To Guide**
- **Object-Oriented Programming**
- **Introduction to Object-Oriented Programming**
- **Object Oriented JavaScript**
- **Instance Variables**

Our Expert Recommends

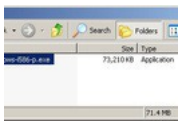
- **Data Encapsulation**
- **Designing and Creating Objects**
- **The Constructor Method**

More from the Web

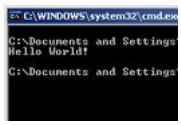
Java Essentials

Wondering What Java Is? Here's the Answer
Not Sure Where to Start? This is the Place for You

Navigate to the Java Download Website



Find the JDK Install File



How to Write Java Source Code
Create Java Applications for Your Desktop

Understanding Reserved Object Variable Words in Java
Learn About the Basic Building Blocks of Java

Tech Slideshows

Great Laptops Under $500
PC Reviews

Need Speakers? Forget Wires!
Home Audio & Headphones

Wireless Headphones: Now Better, Still Convenient
Home Audio & Headphones

Apple Updates the MacBook: Faster and More Battery Life
Macs

Readers Recommend

- **Sample Java Code for Building a Simple GUI App**
- **A Quick Guide to Using Accessors and Mutators in Java**
- **A Quick Guide to Using Constants Syntax in Java**
- **How to Declare Variables in Java**
- **Java Code Listing for a Simple Calculator**