COURSEWARE

Access Modifiers

Contents

Overview

A Java *access modifier* specifies which classes can access a given class and its fields, constructors, and methods. Classes, fields, constructors, and methods can each can have one of four different Java access modifiers:

- `private`
- `default`
- `protected`
- `public`

(*note: A class may not be* `private` *or* `protected` *unless it is a nested class.*)

Tutorial

Private

If a method or variable is marked as `private`, then only code inside the same class can access the variable, or call the method. Code inside subclasses cannot access the variable or method, nor can code from any external class.

```
public class BankAccount {

    private float balance = 0.52F;

}
```

Accessing Private Variables

`private` variables may be accessed with 'getter' and 'setter' methods. Because these methods are part of the class, there will be no issue changing or viewing the variables.

(The use of `this` refers to the current instance of the object being called.)

```
public class BankAccount {

    private float balance = 0.52F;

    public float getBalance() {
        return this.balance;
    }

    public void setBalance(float newBalance) {
        this.balance = newBalance;
    }
}
```

Private Constructors

If a constructor in a class is assigned the `private` access modifier, that means that the constructor cannot be called from anywhere outside the class. A `private` constructor can still get called from other constructors, or from static methods in the same class:

```
public class BankAccount {

    private float balance = 0;
    private long accountNum;

    private BankAccount(long newAccountNum) {
        this.accountNum = newAccountNum;
    }

    public BankAccount(float newBalance, long newAccountNum) {
        this(newAccountNum);
        this.balance = newBalance;
    }

    public static BankAccount newBankAccount() {
        return new BankAccount(80105101114115L);
    }

}
```

The use of `this` calls a constructor with the provided input parameters - in this case, the `private` constructor. It has the same effect as calling `new BankAccount(newAccountNum);`

Default

The `default` access modifier is assigned if no other access modifier is assigned.

Using `default` means that code inside the class itself, as well as code inside classes in the same package as this class, can access the resource which the `default` access modifier is assigned to.

```
public class BankAccount {

    float balance = 0.52F;

}

public class BankReader {
    BankAccount bank = new BankAccount();

    public float readBalance(){
        return bank.balance;
    }
}
```

In this example, the class `BankReader` can access the balance variable because it has the `default` access modifier.

Protected

The `protected` access modifier provides the same access as the `default` access modifier, with the addition that subclasses can access `protected` methods and variables of the superclass. This applies even if the subclass is not located in the same package as the superclass.

```
public class BankAccount {

    protected float balance = 0.52F;

}

public class BigBank extends BankAccount {

    public float getBalance() {
        return this.balance + 1000;
    }

}
```

Public

The `public` access modifier allows all other code in the program to access the resource, irrespective of location:

```
public class BankAccount {

    public float balance = 0.52F;
}

public class BankReader {

    BankAccount bank = new BankAccount();

    public float readBank{
        return bank.balance;
    }

}
```

The main difference between `default` and `protected` is that the class 'BankReader' can be located in a different package in this example.

Class Access Modifiers

It is important to keep in mind that the access modifier that is assigned at class level takes precedence over any access modifiers assigned to the variables, constructors, and methods within that class at a *lower level of exclusivity*.

For example, if the class is marked with the `default` access modifier, then no other class outside the same Java package can access that class, including its constructors, fields, and methods.

Here, the variable `balance` would not take on the `public` access modifier, as the access at class level is `default`:

```
class BankAccount {

    public float balance = 0;

}
```

Conversely, here, the variable `balance` would take on the `private` access modifier, as it is more *exclusive* than the `default` access modifier at the class level:

```
class BankAccount {

    private float balance = 0;

}
```

Exercises

There are no exercises for this module.