Debugging Inheritance

by Sophia

WHAT'S COVERED

In this lesson, you will learn about a couple of common errors in Java classes that make use of inheritance. Specifically, this lesson covers:

Table of Contents

1. Common Errors With Inheritance

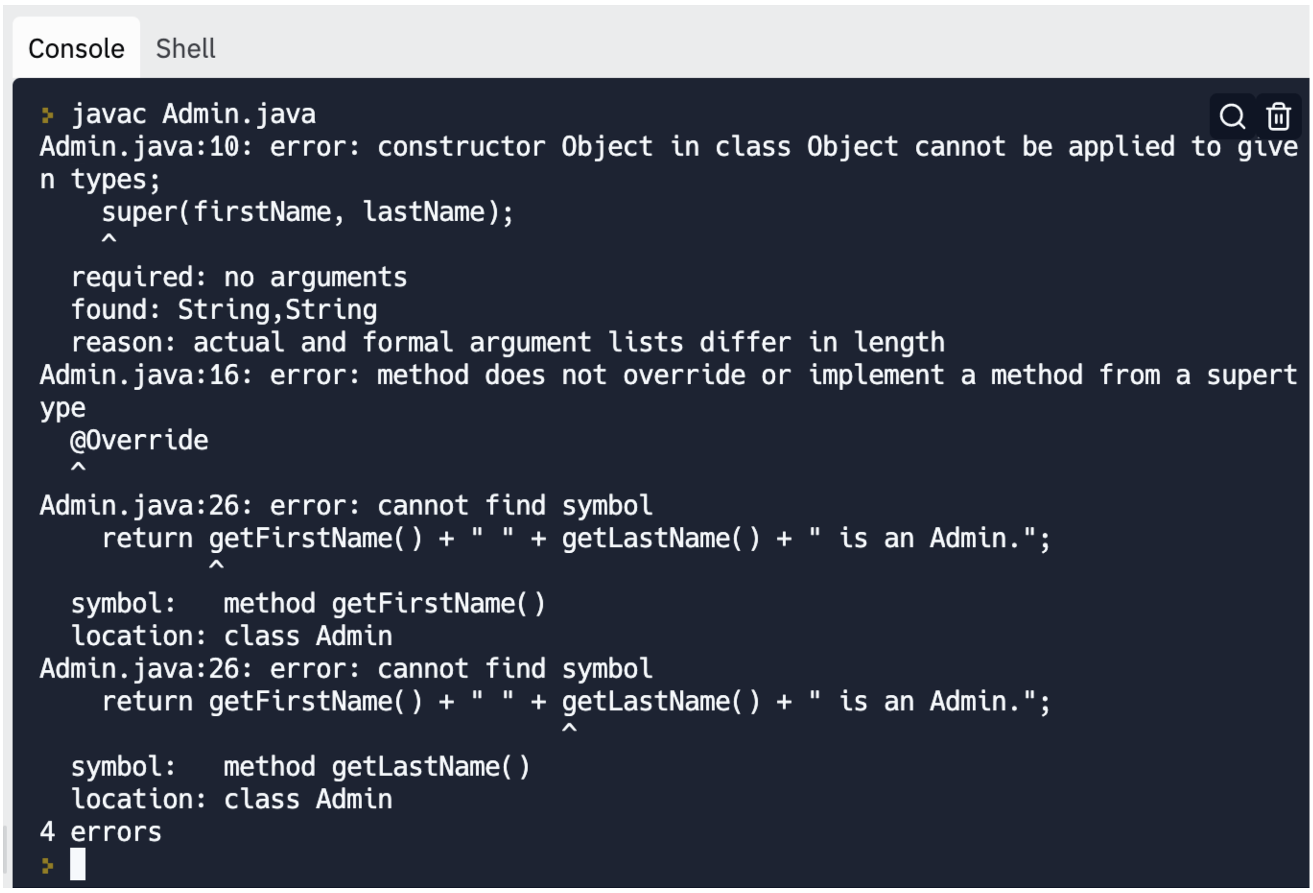
As we have seen, the base class does not differ visibly from any other class. This is not true of a subclass, though. Remember that the first line of a subclass must include the keyword `extends`, and this must be followed by the name of the base class as demonstrated below:

```
public class Admin extends Member{
```

If the `extends Member` is left out so that the `Admin` class looks like this, the code won't compile.

```
import java.time.LocalDate;

// Subclass of Member for administrators
public class Admin {
  private int expiryDays = 100 * 365;
  private LocalDate expiryDate;
  private String secret;

  public Admin(String firstName, String lastName, String secret) {
    super(firstName, lastName);
    expiryDate = LocalDate.now().plusDays(expiryDays);
    this.secret = secret;
  }
  @Override
  public LocalDate getExpiryDate() {
    return expiryDate;
  }

  public String getSecret() {
    return secret;
  }

  public String getStatus() {
    return getFirstName() + " " + getLastName() + " is an Admin.";
  }
}
```

The compiler output looks like this:



```
> javac Admin.java
Admin.java:10: error: constructor Object in class Object cannot be applied to give
n types;
    super(firstName, lastName);
    ^
  required: no arguments
  found: String,String
  reason: actual and formal argument lists differ in length
Admin.java:16: error: method does not override or implement a method from a supert
ype
  @Override
  ^
Admin.java:26: error: cannot find symbol
    return getFirstName() + " " + getLastName() + " is an Admin.";
           ^
  symbol:   method getFirstName()
  location: class Admin
Admin.java:26: error: cannot find symbol
    return getFirstName() + " " + getLastName() + " is an Admin.";
                                  ^
  symbol:   method getLastName()
  location: class Admin
4 errors
> 
```

Since the Admin class does not explicitly extend the Member class, the compiler assumes that the call to super() refers to the Java **Object class**, since all classes are subclasses of Object, but an Object instance can't be created using the arguments passed. The compiler can't process the @Override annotation because there is nothing to override.

The error message also indicates that the compiler can't find the getFirstName() and getLastName() methods. These don't exist in the Admin class because it is supposed to inherit them from the Member base class. The error message doesn't state that the extends Member has been left out, but the clues indicate that the Admin class needs to extend a class other than Object and that the class to be extended must include the getFirstName() and getLastName() methods.

When working with a subclass, it is important to remember that the subclass's constructor must call super() to invoke the base class’s (or superclass’s) constructor. This call must include arguments that match the superclass’s constructor.

Let's look at some code that you have worked with previously.

Here is the Member base class:

```
import java.time.LocalDate;

public class Member {
  private String firstName;
  private String lastName;
  private int expiryDays = 365;
  private LocalDate expiryDate;

  public Member(String firstName, String lastName) {
    this.firstName = firstName;
    this.lastName = lastName;
    expiryDate = LocalDate.now().plusDays(expiryDays);
  }

  public String getFirstName() {
    return firstName;
  }

  public String getLastName() {
    return lastName;
  }

  public LocalDate getExpiryDate() {
    return expiryDate;
  }

  public String getStatus() {
    return firstName + " " + lastName + " is a Member.";
  }
}
```

Remember that the base class does not have any features that distinguish it from other classes. This is not true of a subclass, though, which is marked by the `extends` keyword between the name of the subclass and the base class:

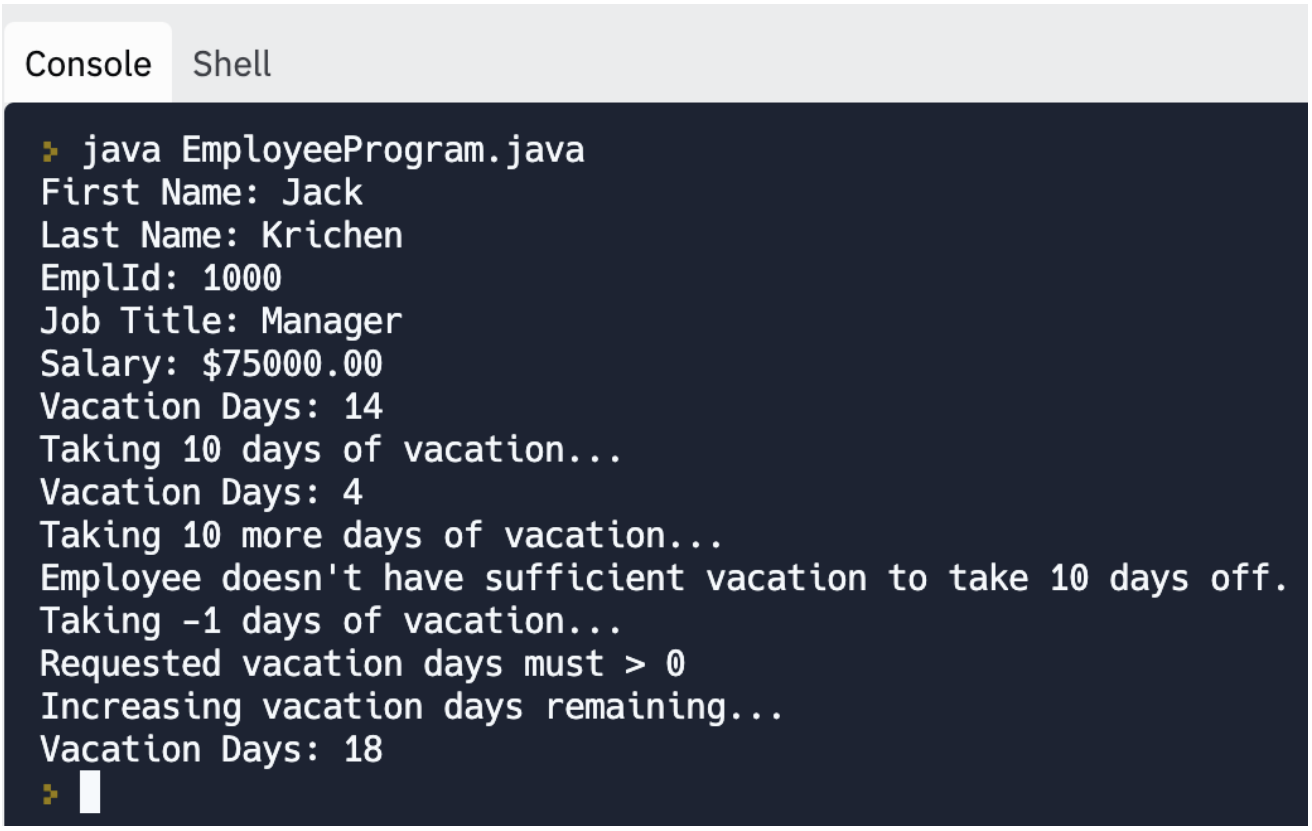
```
import java.time.LocalDate;

// Subclass of Member for administrators
public class Admin extends Member{
  private int expiryDays = 100 * 365;
  private LocalDate expiryDate;
  private String secret;

  public Admin(String firstName, String lastName, String secret) {
    super(firstName, lastName);
    expiryDate = LocalDate.now().plusDays(expiryDays);
    this.secret = secret;
  }

  @Override
  public LocalDate getExpiryDate() {
    return expiryDate;
  }

  public String getSecret() {
    return secret;
  }

  public String getStatus() {
    return getFirstName() + " " + getLastName() + " is an Admin.";
  }
}
```

Note that the code in the subclass's constructor must call the base class's constructor via super() (with arguments to match the base class's arguments). This must be the first statement in the subclass's constructor. If the call to super() doesn't come first, the subclass won't compile. If the Admin() constructor's statements are ordered like this:

↪ EXAMPLE

```
public Admin(String firstName, String lastName, String secret) {
  expiryDate = LocalDate.now().plusDays(expiryDays);
  this.secret = secret;
  super(firstName, lastName); // In wrong place - should be first statement
}
```

The compiler will produce an error message like this:



```
> java EmployeeProgram.java
First Name: Jack
Last Name: Krichen
EmplId: 1000
Job Title: Manager
Salary: $75000.00
Vacation Days: 14
Taking 10 days of vacation...
Vacation Days: 4
Taking 10 more days of vacation...
Employee doesn't have sufficient vacation to take 10 days off.
Taking -1 days of vacation...
Requested vacation days must > 0
Increasing vacation days remaining...
Vacation Days: 18
> 
```

This output may seem a bit hard to understand, but in the final couple of lines it indicates that the call to `super()` with the appropriate arguments must be the first statement in the constructor.

TERM TO KNOW

Object Class
Every class in Java automatically inherits from the Object class, so it is the base class for all other Java classes.

SUMMARY

In this lesson, you learned about **common errors when working with inheritance**. You learned how to recognize them when confronted with compiler errors, which are not always so clear. Finally, you learned about correct uses of the extends keyword when defining a subclass and that getting the order of statements in the constructor incorrect is an error that you can easily avoid.

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

Object Class
Every class in Java automatically inherits from the Object class, so it is the base class for all other Java classes.