5.7 Class `AutoPolicy` Case Study: `Strings` in `switch` Statements

`String`s can be used as controlling expressions in `switch` statements, and `String` literals can be used in `case` labels. To demonstrate this, we'll implement an app that meets the following requirements:

You've been hired by an auto insurance company that serves these northeast states—Connecticut, Maine, Massachusetts, New Hampshire, New Jersey, New York, Pennsylvania, Rhode Island and Vermont. The company would like you to create a program that produces a report indicating for each of their auto insurance policies whether the policy is held in a state with "no-fault" auto insurance—Massachusetts, New Jersey, New York and Pennsylvania.

The Java app that meets these requirements contains two classes—`AutoPolicy` (Fig. 5.11) and `AutoPolicyTest` (Fig. 5.12).

Class `AutoPolicy`

Class `AutoPolicy` (Fig. 5.11) represents an auto insurance policy. The class contains:

- `int` instance variable `accountNumber` (line 4) to store the policy's account number
- `String` instance variable `makeAndModel` (line 5) to store the car's make and model (such as a `"Toyota Camry"`)
- `String` instance variable `state` (line 6) to store a two-character state abbreviation representing the state in which the policy is held (e.g., `"MA"` for Massachusetts)
- a constructor (lines 9–14) that initializes the class's instance variables
- methods `setAccountNumber` and `getAccountNumber` (lines 17–24) to *set* and *get* an `AutoPolicy`'s `accountNumber` instance variable
- methods `setMakeAndModel` and `getMakeAndModel` (lines 27–34) to *set* and *get* an `AutoPolicy`'s `makeAndModel` instance variable
- methods `setState` and `getState` (lines 37–44) to *set* and *get* an `AutoPolicy`'s `state` instance variable
- method `isNoFaultState` (lines 47–61) to return a `boolean` value indicating whether the policy is held in a no-fault auto insurance state; note the method name—the naming convention for a *get* method that returns a `boolean` value is to begin the name with `"is"` rather than `"get"` (such a method is commonly called a *predicate method*).

```
        1   // Fig. 5.11: AutoPolicy.java
2   // Class that represents an auto insurance polic
          3   public class AutoPolicy {
4      private int accountNumber; // policy account
5      private String makeAndModel; // car that the
6      private String state; // two-letter state abb
                        7
               8      // constructor
9      public AutoPolicy(int accountNumber, String m
           10         String state) {
```

```
11          this.accountNumber = accountNumber;
12           this.makeAndModel = makeAndModel;
13           this.state = state;
14       }
15
16       // sets the accountNumber
17       public void setAccountNumber(int accountNumbe
18          this.accountNumber = accountNumber;
19       }
20
21       // returns the accountNumber
22       public int getAccountNumber() {
23          return accountNumber;
24       }
25
26       // sets the makeAndModel
27       public void setMakeAndModel(String makeAndMod
28          this.makeAndModel = makeAndModel;
29       }
30
31       // returns the makeAndModel
32       public String getMakeAndModel() {
33          return makeAndModel;
34       }
35
36       // sets the state
37       public void setState(String state) {
38           this.state = state;
39       }
40
41       // returns the state
42       public String getState() {
43          return state;
44       }
45
46       // predicate method returns whether the state
47       public boolean isNoFaultState(){
48          boolean noFaultState;
49
50          // determine whether state has no-fault au
```

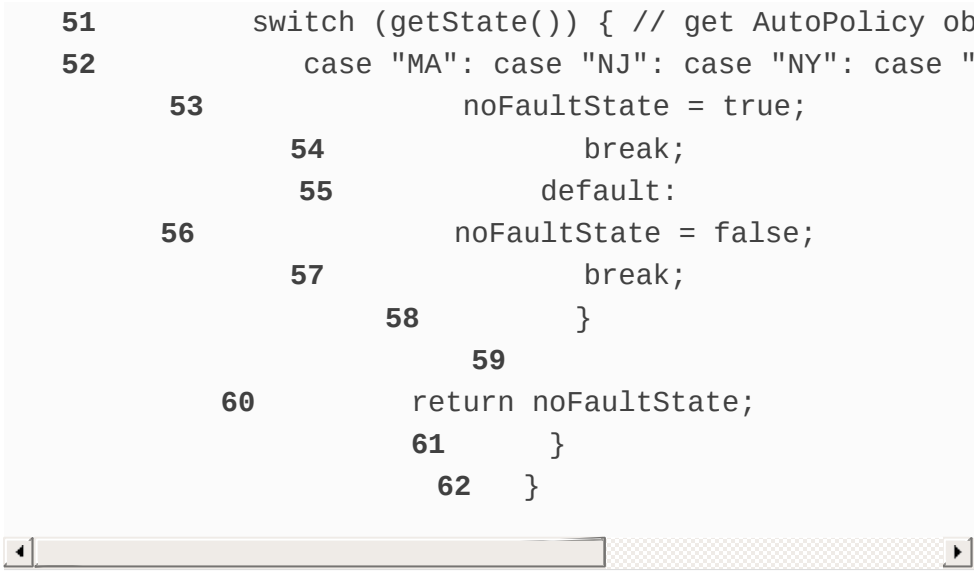

Fig. 5.11

Class that represents an auto insurance policy.

In method `isNoFaultState`, the `switch` statement's controlling expression (line 51) is the `String` returned by `AutoPolicy` method `getState`. The `switch` statement compares the controlling expression's value with the `case` labels (line 52) to determine whether the policy is held in Massachusetts, New Jersey, New York or Pennsylvania (the no-fault states). If there's a match, then line 53 sets local variable `noFaultState` to `true` and the `switch` statement terminates; otherwise, the `default` case sets `noFaultState` to `false`(line 56). Then method `isNoFaultState` returns local variable `noFaultState`'s value.

For simplicity, we did not validate an `AutoPolicy`'s data in

the constructor or *set* methods, and we assume that state abbreviations are always two uppercase letters. In addition, a real `AutoPolicy` class would likely contain many other instance variables and methods for data such as the account holder's name, address, etc. In Exercise 5.30, you'll be asked to enhance class `AutoPolicy` by validating its state abbreviation using techniques that you'll learn in Section 5.9.

Class `AutoPolicyTest`

Class `AutoPolicyTest` (Fig. 5.12) creates two `AutoPolicy` objects (lines 6–9 in `main`). Lines 12–13 pass each object to `static` method `policyInNoFaultState` (lines 18–25), which uses `AutoPolicy` methods to determine and display whether the object it receives represents a policy in a no-fault auto insurance state.

```
   1   // Fig. 5.12: AutoPolicyTest.java
   2   // Demonstrating Strings in switch.
   3   public class AutoPolicyTest {
   4      public static void main(String[] args) {
   5         // create two AutoPolicy objects
   6         AutoPolicy policy1 =
   7            new AutoPolicy(11111111, "Toyota Camry"
   8         AutoPolicy policy2 =
   9            new AutoPolicy(22222222, "Ford Fusion",
  10
  11        // display whether each policy is in a no-f
  12        policyInNoFaultState(policy1);
  13        policyInNoFaultState(policy2);
  14     }
  15
```

```
16     // method that displays whether an AutoPolicy
17     // is in a state with no-fault auto insurance
18     public static void policyInNoFaultState(AutoPo
19        System.out.println("The auto policy:");
20        System.out.printf(
21           "Account #: %d; Car: %s;%nState %s %s a
22           policy.getAccountNumber(), policy.getMak
23              policy.getState(),
24           (policy.isNoFaultState() ? "is": "is not
25        }
26     }
```

```
The auto policy:
Account #: 11111111; Car: Toyota Camry;
State NJ is a no-fault state

The auto policy:
Account #: 22222222; Car: Ford Fusion;
State ME is not a no-fault state
```

Fig. 5.12

Demonstrating Strings in switch.