Exercise 2.63 *Challenge exercise* Suppose we wished a single `TicketMachine` object to be able to issue tickets of different prices. For instance, users might press a button on the physical machine to select a discounted ticket price. What further methods and/or fields would need to be added to `TicketMachine` to allow this kind of functionality? Do you think that many of the existing methods would need to be changed as well?

Save the *better-ticket-machine* project under a new name, and implement your changes in the new project.

2.18 Summary of the better ticket machine

In developing a better version of the `TicketMachine` class, we have been able to address the major inadequacies of the naïve version. In doing so, we have introduced two new language constructs: the conditional statement and local variables.

- A conditional statement gives us a means to perform a test and then, on the basis of the result of that test, perform one or the other of two distinct actions.
- Local variables allow us to calculate and store temporary values within a constructor or method. They contribute to the behavior that their defining method implements, but their values are lost once that constructor or method finishes its execution.

You can find more details of conditional statements and the form their tests can take in Appendix D.

2.19 Self-review exercises

This chapter has covered a lot of new ground, and we have introduced a lot of new concepts. We will be building on these in future chapters, so it is important that you are comfortable with them. Try the following pencil-and-paper exercises as a way of checking that you are becoming used to the terminology that we have introduced in this chapter. Don't be put off by the fact that we suggest that you do these on paper rather than within BlueJ. It will be good practice to try things out without a compiler.

Exercise 2.64 List the name and return type of this method:

```
public String getCode()
{
    return code;
}
```

Exercise 2.65 List the name of this method and the name and type of its parameter:

```
public void setCredits(int creditValue)
{
    credits = creditValue;
}
```

Exercise 2.66 Write out the outer wrapping of a class called **Person**. Remember to include the curly brackets that mark the start and end of the class body, but otherwise leave the body empty.

Exercise 2.67 Write out definitions for the following fields:

- a field called **name** of type **String**
- a field of type **int** called **age**
- a field of type **String** called **code**
- a field called **credits** of type **int**

Exercise 2.68 Write out a constructor for a class called **Module**. The constructor should take a single parameter of type **String** called **moduleCode**. The body of the constructor should assign the value of its parameter to a field called **code**. You don't have to include the definition for **code**, just the text of the constructor.

Exercise 2.69 Write out a constructor for a class called **Person**. The constructor should take two parameters. The first is of type **String** and is called **myName**. The second is of type **int** and is called **myAge**. The first parameter should be used to set the value of a field called **name**, and the second should set a field called **age**. You don't have to include the definitions for the fields, just the text of the constructor.

Exercise 2.70 Correct the error in this method:

```
public void getAge()
{
    return age;
}
```

Exercise 2.71 Write an accessor method called **getName** that returns the value of a field called **name**, whose type is **String**.