2.10 Method summary

It is worth summarizing a few features of methods at this point, because methods are fundamental to the programs we will be writing and exploring in this book. They implement the core actions of every object.

A method with parameters will receive data passed to it from the method's caller, and will then use that data to help it perform a particular task. However, not all methods take parameters; many simply use the data stored in the object's fields to carry out their task.

If a method has a non-**`void`** return type, it will return some data to the place it was called from—and that data will almost certainly be used in the caller for further calculations or program manipulations. Many methods, though, have a **`void`** return type and return nothing, but they still perform a useful task within the context of their object.

Accessor methods have non-**`void`** return types and return information about the object's state. Mutator methods modify an object's state. Mutators often take parameters whose values are used in the modification, although it is still possible to write a mutating method that does not take parameters.

2.11 Summary of the naíve ticket machine

We have now examined the internal structure of the naíve **`TicketMachine`** class in some detail. We have seen that the class has a small outer layer that gives a name to the class, and a more substantial inner body containing fields, a constructor, and several methods. Fields are used to store data that enable objects to maintain a state that persists between method calls. Constructors are used to set up an initial state when an object is created. Having a proper initial state will enable an object to respond appropriately to method calls immediately following its creation. Methods implement the defined behavior of the class's objects. Accessor methods provide information about an object's state, and mutator methods change an object's state.

We have seen that constructors are distinguished from methods by having the same name as the class in which they are defined. Both constructors and methods may take parameters, but only methods may have a return type. Non-**`void`** return types allow us to pass a value out of a method to the place where the method was called from. A method with a non-**`void`** return type must have at least one return statement in its body; this will often be the final statement. Constructors never have a return type of any sort—not even **`void`**.

Before attempting these exercises, be sure that you have a good understanding of how ticket machines behave and how that behavior is implemented through the fields, constructor, and methods of the class.

Exercise 2.43 Modify the constructor of **`TicketMachine`** so that it no longer has a parameter. Instead, the price of tickets should be fixed at 1,000 cents. What effect does this have when you construct ticket-machine objects within BlueJ?