Exercise 2.4 Try to obtain a good understanding of a ticket machine's behavior by interacting with it on the object bench before we start looking, in the next section, at how the `TicketMachine` class is implemented.

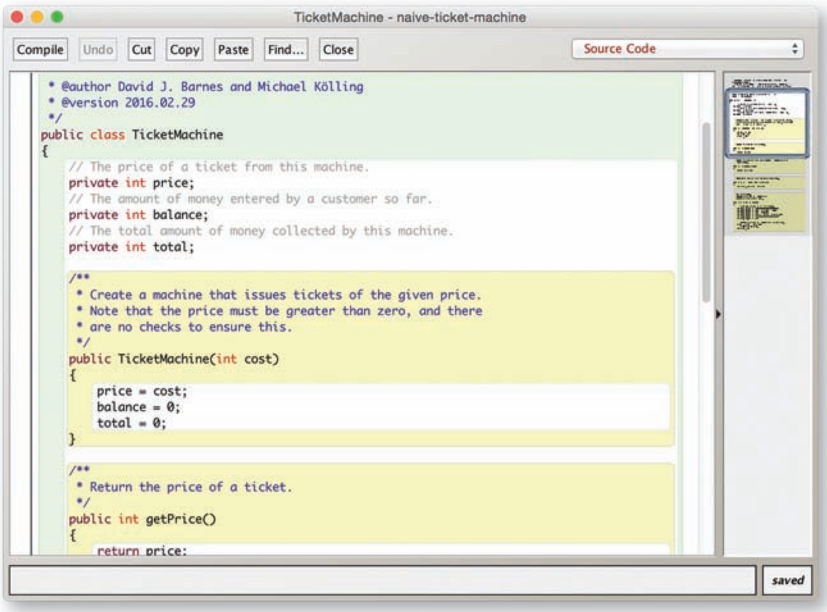
Exercise 2.5 Create another ticket machine for tickets of a different price; remember that you have to supply this value when you create the machine object. Buy a ticket from that machine. Does the printed ticket look any different from those printed by the first machine?

2.2 Examining a class definition

The exercises at the end of the previous section reveal that `TicketMachine` objects only behave in the way we expect them to if we insert the exact amount of money to match the price of a ticket. As we explore the internal details of the class in this section, we shall see why this is so.

Take a look at the source code of the `TicketMachine` class by double-clicking its icon in the class diagram within BlueJ. It should look something like Figure 2.1.

The complete text of the class is shown in Code 2.1. By looking at the text of the class definition piece by piece, we can flesh out some of the object-oriented concepts that discussed in Chapter 1. This class definition contains many of the features of Java that we will see over and over again, so it will pay greatly to study it carefully.



```
 * @author David J. Barnes and Michael Kölling
 * @version 2016.02.29
 */
public class TicketMachine
{
    // The price of a ticket from this machine.
    private int price;
    // The amount of money entered by a customer so far.
    private int balance;
    // The total amount of money collected by this machine.
    private int total;

    /**
     * Create a machine that issues tickets of the given price.
     * Note that the price must be greater than zero, and there
     * are no checks to ensure this.
     */
    public TicketMachine(int cost)
    {
        price = cost;
        balance = 0;
        total = 0;
    }

    /**
     * Return the price of a ticket.
     */
    public int getPrice()
    {
        return price;
```

Figure 2.1
The BlueJ editor window

Code 2.1
The `TicketMachine` class

```java
/**
 * TicketMachine models a naive ticket machine that issues
 * flat-fare tickets.
 * The price of a ticket is specified via the constructor.
 * It is a naive machine in the sense that it trusts its users
 * to insert enough money before trying to print a ticket.
 * It also assumes that users enter sensible amounts.
 *
 * @author David J. Barnes and Michael Kölling
 * @version 2016.02.29
 */
public class TicketMachine
{
    // The price of a ticket from this machine.
    private int price;
    // The amount of money entered by a customer so far.
    private int balance;
    // The total amount of money collected by this machine.
    private int total;

    /**
     * Create a machine that issues tickets of the given price.
     * Note that the price must be greater than zero, and there
     * are no checks to ensure this.
     */
    public TicketMachine(int cost)
    {
        price = cost;
        balance = 0;
        total = 0;
    }

    /**
     * Return the price of a ticket.
     */
    public int getPrice()
    {
        return price;
    }

    /**
     * Return the amount of money already inserted for the
     * next ticket.
     */
    public int getBalance()
    {
        return balance;
    }

    /**
     * Receive an amount of money from a customer.
     */
```