CONDITIONAL EXECUTION IN JAVA

Recall our `MovingBall` program of Figure 6.3, in which a red ball moved steadily down and to the right, eventually moving off the screen. Suppose that instead we want the ball to bounce when it reaches the window's edge. To do this, we need some way to test whether it has reached the edge. The most natural way of accomplishing this in Java is to use the **`if`** statement that we study in this chapter.

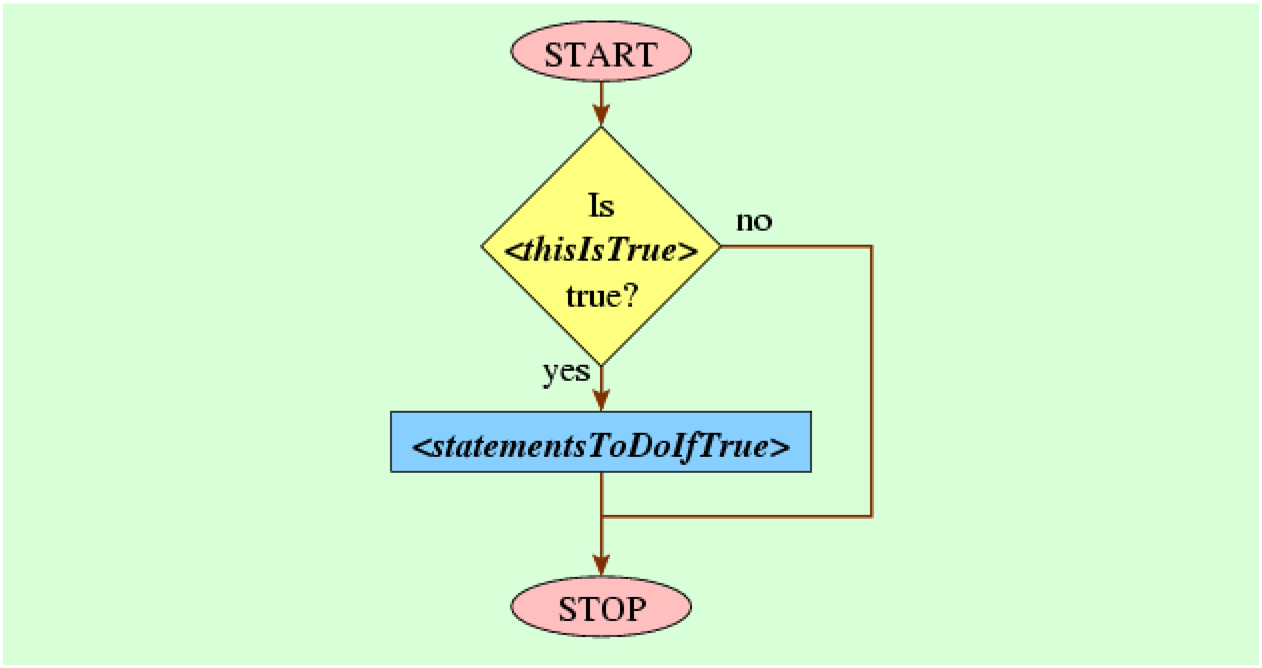
7.1. The `if` statement

An **`if` statement** tells the computer to execute a sequence of statements only *if* a particular condition holds. It is sometimes called a conditional statement, since it allows us to indicate that the computer should execute some statements only in some conditions.

```
if(<thisIsTrue>) {
    <statementsToDoIfTrue>
}
```

When the computer reaches the **`if`** statement, it evaluates the condition in parentheses. If the condition turns out to be **`true`**, then the computer executes the **`if`** statement's body and then continues to any statements following. If the condition turns out to be **`false`**, it skips over the body and goes directly to whatever follows the closing brace. This corresponds to the flowchart in Figure 7.1.

Figure 7.1: A flowchart for the **`if`** statement.



An **if** statement looks a lot like a **while** loop: The only visual difference is that it uses the **if** keyword in place of **while**. And it acts like a **while** loop also, except that after completing the **if** statement's body, the computer does not consider whether to execute the body again.

The following fragment includes a simple example of an **if** statement in action.

```
double num = readDouble();
if(num < 0.0) {
    num = -num;
}
```

In this fragment, we have a variable `num` referring to a number typed by the user. If `num` is less than 0, then we change `num` to refer to the value of `-num` instead, so that `num` will now be the absolute value of what the user typed. But if `num` is not negative, the computer will skip the `num = -num` statement; `num` will still be the absolute value of what the user typed.

7.2. The bouncing ball

We can now turn to our `MovingBall` program, modifying it so that the ball will bounce off the edges of the window. To accomplish this, before each movement, we will test whether the ball has crossed an edge of the window: The movement in the *x* direction should invert if the ball has crossed the east or west side, and the movement in the *y*direction should invert if the ball has crossed the north or south side. The program of Figure 7.2 accomplishes this.

Figure 7.2: The `BouncingBall` program.

```java
import acm.program.*;
import acm.graphics.*;
import java.awt.*;

public class BouncingBall extends GraphicsProgram {
    public void run() {
        GOval ball = new GOval(25, 25, 50, 50);
        ball.setFilled(true);
        ball.setFillColor(new Color(255, 0, 0));
        add(ball);

        double dx = 3;
        double dy = -4;
        while(true) {
            pause(40);
            if(ball.getX() + ball.getWidth() >= getWidth()
                    || ball.getX() <= 0.0) {
                dx = -dx;
            }
            if(ball.getY() + ball.getHeight() >= getHeight()
                    || ball.getY() <= 0.0) {
                dy = -dy;
            }
            ball.move(dx, dy);
        }
    }
}
```

If you think about it carefully, the simulation of this program isn't perfect: There will be frames where the ball has crossed over the edge, so that only part of the circle is visible. This isn't a big problem, for a combination of reasons: First, the ball will be over the border only for a single frame (i.e., for only a fleeting few milliseconds), and second, the effect is barely visible because it affects only a few pixels of the ball's total size. Anyway, our eye expects to see a rubber ball flatten against the surface briefly before bouncing back in the opposite direction. If we were to repair this problem, then we would perceive the ball to be behaving more like a billiard ball. In any case, we won't attempt to repair the problem here.

Source : http://www.toves.org/books/java/ch07-if/index.html