Special Topic 6.1

do Loops

Sometimes you want to execute the body of a loop at least once and perform the loop test after the body was executed. The do loop serves that purpose:

```
do
   statement
while (condition);
```

The *statement* is executed while the *condition* is true. The condition is tested after the statement is executed, so the statement is executed at least once.

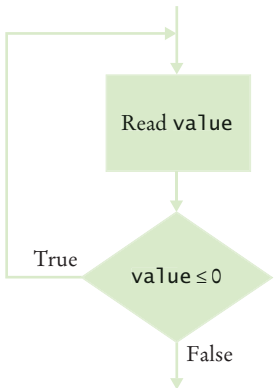
For example, suppose you want to make sure that a user enters a positive number. As long as the user enters a negative number or zero, just keep prompting for a correct input. In this situation, a do loop makes sense, because you need to get a user input before you can test it.

```
double value;
do
{
   System.out.print("Please enter a positive number: ");
   value = in.nextDouble();
}
while (value <= 0);
```

The figure shows a flowchart of this loop.

In practice, do loops are not very common. (The library code in Java 6 contains about 10,000 loop statements, but only about 2 percent are do loops.) Consider again the example of prompting for a positive value. In practice, you also need to guard against users who provide an input that isn't a number. Now the loop becomes so complex that you are better off controlling it with a Boolean variable:

```
boolean valid = false;
while (!valid)
{
   System.out.print("Please enter a positive number: ");
   if (in.hasNextDouble())
   {
      value = in.nextDouble();
      if (value > 0) valid = true;
   }
   else
     in.nextLine(); // Consume input
}
```



Flowchart of a do Loop