The do/while loop

- **do/while loop**: Performs its test at the *end* of each repetition.
- Guarantees that the loop's {} body will run at least once.

```java
do {
    <statement(s)>;
} while (<test>);
```

// Example: prompt until correct password is typed
String phrase;
do {
    System.out.print("Type your password: ");
    phrase = console.next();
} while (!phrase.equals("abracadabra"));
**do/while question**

- **Modify the previous Dice program to use do/while.**

  2 + 4 = 6  
  3 + 5 = 8  
  5 + 6 = 11  
  1 + 1 = 2  
  4 + 3 = 7  
  You won after 5 tries!

**do/while answer**

// Rolls two dice until a sum of 7 is reached.  
import java.util.*;

public class Dice {  
    public static void main(String[] args) {  
        Random rand = new Random();  
        int tries = 0;  
        int sum;  
        do {  
            int roll1 = rand.nextInt(6) + 1;  // one roll  
            int roll2 = rand.nextInt(6) + 1;  
            sum = roll1 + roll2;  
            System.out.println(roll1 + " + " + roll2 + " = " + sum);  
            tries++;  
        } while (sum != 7);  
        System.out.println("You won after " + tries + " tries!");  
    }  
}
**break**

- **break statement:** Immediately exits a loop.
  - Can be used to write a loop whose test is in the middle.
  - The loop’s test is often changed to `true` ("always repeat").
  ```java
  while (true) {
    <statement(s)>;
    if (<test>) {
      break;
    }
    <statement(s)>;
  }
  ```

- **break** is considered to be bad style by some programmers.

---

**Sentinel loop with break**

```java
Scanner console = new Scanner(System.in);
int sum = 0;
while (true) {
  System.out.print("Enter a number (-1 to quit): ");
  int number = console.nextInt();
  if (number == -1) {
    // don't add -1 to sum
    break;
  } // number != -1 here
  sum = sum + number;
}
System.out.println("The total was "+ sum);
```
Assertions

reading: 5.5

Logical assertions

- assertion: A statement that is either true or false.
  Examples:
  - Java was created in 1995.
  - The sky is purple.
  - The capital of North Dakota is Bismarck.
  - Mr. Marty met a monkey.
  - $x$ divided by 2 equals 7. (depends on the value of $x$)

- An assertion might be false ("The sky is purple" above), but it is still an assertion because it is a true/false statement.
Reasoning about assertions

• Suppose you have the following code:

```java
if (x > 3) {
    // Point A
    x--;  
} else {
    // Point B
    x++;  
    // Point C
}
// Point D
```

• What do you know about x's value at the three points?
  • Is x > 3? Always? Sometimes? Never?

Assertions in code

• We can make assertions about our code and ask whether they are true at various points in the code.
  • Valid answers are ALWAYS, NEVER, or SOMETIMES.

```java
System.out.print("Type a nonnegative number: ");
double number = console.nextDouble();
// Point A: is number < 0.0 here? (SOMETIMES)
while (number < 0.0) {
    // Point B: is number < 0.0 here? (ALWAYS)
    System.out.print("Negative; try again: ");
    number = console.nextDouble();
    // Point C: is number < 0.0 here? (SOMETIMES)
}
// Point D: is number < 0.0 here? (NEVER)
```
Reasoning about assertions

- Right after a variable is initialized, its value is known:
  ```java
  int x = 3;
  // is x > 0? ALWAYS
  ```

- In general you know nothing about parameters' values:
  ```java
  public static void mystery(int a, int b) {
  // is a == 10? SOMETIMES
  ```

- But inside an if, while, etc., you may know something:
  ```java
  public static void mystery(int a, int b) {
  if (a < 0) {
    // is a == 10? NEVER
    ...
  }
  ```

Assertions and loops

- At the start of a loop's body, the loop's test must be true:
  ```java
  while (y < 10) {
    // is y < 10? ALWAYS
    ...
  }
  ```

- After a loop, the loop's test must be false:
  ```java
  while (y < 10) {
    ...
  }
  // is y < 10? NEVER
  ```

- Inside a loop's body, the loop's test may become false:
  ```java
  while (y < 10) {
    y++;
    // is y < 10? SOMETIMES
  }
"Sometimes"

- Things that cause a variable's value to be unknown (often leads to "sometimes" answers):
  - reading from a Scanner
  - reading a number from a Random object
  - a parameter's initial value to a method

- If you can reach a part of the program both with the answer being "yes" and the answer being "no", then the correct answer is "sometimes".

Assertion example 1

```java
public static void mystery(int x, int y) {
    int z = 0;
    // Point A
    while (x >= y) {
        // Point B
        x = x - y;
        z++;
        if (x != y) {
            // Point C
            z = z * 2;
        }
        // Point D
    }
    // Point E
    System.out.println(z);
}
```

Which of the following assertions are true at which point(s) in the code? Choose ALWAYS, NEVER, or SOMETIMES.

<table>
<thead>
<tr>
<th></th>
<th>x &lt; y</th>
<th>x == y</th>
<th>z == 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point A</td>
<td>SOMETIMEs</td>
<td>SOMETIMEs</td>
<td>ALWAYS</td>
</tr>
<tr>
<td>Point B</td>
<td>NEVER</td>
<td>SOMETIMEs</td>
<td>SOMETIMEs</td>
</tr>
<tr>
<td>Point C</td>
<td>SOMETIMEs</td>
<td>NEVER</td>
<td>NEVER</td>
</tr>
<tr>
<td>Point D</td>
<td>SOMETIMEs</td>
<td>SOMETIMEs</td>
<td>NEVER</td>
</tr>
<tr>
<td>Point E</td>
<td>ALWAYS</td>
<td>NEVER</td>
<td>SOMETIMEs</td>
</tr>
</tbody>
</table>
Statement example 2

public static int mystery(Scanner console) {
    int prev = 0;
    int count = 0;
    int next = console.nextInt();

    // Point A
    while (next != 0) {
        // Point B
        if (next == prev) {
            // Point C
            count++;
        }
        prev = next;
        next = console.nextInt();
        // Point D
    }
    // Point E
    return count;
}

Which of the following assertions are true at which point(s) in the code?
Choose ALWAYS, NEVER, or SOMETIMES.

<table>
<thead>
<tr>
<th>Point</th>
<th>next == 0</th>
<th>prev == 0</th>
<th>next == prev</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>SOMETIMES</td>
<td>ALWAYS</td>
<td>SOMETIMES</td>
</tr>
<tr>
<td>B</td>
<td>NEVER</td>
<td>SOMETIMES</td>
<td>SOMETIMES</td>
</tr>
<tr>
<td>C</td>
<td>NEVER</td>
<td>NEVER</td>
<td>ALWAYS</td>
</tr>
<tr>
<td>D</td>
<td>NEVER</td>
<td>ALWAYS</td>
<td>SOMETIMES</td>
</tr>
<tr>
<td>E</td>
<td>ALWAYS</td>
<td>SOMETIMES</td>
<td>SOMETIMES</td>
</tr>
</tbody>
</table>

Statement example 3

// Assumes y >= 0, and returns x^y
public static int pow(int x, int y) {
    int prod = 1;

    // Point A
    while (y > 0) {
        // Point B
        if (y % 2 == 0) {
            // Point C
            x = x * x;
            y = y / 2;
            // Point D
        } else {
            // Point E
            prod = prod * x;
            y--;
            // Point F
        }
    }
    // Point G
    return prod;
}

Which of the following assertions are true at which point(s) in the code?
Choose ALWAYS, NEVER, or SOMETIMES.

<table>
<thead>
<tr>
<th>Point</th>
<th>y &gt; 0</th>
<th>y % 2 == 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>SOMETIMES</td>
<td>SOMETIMES</td>
</tr>
<tr>
<td>B</td>
<td>ALWAYS</td>
<td>SOMETIMES</td>
</tr>
<tr>
<td>C</td>
<td>ALWAYS</td>
<td>ALWAYS</td>
</tr>
<tr>
<td>D</td>
<td>ALWAYS</td>
<td>SOMETIMES</td>
</tr>
<tr>
<td>E</td>
<td>ALWAYS</td>
<td>NEVER</td>
</tr>
<tr>
<td>F</td>
<td>SOMETIMES</td>
<td>ALWAYS</td>
</tr>
<tr>
<td>G</td>
<td>NEVER</td>
<td>ALWAYS</td>
</tr>
</tbody>
</table>