Building Java Programs

Chapter 5
Lecture 5-3: Assertions, do/while loops

reading: 5.4 - 5.5
self-check: 22-24, 26-28
Logical assertions

- **assertion**: A statement that is either true or false.

Examples:
- Java was created in 1995.
- The sky is purple.
- 23 is a prime number.
- 10 is greater than 20.
- 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:

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

• 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? SOMETIME
  ```

- 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".

- If you're unsure, "Sometimes" is a good guess.
  - Often around 1/2 of the correct answers are "sometimes."
public static void mystery(int x, int y) {
    int z = 0;

    // Point A
    while (x >= y) {
        // Point B
        x = x - y;

        // Point C
        z++;

        // 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>Point</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>SOMETIMES</td>
<td>SOMETIMES</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>
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>Point A</td>
<td>SOMETIMES</td>
<td>ALWAYS</td>
<td>SOMETIMES</td>
</tr>
<tr>
<td>Point B</td>
<td>NEVER</td>
<td>SOMETIMES</td>
<td>SOMETIMES</td>
</tr>
<tr>
<td>Point C</td>
<td>NEVER</td>
<td>NEVER</td>
<td>ALWAYS</td>
</tr>
<tr>
<td>Point D</td>
<td>SOMETIMES</td>
<td>NEVER</td>
<td>SOMETIMES</td>
</tr>
<tr>
<td>Point E</td>
<td>ALWAYS</td>
<td>SOMETIMES</td>
<td>SOMETIMES</td>
</tr>
</tbody>
</table>
Assertion example 3

```java
// 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></th>
<th>y &gt; 0</th>
<th>y % 2 == 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point A</td>
<td>SOMETHES</td>
<td>SOMETHES</td>
</tr>
<tr>
<td>Point B</td>
<td>ALWAYS</td>
<td>SOMETHES</td>
</tr>
<tr>
<td>Point C</td>
<td>ALWAYS</td>
<td>ALWAYS</td>
</tr>
<tr>
<td>Point D</td>
<td>ALWAYS</td>
<td>SOMETHES</td>
</tr>
<tr>
<td>Point E</td>
<td>ALWAYS</td>
<td>NEVER</td>
</tr>
<tr>
<td>Point F</td>
<td>SOMETHES</td>
<td>ALWAYS</td>
</tr>
<tr>
<td>Point G</td>
<td>NEVER</td>
<td>ALWAYS</td>
</tr>
</tbody>
</table>
while loop variations

reading: 5.4
self-checks: #22-24
exercises: #6
The **do/while** loop

**do/while loop**: Executes statements repeatedly while a condition is **true**, testing it at the **end** of each repetition.

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

**Example:**

```java
// prompt until the user gets the right password
String phrase;
do {
    System.out.print("Password: ");
    phrase = console.next();
} while (!phrase.equals("abracadabra"));
```
**do/while flow chart**

- How does this differ from the while loop?
  - The controlled **statement(s)** will always execute the first time, regardless of whether the **test** is **true** or **false**.

![Flow Chart Image]

Copyright 2008 by Pearson Education
**do/while question**

- Modify the previous **Dice program to use do/while**.
  - Example log of execution:
    
    \[
    \begin{align*}
    2 + 4 &= 6 \\
    3 + 5 &= 8 \\
    5 + 6 &= 11 \\
    1 + 1 &= 2 \\
    4 + 3 &= 7
    \end{align*}
    \]
    
    You won after 5 tries!

- Modify the previous **Sentinel program to use do/while**.
  - Is **do/while** a good fit for solving this problem?
do/while answer

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

class Dice {
    public static void main(String[] args) {
        Random rand = new Random();
        int tries = 0;
        int sum;
        do {
            int roll1 = rand.nextInt(6) + 1;
            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.
  - Such loops are often called "forever" loops because their header's boolean test is often changed to a trivial `true`.

```java
while (true) {
    statement(s);
    if (test) {
        break;
    }
    statement(s);
}
```

- **break** is bad style! Do not use it on CSE 142 homework.
Sentinel loop with `break`

- A working sentinel loop solution using `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;
    } else {
        sum = sum + number;  // number != -1 here
    }
}
System.out.println("The total was "+ sum);
```