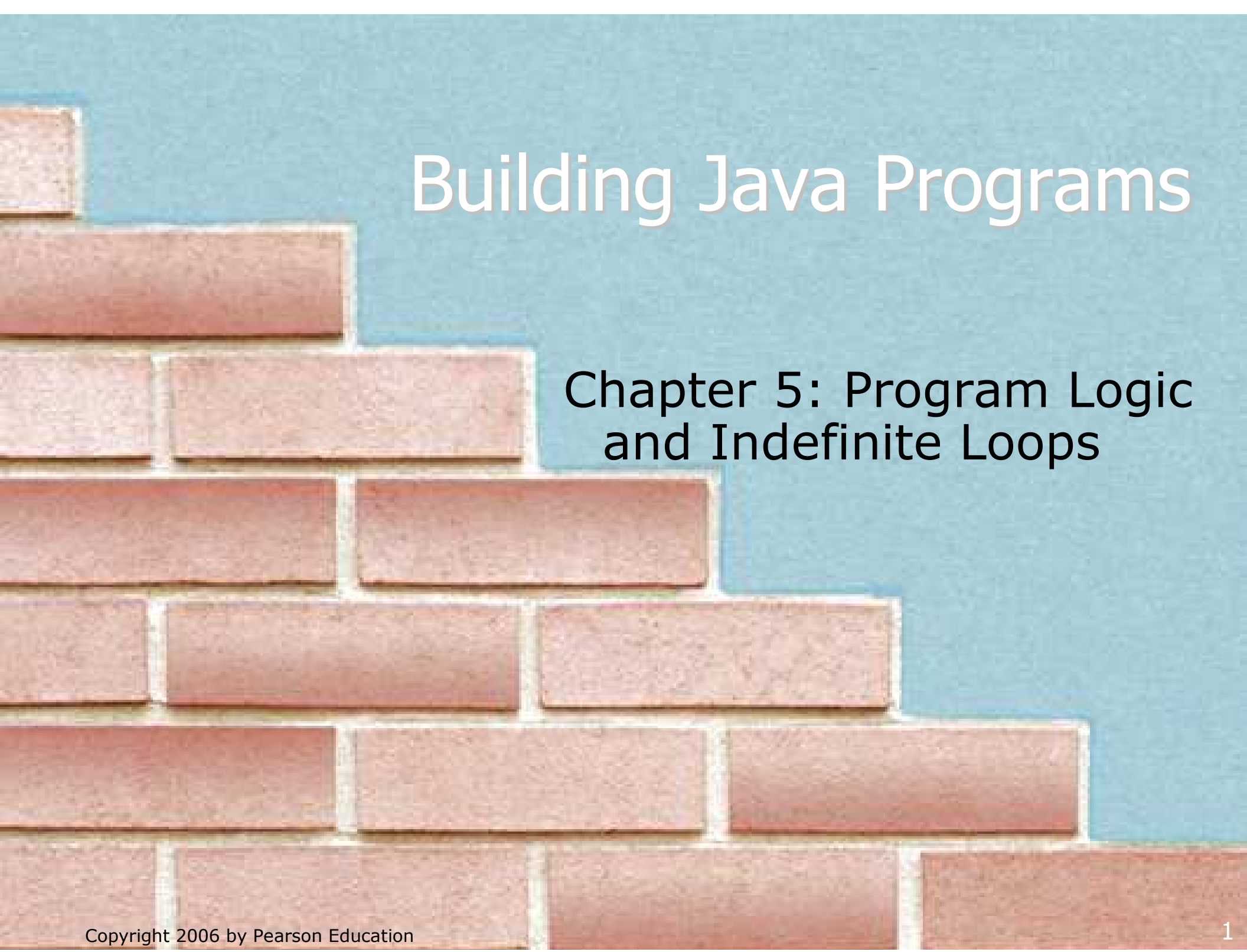
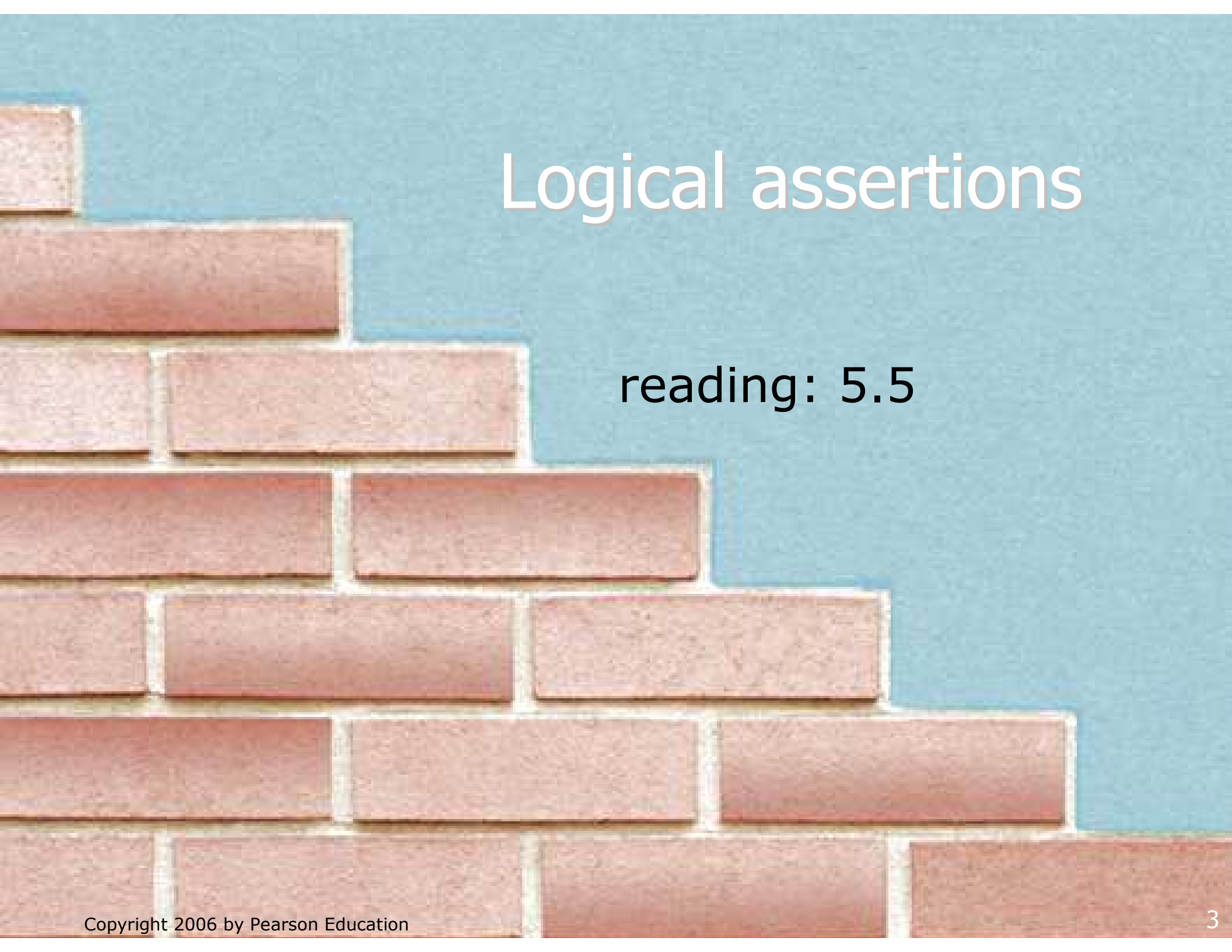


Lecture outline

- logical assertions

- indefinite loop variations
  - the `do/while` loop
  - the `break` statement



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)*

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.

```
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)
```

Assertion example 1

```
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.

	next == 0	prev == 0	next == prev
Point A	SOMETIMES	ALWAYS	SOMETIMES
Point B	NEVER	SOMETIMES	SOMETIMES
Point C	NEVER	NEVER	ALWAYS
Point D	SOMETIMES	NEVER	SOMETIMES
Point E	ALWAYS	SOMETIMES	SOMETIMES

Assertion example 2

```
public static void mystery(int x, int y) {
    int z = 0;

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

        // Point C
        z++;

        // Point D
    }

    // Point E
    System.out.println(
        z + " " + x);
}
```

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

	x < y	x == y	z == 0
Point A	SOMETIMES	SOMETIMES	ALWAYS
Point B	NEVER	SOMETIMES	SOMETIMES
Point C	SOMETIMES	SOMETIMES	SOMETIMES
Point D	SOMETIMES	SOMETIMES	NEVER
Point E	ALWAYS	NEVER	SOMETIMES

Assertion example 3

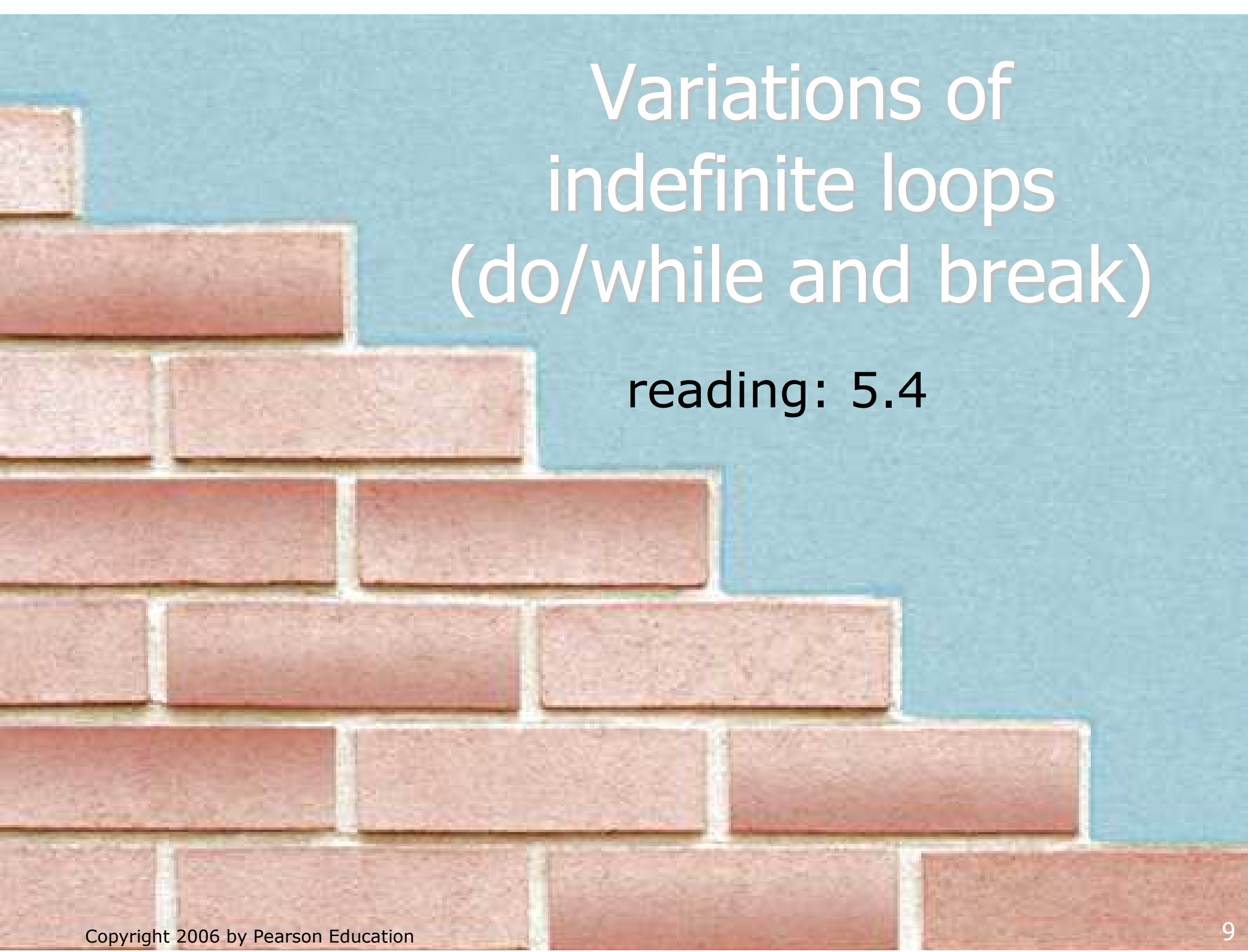
```
// Assumes y >= 0, and returns x^y
public static int pow(int x, int y) {
    int prod = 1;

    // Point A
    while (y > 0) {
        // Point B
        if (y % 2 == 0) {
            // Point C
            x *= x;
            y /= 2;

            // Point D
        } else {
            // Point E
            prod *= x;
            y--;
            // Point F
         }
         // Point G
     }
     // Point H
     return prod;
 }
```

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

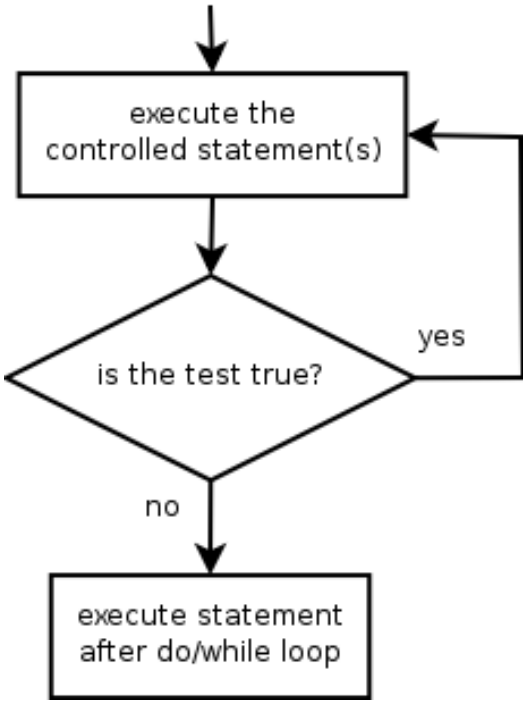
	y == 0	y % 2 == 0
Point A	SOMETIMES	SOMETIMES
Point B	NEVER	SOMETIMES
Point C	NEVER	ALWAYS
Point D	NEVER	SOMETIMES
Point E	NEVER	NEVER
Point F	SOMETIMES	ALWAYS
Point G	SOMETIMES	SOMETIMES
Point H	ALWAYS	ALWAYS



The do/while loop

- **`do/while` loop**: Executes statements repeatedly while a condition is `true`, testing it at the *end* of each repetition.
  - Similar to a `while` loop, except that its body statement(s) will always execute the first time.

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



  - Example:

```
// roll until we get a # other than 3
Random rand = new Random();
int dice;
do {
    dice = rand.nextInt();
} while (dice == 3);
```

do/while question

- Modify the previous Sentinel program to use a `do/while`.

- Modify the previous dice program to use a `do/while` loop.
  - Example log of execution:

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

do/while solution

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

public class Roll {
    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!");
    }
}
```

"Forever" loop with 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`.

- "forever" loop, general syntax:

```
while (true) {
    <statement(s)> ;

    if (<condition>) {
        break;
    }

    <statement(s)> ;
}
```

  - Exercise: Modify our Sentinel program to use `break`.

Sentinel loop with break

- A working sentinel loop solution using `break`:

```
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;
    }
    sum = sum + number;     // number != -1 here
}

System.out.println("The total was " + sum);
```