

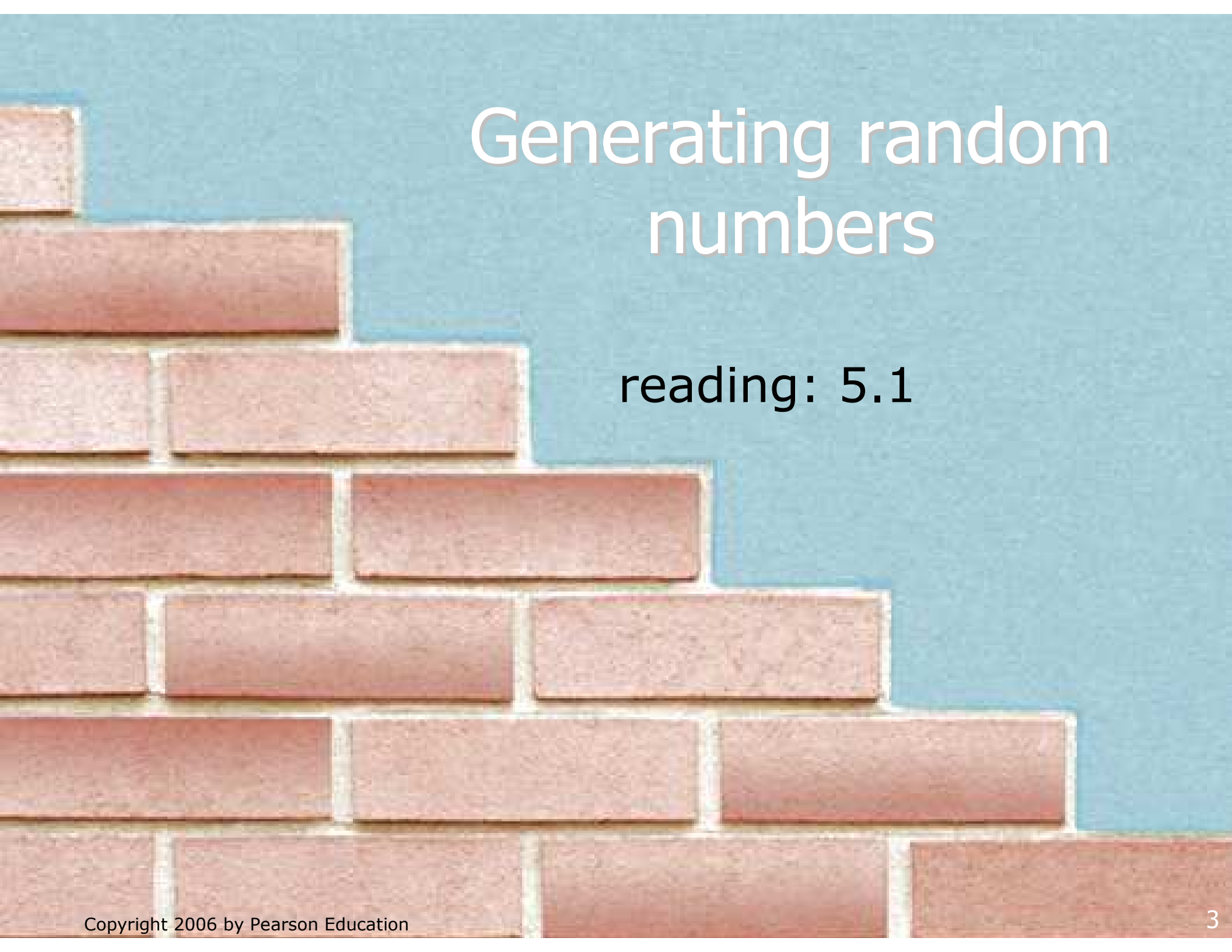
A brick wall on the left side of a blue background. The bricks are reddish-brown with white mortar lines. The wall is partially visible, extending from the left edge towards the center of the frame.

Building Java Programs

Chapter 5: Program Logic and Indefinite Loops

Lecture outline

- generating random numbers
- Boolean logic
 - boolean expressions and variables
 - logical operators

A brick wall on the left side of a blue background. The bricks are reddish-brown with white mortar. The wall is partially visible, extending from the left edge towards the center of the frame.

Generating random numbers

reading: 5.1

The Random class

- Random objects generate pseudo-random numbers.
 - Class Random is found in the `java.util` package.

```
import java.util.*;
```

Method name	Description
<code>nextInt()</code>	returns a random integer
<code>nextInt(max)</code>	returns a random integer in the range $[0, max)$ in other words, 0 to $max-1$ inclusive
<code>nextDouble()</code>	returns a random real number in the range $[0.0, 1.0)$

- Example:

```
Random rand = new Random();  
int randomNumber = rand.nextInt(10);  
// randomNumber has a random value between 0 and 9
```

Generating random numbers

- Common usage: to get a random number from 1 to N
 - Example: A random integer between 1 and 20, inclusive:

```
int n = rand.nextInt(20) + 1;
```

- To get a number in arbitrary range [min , max]:

```
nextInt( <size of range> ) + <min>
```

where **<size of range>** is **<max>** - **<min>** + 1

- Example: A random integer between 5 and 10 inclusive:

```
int n = rand.nextInt(6) + 5;
```

Random questions

- Given the following declaration, how would you get:

```
Random rand = new Random();
```

- A random number between 1 and 100 inclusive?
- A random number between 50 and 100 inclusive?
- A random number between 4 and 17 inclusive?

Random answers

- Given the following declaration, how would you get:

```
Random rand = new Random();
```

- A random number between 1 and 100 inclusive?

```
int random1 = rand.nextInt(100) + 1;
```

- A random number between 50 and 100 inclusive?

```
int random2 = rand.nextInt(51) + 50;
```

- A random number between 4 and 17 inclusive?

```
int random3 = rand.nextInt(14) + 4;
```

Other uses of Random

- Random can be used to pick between arbitrary choices

- Code to randomly play Rock-Paper-Scissors:

```
int r = rand.nextInt(3);
if (r == 0) {
    System.out.println("Rock");
} else if (r == 1) {
    System.out.println("Paper");
} else {
    System.out.println("Scissors");
}
```

- Random can also be used with double

- `nextDouble` method returns a double between 0.0 and 1.0

- Example: Get a random GPA value between 1.5 and 4.0:

```
double randomGpa = rand.nextDouble() * 2.5 + 1.5;
```


Random question

- Write a program that simulates rolling of two six-sided dice until their combined result comes up as 7.

- Example log of execution:

$$2 + 4 = 6$$

$$3 + 5 = 8$$

$$5 + 6 = 11$$

$$1 + 1 = 2$$

$$4 + 3 = 7$$

You won after 5 tries!

Random answer

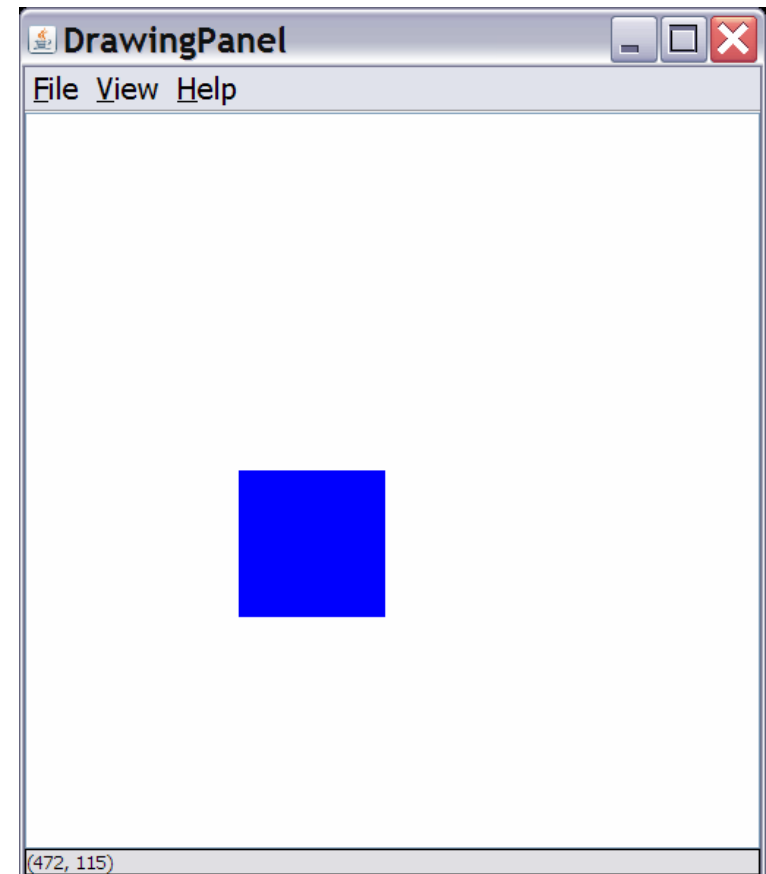
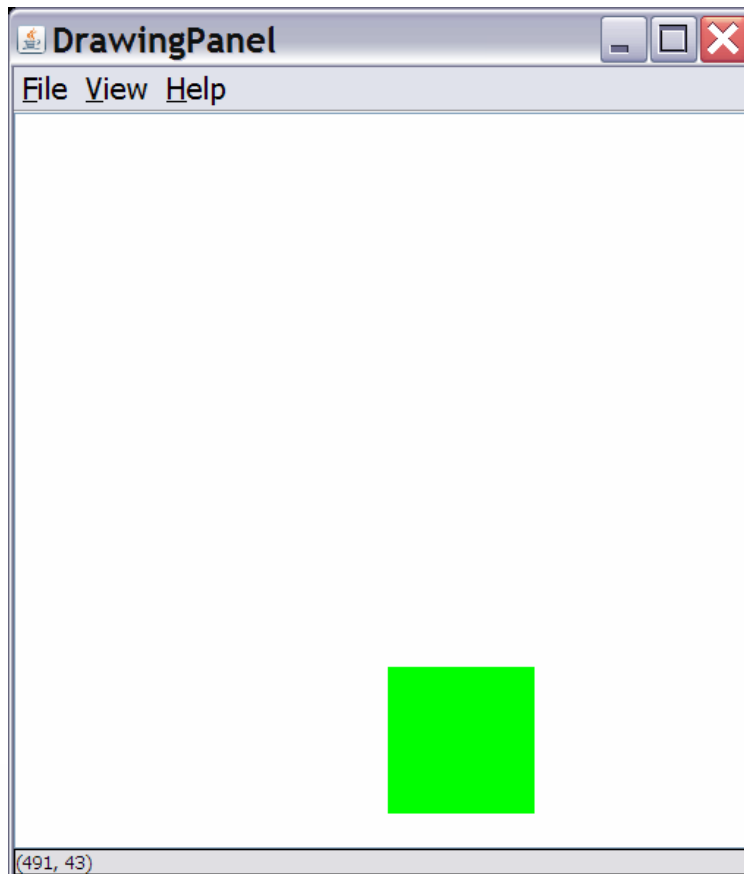
```
// Rolls two dice until a sum of 7 is reached.
import java.util.*;

public class Roll {
    public static void main(String[] args) {
        Random rand = new Random();
        int sum = 0;
        int tries = 0;
        while (sum != 7) {
            int roll1 = rand.nextInt(6) + 1;
            int roll2 = rand.nextInt(6) + 1;
            sum = roll1 + roll2;
            System.out.println(roll1 + " + " + roll2 + " = " + sum);
            tries++;
        }

        System.out.println("You won after " + tries + " tries!");
    }
}
```

Random drawing question

- Write a program that draws a 100x100 rectangle at a random (x, y) position within a 500x500 `DrawingPanel`. The rectangle's color should be randomly chosen between red, green, and blue.



Random drawing answer

```
// Draws a random 100x100 rectangle in a random color.
import java.awt.*;
import java.util.*;

public class RandomRectangle {
    public static void main(String[] args) {
        DrawingPanel panel = new DrawingPanel(500, 500);
        Graphics g = panel.getGraphics();
        Random rand = new Random();

        // choose random location
        Point rectPoint = new Point();
        rectPoint.x = rand.nextInt(500);
        rectPoint.y = rand.nextInt(500);

        // choose random color
        int randomColor = rand.nextInt(3);
        if (randomColor == 0) {
            g.setColor(Color.RED);
        } else if (randomColor == 1) {
            g.setColor(Color.GREEN);
        } else {
            g.setColor(Color.BLUE);
        }

        g.fillRect(rectPoint.x, rectPoint.y, 100, 100);
    }
}
```

A brick wall on the left side of a blue background. The bricks are reddish-brown with white mortar. The wall is partially visible, extending from the left edge towards the center of the frame.

Boolean logic

reading: 5.2

Type boolean

- **boolean**: A primitive type to represent logical values.
 - A boolean expression produces either `true` or `false`.
 - A **<condition>** in an `if`, `for`, `while` is a boolean expression.

- Examples:

```
boolean minor = (age < 21);  
boolean expensive = (iPhonePrice > 500.00);  
boolean iLoveCS = true;  
  
if (minor) {  
    System.out.println("Can't purchase alcohol!");  
}
```

- You can create boolean variables, pass boolean parameters, return boolean values from methods, ...

Methods that return boolean

- There are methods in Java that return `boolean` values.
 - A call to one of these methods can be used as a **<condition>** in a loop or `if` statement.
 - Examples:

```
Scanner console = new Scanner(System.in);  
System.out.print("Type your name: ");  
String line = console.next();
```

```
if (line.startsWith("Dr.")) {  
    System.out.println("Will you marry me?");  
} else if (line.endsWith(", Esq.")) {  
    System.out.println("And I am Ted 'Theodore' Logan!");  
}
```

Writing boolean methods

- Methods can return a boolean result.

```
public static boolean bothOdd(int n1, int n2) {  
    if (n1 % 2 != 0 && n2 % 2 != 0) {  
        return true;  
    } else {  
        return false;  
    }  
}
```

- Calls to such methods can be used as conditions:

```
if (bothOdd(7, 13)) {  
    ...  
}
```


Boolean questions

- Modify our previous *Primes* program to use methods with return values to tell whether or not a number is prime.

- Example output of primes up to 50:

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]
```

- Modify our previous *Rhyme* program to use methods with return values to tell whether the two words rhyme and/or alliterate.

- Example log of execution:

```
Type two words: car STAR
```

```
They rhyme!
```

Boolean answer

```
// Determines whether two words rhyme and/or start with the same letter.
import java.util.*;

public class Rhyme {
    public static void main(String[] args) {
        Scanner console = new Scanner(System.in);

        System.out.print("Type two words: ");
        String word1 = console.next();
        String word2 = console.next();

        if (rhyme(word1, word2)) {
            System.out.println("They rhyme!");
        }

        if (alliterate(word1, word2)) {
            System.out.println("They alliterate!");
        }
    }

    // Returns true if s1 and s2 end with the same two letters.
    public static boolean rhyme(String s1, String s2) {
        return s2.length() >= 2 &&
            s1.endsWith(s2.substring(s2.length() - 2));
    }

    // Returns true if s1 and s2 start with the same letter.
    public static boolean alliterate(String s1, String s2) {
        return s1.startsWith(s2.substring(0, 1));
    }
}
```

Boolean question

- Modify the rectangle program to draw randomly placed/colored 10x10 rectangles until it draws 20 red ones.
 - Break up your program using static methods.
 - Print a line of output each time a red rectangle is drawn:

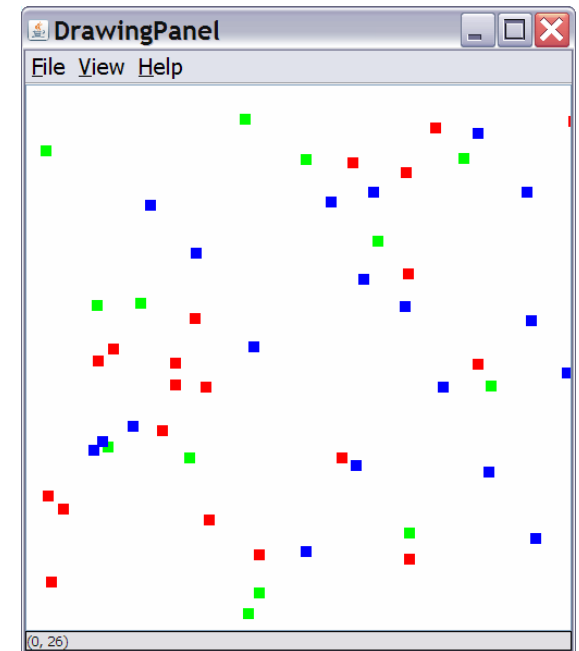
```
Drew red #1 at (120, 312)
```

```
Drew red #2 at (285, 337)
```

```
Drew red #3 at (410, 251)
```

```
Drew red #4 at (15, 372)
```

```
Drew red #5 at (61, 248)
```



- Consider making the `DrawingPanel` animate by calling its `sleep` method between each rectangle drawn.

Boolean answer 1

```
// Draws randomly placed/colored 'confetti' rectangles on a DrawingPanel.
import java.awt.*;
import java.util.*;

public class Confetti {
    public static final boolean DEBUG = true; // turns on/off debug printlns

    public static void main(String[] args) {
        DrawingPanel panel = new DrawingPanel(500, 500);
        Graphics g = panel.getGraphics();
        Random rand = new Random();

        // repeat until 20 red rectangles are drawn
        int redCount = 0;
        Point staticPoint = new Point();
        while (redCount < 20) {
            if (randomRect(g, rand, staticPoint)) {
                redCount++;
                System.out.println("Drew red #" + redCount + " at (" +
                    staticPoint.x + ", " + staticPoint.y + ")");
            }
            panel.sleep(400); // pause for animation
        }
    }

    ...
}
```

Boolean answer 2

...

```
// Draws a rectangle on the panel in a random place/color.
// Returns true if the rectangle was red.
public static boolean randomRect(Graphics g, Random r, Point p) {
    // choose random location
    p.x = r.nextInt(500);
    p.y = r.nextInt(500);

    // choose random color
    int randomColor = r.nextInt(3);
    if (randomColor == 0) {
        g.setColor(Color.RED);
    } else if (randomColor == 1) {
        g.setColor(Color.GREEN);
    } else {
        g.setColor(Color.BLUE);
    }

    g.fillRect(p.x, p.y, 10, 10);

    return (randomColor == 0);
}
}
```

Boolean flags

- **boolean flag:** A boolean value, often a class constant, that can be used to signal program behavior.

```
public static final boolean <name> = <value>;
```

- Boolean flags are useful to enable/disable program behavior, such as `println` messages you only sometimes want to see.
- Example:

```
public static final boolean SHOW_OUTPUT = true;
```

```
...
```

```
if (SHOW_OUTPUT) { // show my variables' values
    System.out.println(a + " " + b + " " + c);
}
```

- Exercise: Add a boolean flag to the colored rectangle program.

Boolean flag answer


```
// Draws randomly placed/colored 'confetti' rectangles on a DrawingPanel.
import java.awt.*;
import java.util.*;

public class Confetti {
    public static final boolean DEBUG = true; // turns on/off debug printlns

    public static void main(String[] args) {
        DrawingPanel panel = new DrawingPanel(500, 500);
        Graphics g = panel.getGraphics();
        Random rand = new Random();

        // repeat until 20 red rectangles are drawn
        int redCount = 0;
        Point staticPoint = new Point();
        while (redCount < 20) {
            if (randomRect(g, rand, staticPoint)) {
                redCount++;
                if (DEBUG) { // print message for debugging
                    System.out.println("Drew red #" + redCount + " at (" +
                        staticPoint.x + ", " + staticPoint.y + ")");
                }
            }
            panel.sleep(400); // pause for animation
        }
    }
}

...
```

A brick wall with a blue background behind it. The bricks are arranged in a staggered pattern, with some missing or cut off on the left side. The text is overlaid on the right side of the image.

Case study: Multiplication tutor

Random/while question

- Write a multiplication tutor program. Example log of execution:

14 * 8 = 112

Correct!

5 * 12 = 60

Correct!

8 * 3 = 24

Correct!

5 * 5 = 25

Correct!

20 * 14 = 280

Correct!

19 * 14 = 256

Incorrect; the answer was 266

You solved 5 correctly.

Random/while answer

```
// Asks the user to do multiplication problems and scores them.
import java.util.*;

public class MultTutor {
    public static void main(String[] args) {
        introduction();
        Scanner console = new Scanner(System.in);
        Random rand = new Random();
        int num1 = 0;
        int num2 = 0;
        int guess = 0;
        int correct = 0;

        // loop until user gets one wrong
        while (guess == num1 * num2) {
            // pick two random numbers between 1 and 20 inclusive
            num1 = rand.nextInt(20) + 1;
            num2 = rand.nextInt(20) + 1;

            System.out.print(num1 + " * " + num2 + " = ");
            int guess = console.nextInt();
            if (guess == num1 * num2) {
                System.out.println("Correct!");
            } else {
                System.out.println("Incorrect; the answer was " + (num1 * num2));
            }
        }

        System.out.println("You solved " + correct + " correctly.");
    }
}
```

Boolean question

- Modify the previous multiplication tutor program to use a static method that returns a boolean value.

$$14 * 8 = \underline{112}$$

Correct!

$$5 * 12 = \underline{60}$$

Correct!

$$8 * 3 = \underline{24}$$

Correct!

$$5 * 5 = \underline{25}$$

Correct!

$$20 * 14 = \underline{280}$$

Correct!

$$19 * 14 = \underline{256}$$

Incorrect; the answer was 266

You solved 5 correctly.

Boolean answer

```
import java.util.*;

// Asks the user to do multiplication problems and scores them.
public class MultTutor {
    public static void main(String[] args) {
        introduction();
        Scanner console = new Scanner(System.in);
        Random rand = new Random();

        // loop until user gets one wrong
        int correct = 0;
        while (askQuestion(console, rand)) {
            correct++;
        }

        System.out.println("You solved " + correct + " correctly.");
    }

    ...
}
```

Boolean answer 2

...

```
// Asks the user one multiplication problem,  
// returning true if they get it right and false if not.  
public static boolean askQuestion(Scanner console, Random rand) {  
    // pick two random numbers between 1 and 20 inclusive  
    int num1 = rand.nextInt(20) + 1;  
    int num2 = rand.nextInt(20) + 1;  
  
    System.out.print(num1 + " * " + num2 + " = ");  
    int guess = console.nextInt();  
    if (guess == num1 * num2) {  
        System.out.println("Correct!");  
        return true;  
    } else {  
        System.out.println("Incorrect; the correct answer was " +  
            (num1 * num2));  
        return false;  
    }  
}
```