Nesting Loops

- We can produce our rows and columns of tiles, using nested for loops.

```java
for (int row = 0; row < 8; row++)
{
    for (int col = 0; col < 8; col++)
    {
        // in even-numbered rows (0, 2, 4, ...)
        if (row % 2 == 0)
        {
            // in even-numbered rows (0, 2, 4, ...)
            if (col % 2 == 0)
            {
                rect.setBackground(Color.red);
            }
            // in odd-numbered rows (1, 3, 5, ...)
            else
            {
                if (col % 2 == 1)
                {
                    rect.setBackground(Color.red);
                }
            }
        }
        // in odd-numbered rows (1, 3, 5, ...)
        else
        {
            // in odd-numbered rows (1, 3, 5, ...)
            if (col % 2 == 1)
            {
                rect.setBackground(Color.red);
            }
        }
        window.add(rect);
    }
}
```

- Nested Loop declaration – we loop over a second variable inside the original, going row by row, then column by column for each row.
- Main work – Here, we use both variables, in order to generate the full board.

Review: Tiling with a for Loop

- Again, we produce our row of tiles, using for

```java
for (int col = 0; col < numSquares; col++)
{
    int x = col * size;
    Rectangle rect = new Rectangle(...);
    if (col % 2 == 0)
    {
        rect.setBackground(Color.red);
    }
    window.add(rect);
}
```

- Loop declaration – this time, the initialization of loop variable, condition, and progress all happen on the same line, at the start.
- Main work – Code to be repeatedly executed.

Understanding Nested Loops

- We can place any type of loop inside of another.
- When we do so, we work from the outside to the inside.

```java
for (int i = 0; i < 4; i++)
{
    for (int j = 0; j < 2; j++)
    {
        System.out.println(i + " + " + j);
    }
}
```

- Outer loop (i): runs 4 times
- Inner loop (j): runs 2 times
- 8 total: 4 i-loops * 2 j-loops
Understanding Nested Loops

Here, we reverse the values, so i only goes to 2, and j goes to 4 instead.

We get the same number of loops, but the output result is different.

```java
for (int i = 0; i < 2; i++) {
    for (int j = 0; j < 4; j++)
        System.out.println(i + " " + j);
}
```

outer loop (i): runs 2 times
inner loop (j): runs 4 times
8 total: 2 i-loops * 4 j-loops

This Week:

- **Meetings this week:**
  - Monday/Wednesday: regular classroom
  - Tuesday: in the CS Lab (16 Wing)

- **Program 03:** due 11:59 PM, Thursday 03 October

- **Midterm:** Monday 07 October

- **Practice Midterm:** Available on my website

- **Office Hours:** Wing 212
  - Monday/Friday: 2:15 PM–3:15 PM
  - Tuesday: 1:30 PM–2:30 PM
  - Wednesday: 12:05 PM–1:00 PM

- **Lab and Tutor Hours:** On my website

---

Exercise: Prime Numbers

- Recall that an integer is prime if it is divisible only by 1 and itself, otherwise it is composite.

- Write code that reads an integer from the user and determines if it is prime.
  - You may use a while loop or a for loop.
  - Develop an algorithm before you start writing Java code.

```java
// Example code
```