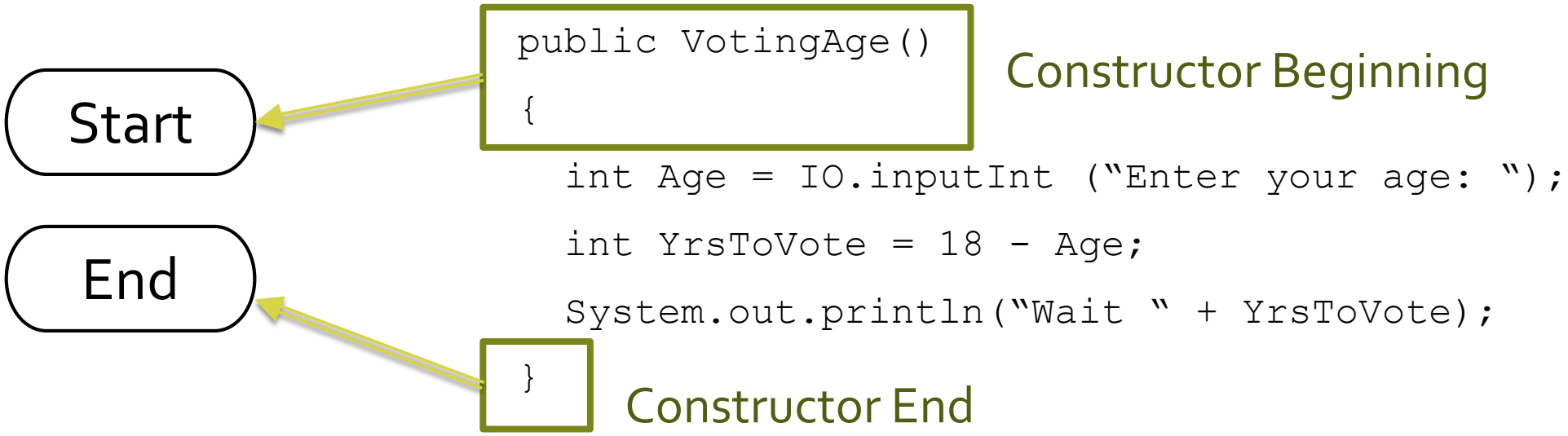
LOOP FLOW CHARTS

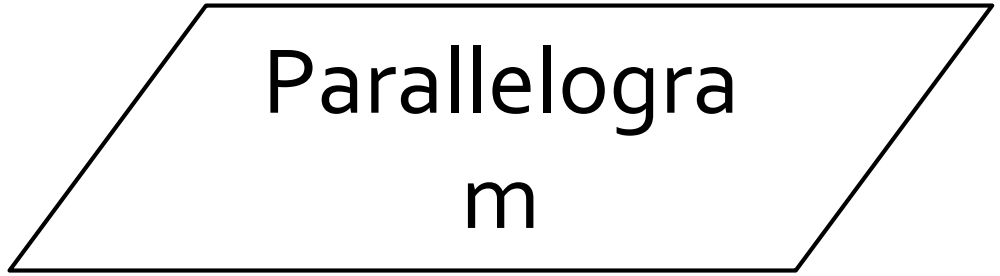
For Loops in Flow Charts

Oval

Terminal



- One Start and One End for each program. No more, no less.
- Only holds "Start" and "End"
- Only one arrow comes out of the Start. No arrows for End.



Input

IO lines

```
public VotingAge()
{
  int Age = IO.inputInt ("Enter your age: ");
  int YrsToVote = 18 - Age;
  System.out.println("Wait " + YrsToVote);
}
```

- Used for IO lines.
- Only write "Get" + variable name
- Don't write the prompt (question)
- Only one arrow comes out of it.

Rectangle

Processing

YrsToVote = 18-Age

```
public VotingAge()
{
   int Age = IO.inputInt ("Enter your age: ");
   int YrsToVote = 18 - Age;
   System.out.println("Wait " + YrsToVote);
}
```

Math lines

- Used for Math Lines.
- Leave out the variable type, but write everything else
- Only one arrow comes out of it.

Pencil

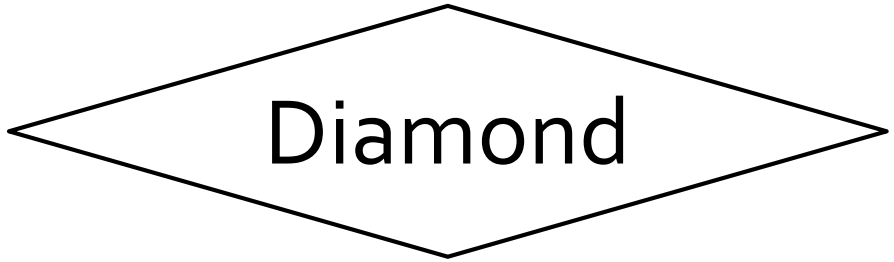
Output

"Wait " + YrsToVote

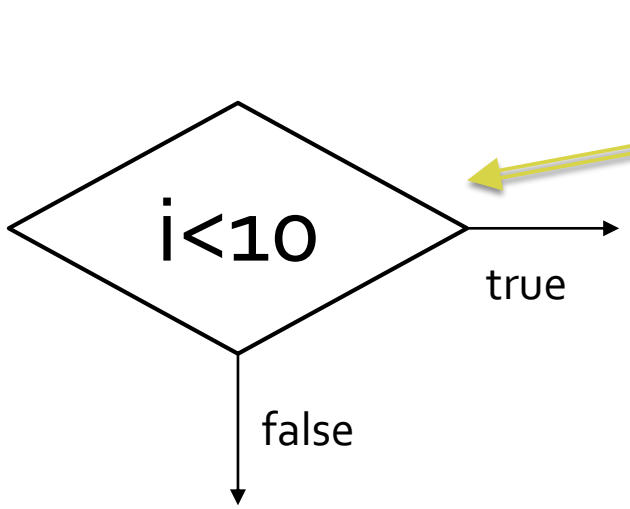
```
public VotingAge()
{
   int Age = IO.inputInt ("Enter your age: ");
   int YrsToVote = 18 - Age;
   System.out.println("Wait " + YrsToVote);
}
```

System.out lines

- Used for System.out.println lines.
- Leave out the System.out.println(); but write everything else
- Only one arrow comes out of it.



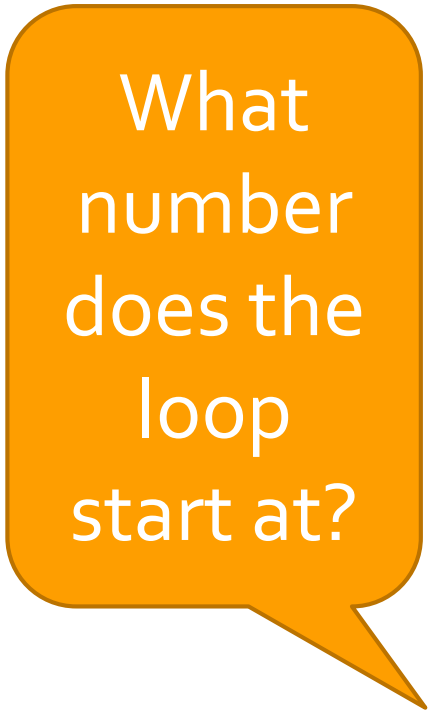
Boolean Expression




```
for (int i=0; i<10; i++)
{
   int Age = IO.inputInt ("Enter your age: ");
   int YrsToVote = 18 - Age;
   System.out.println("Wait " + YrsToVote);
}
```

- Used for Boolean Exressions – ifs && loops.
- Exactly 2 arrows come out of them. One is labelled true and one is labelled false.

```
for (int j = 1 ; j < 10 ; j++)
{
    System.out.print ("*");
}
```



```
for (int j = 1 ; j < 10 ; j++)
{
    System.out.print ("*");
}
```

It starts at 1.

```
for (int j = 1 ; j < 10 ; j++)
{
    System.out.print ("*");
}
```

What number does the loop stop at?

```
for (int j = 1 ; j < 10 ; j++)
{
    System.out.print ("*");
}
```

It stops below 10, at 9.

```
for (int j = 1 ; j < 10 ; j++)
{
    System.out.print ("*");
}
```

What does the loop do each time?

```
for (int j = 1 ; j < 10 ; j++)
{
    System.out.print ("*");
}
```

It prints a star.

That means 9 stars will be on the screen when it is done.

```
for (int j = 1 ; j < 10 ; j++)
{
    System.out.print ("*");
}
```

****

What shape do we start with?

```
for (int j = 1 ; j < 10 ; j++)
{
    System.out.print ("*");
}
```

Prints 9 stars

****

Start

We always start with START.

```
for (int j = 1 ; j < 10 ; j++)
{
    System.out.print ("*");
}
```

Prints 9 stars

****

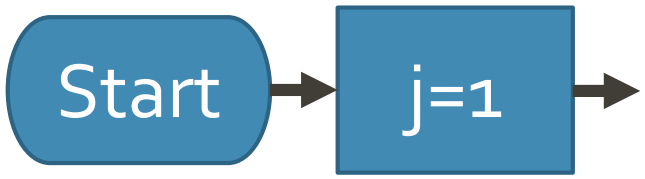
Start

Then, we initialize the loop stopping condition. What shape is needed for that?

```
for (int j = 1 ; j < 10 ; j++)
{
    System.out.print ("*");
}
```

Prints 9 stars

****

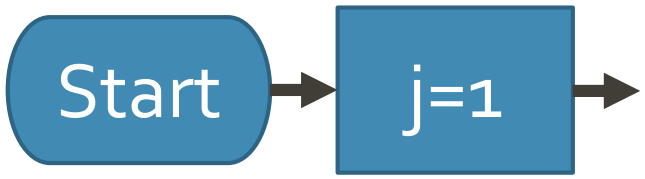


It is a math operation, so it goes in a rectangle.

```
for (int j = 1 ; j < 10 ; j++)
{
    System.out.print ("*");
}
```

Prints 9 stars

```
*********
```

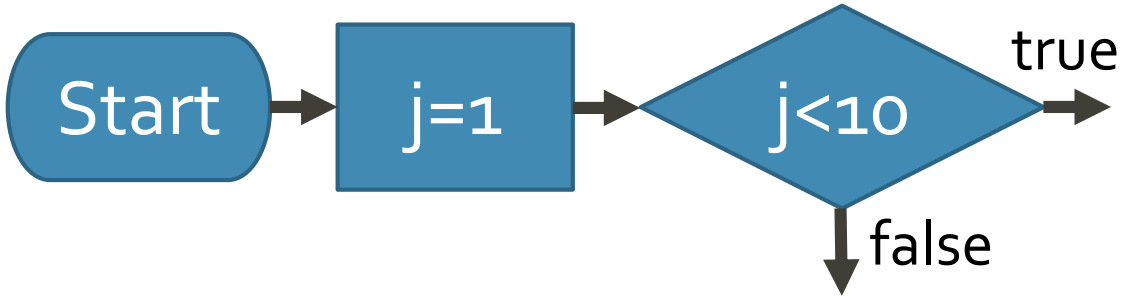


Next comes the stopping condition. It is a Boolean expression, so what shape is needed?

```
for (int j = 1 ; j < 10 ; j++)
{
    System.out.print ("*");
}
```

Prints 9 stars

****

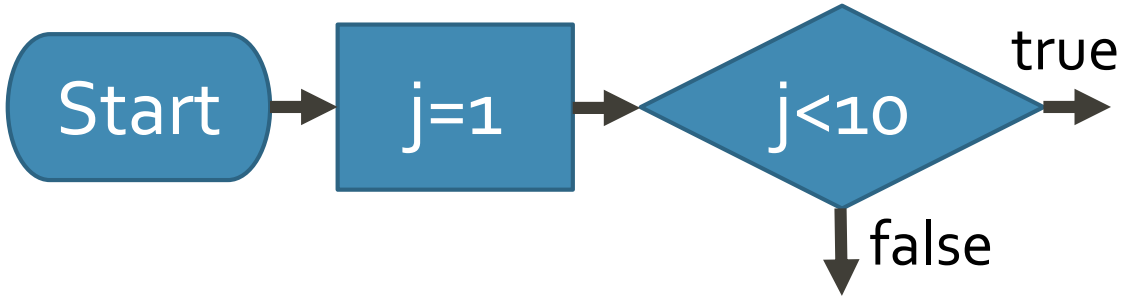


Boolean expressions go in diamonds.

```
for (int j = 1 ; j < 10 ; j++)
{
    System.out.print ("*");
}
```

Prints 9 stars

****

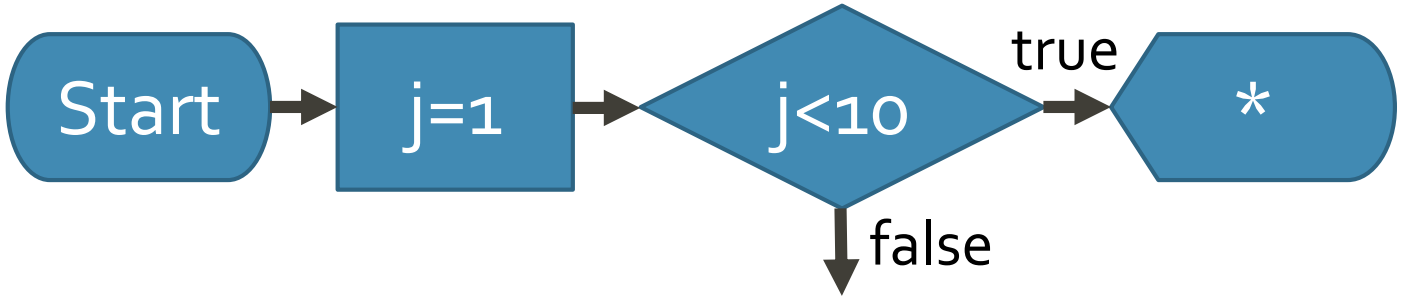


The next operation is the steps to repeat. What shape holds a print?

```
for (int j = 1 ; j < 10 ; j++)
{
    System.out.print ("*");
}
```

Prints 9 stars

****

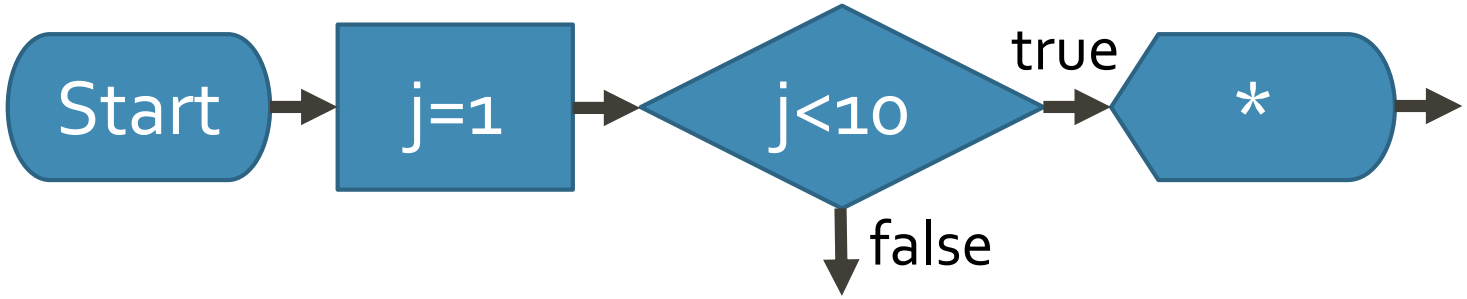


Pencil shape.

```
for (int j = 1 ; j < 10 ; j++)
{
    System.out.print ("*");
}
```

Prints 9 stars

****

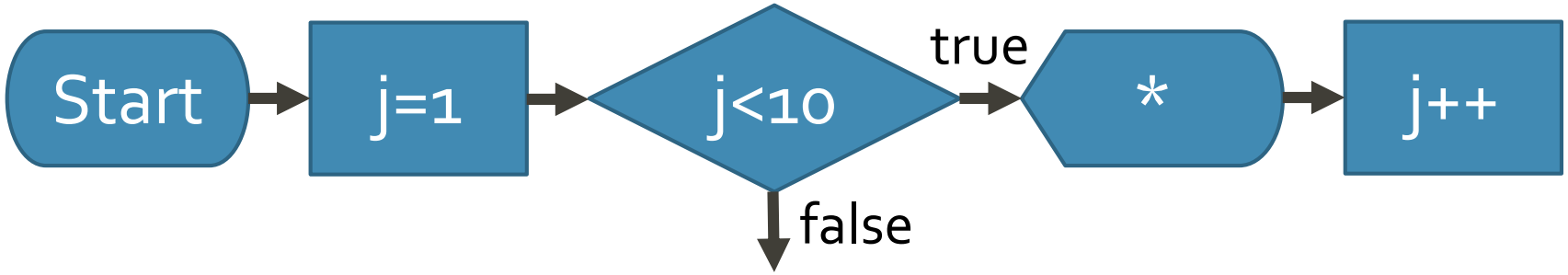


Then, we need the progress to the stopping condition.

```
for (int j = 1 ; j < 10 ; j++)
{
    System.out.print ("*");
}
```

Prints 9 stars

****

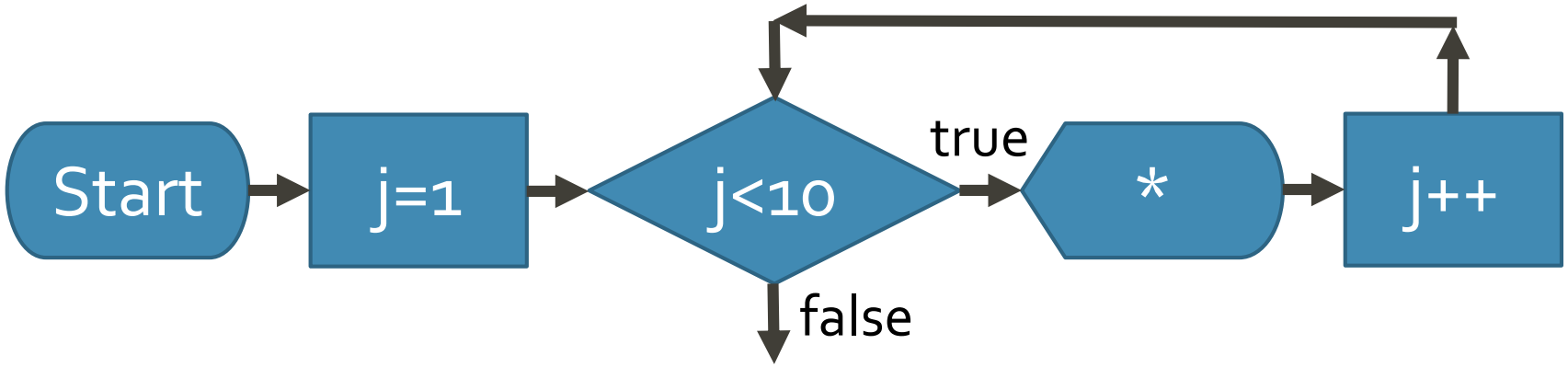


It is a math operation. Rectangle.

```
for (int j = 1 ; j < 10 ; j++)
{
    System.out.print ("*");
}
```

Prints 9 stars

```
*********
```

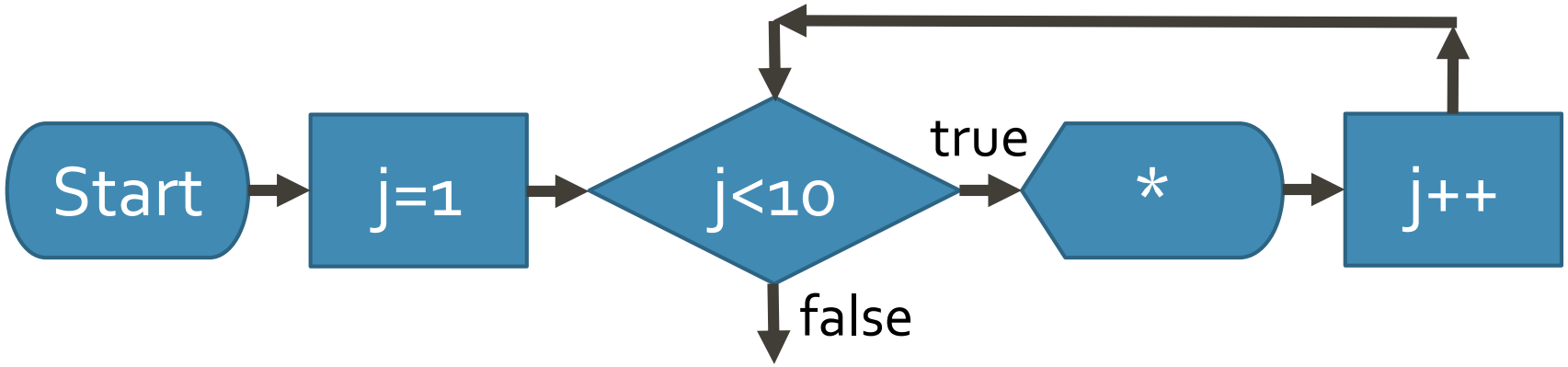


Then we need to finish up the loop … it goes back to the test.

```
for (int j = 1 ; j < 10 ; j++)
{
    System.out.print ("*");
}
```

Prints 9 stars

****

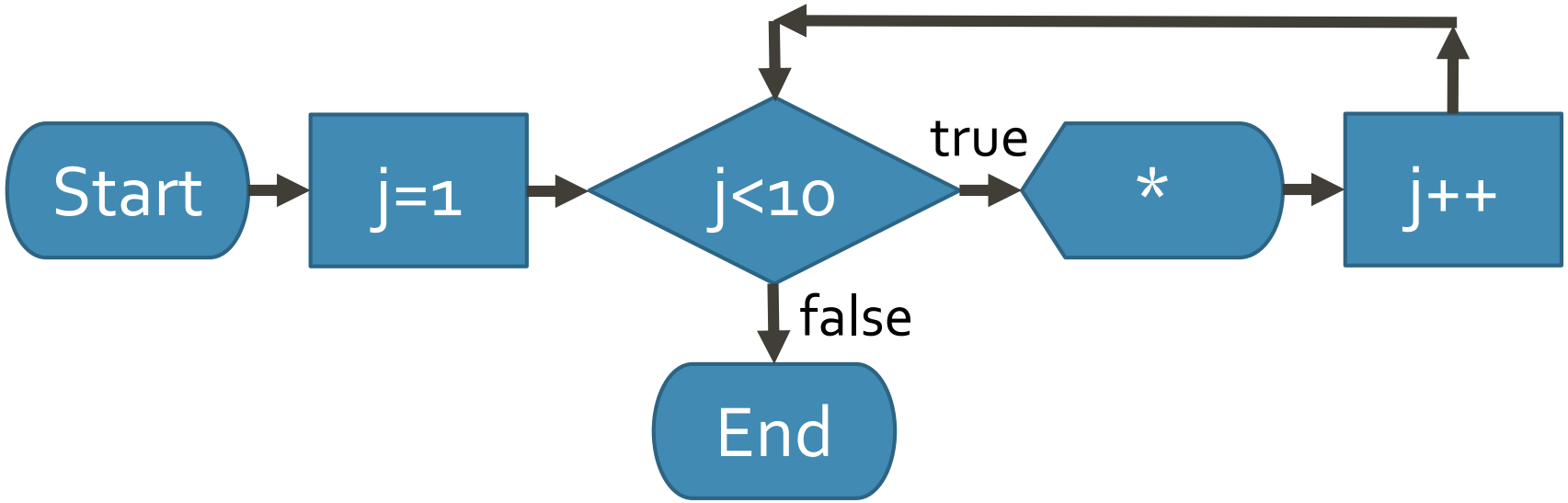


And how do we end it?

```
for (int j = 1 ; j < 10 ; j++)
{
    System.out.print ("*");
}
```

Prints 9 stars

****

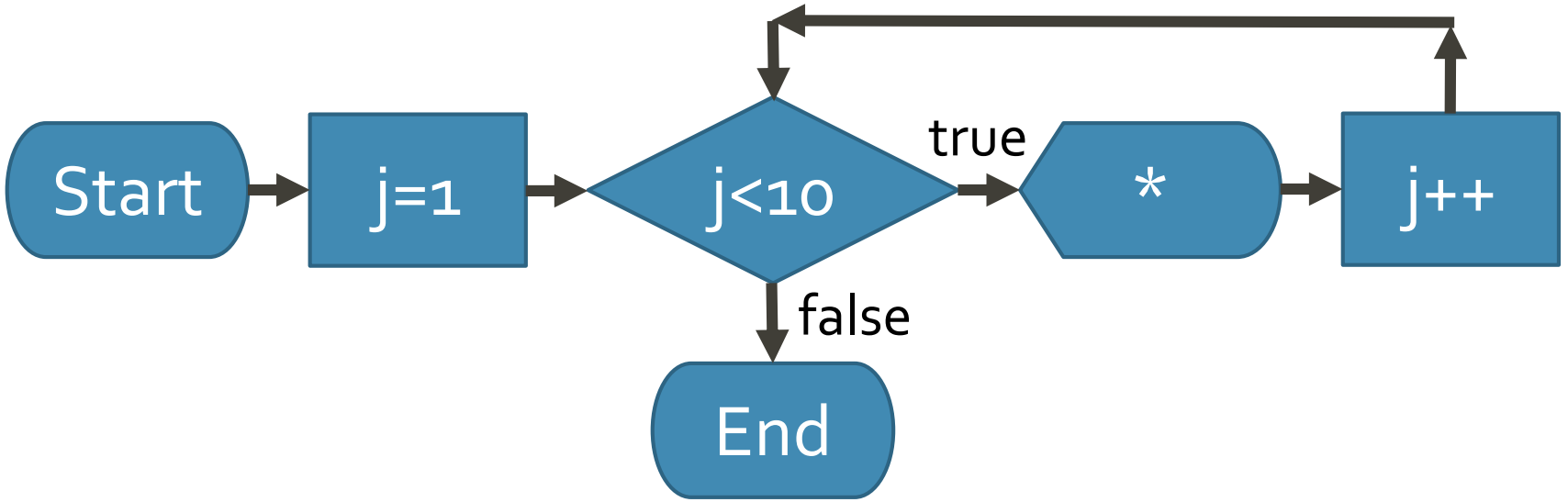


With end, naturally.

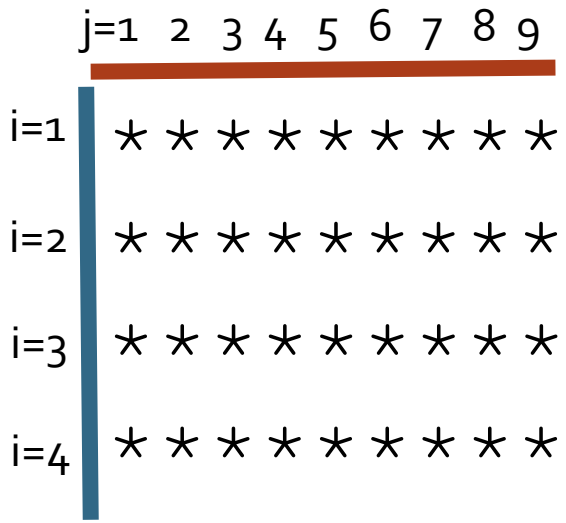
```
for (int j = 1 ; j < 10 ; j++)
{
    System.out.print ("*");
}
```

Prints 9 stars

****

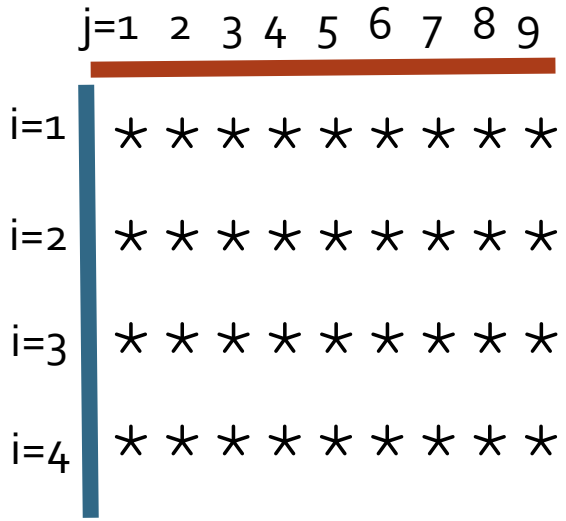


```
for (int i = 1 ; i < 5 ; i++)
{
   for (int j = 1 ; j < 10 ; j++)
   {
      System.out.print ("*");
   }
   System.out.println ("");
}
```



If we want a square, we put a loop in a loop.

```
for (int i = 1 ; i < 5 ; i++)
{
   for (int j = 1 ; j < 10 ; j++)
   {
      System.out.print ("*");
   }
   System.out.println ("");
}
```



So let's draw the flow chart.

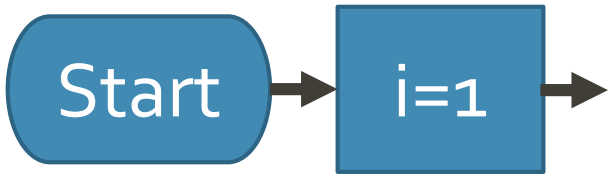
The one row is repeated many times by the second loop.

```
for (int i = 1 ; i < 5 ; i++)
{
   for (int j = 1 ; j < 10 ; j++)
   {
      System.out.print ("*");
   }
   System.out.println ("");
}
```

Start with the beginning.

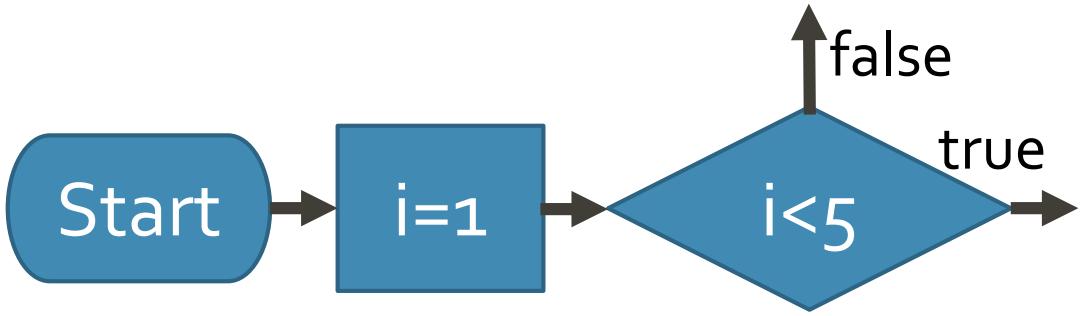
```
for (int i = 1 ; i < 5 ; i++)
{
   for (int j = 1 ; j < 10 ; j++)
   {
      System.out.print ("*");
   }
   System.out.println ("");
}
```

Add the first loop's initialize loop stopping variable.



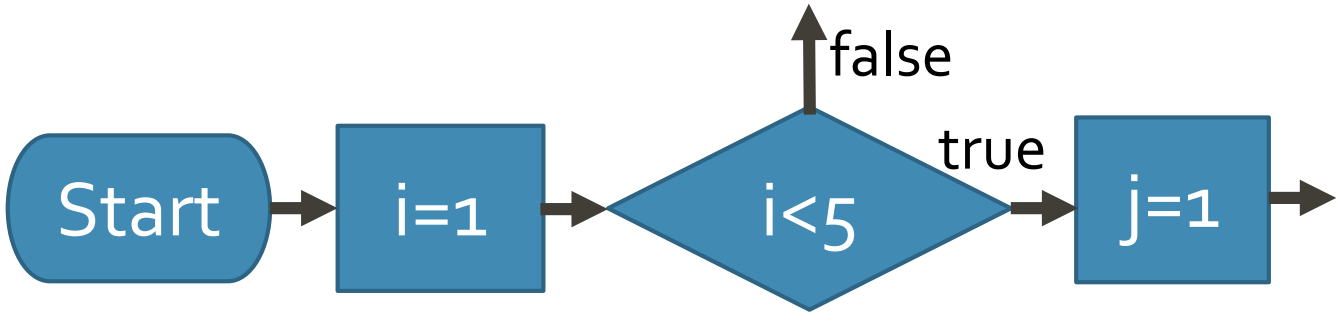
```
for (int i = 1 ; i < 5 ; i++)
{
   for (int j = 1 ; j < 10 ; j++)
   {
      System.out.print ("*");
   }
   System.out.println ("");
}
```

Add the first loop's stopping condition.



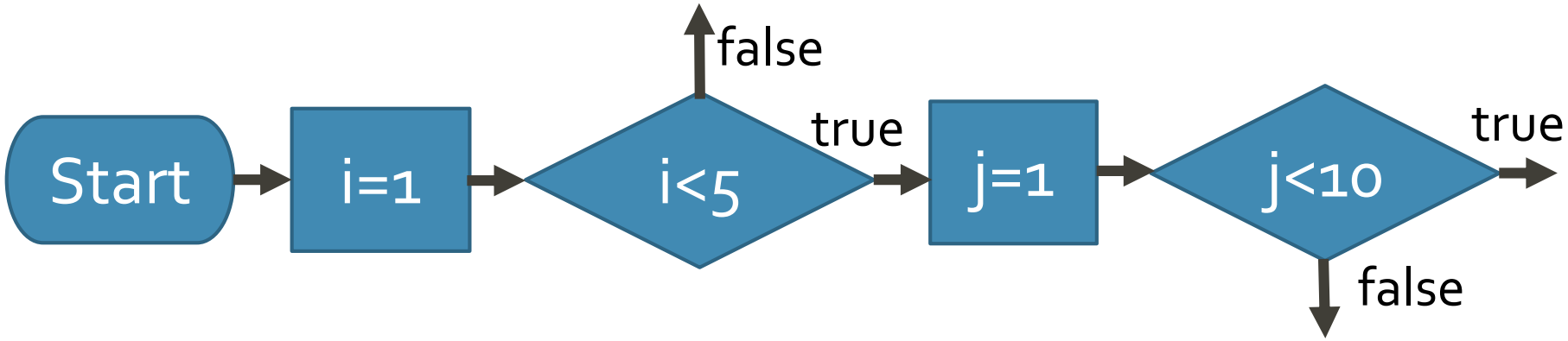
```
for (int i = 1 ; i < 5 ; i++)
{
   for (int j = 1 ; j < 10 ; j++)
   {
      System.out.print ("*");
   }
   System.out.println ("");
}
```

Add the second loop's initialize loop stopping variable.



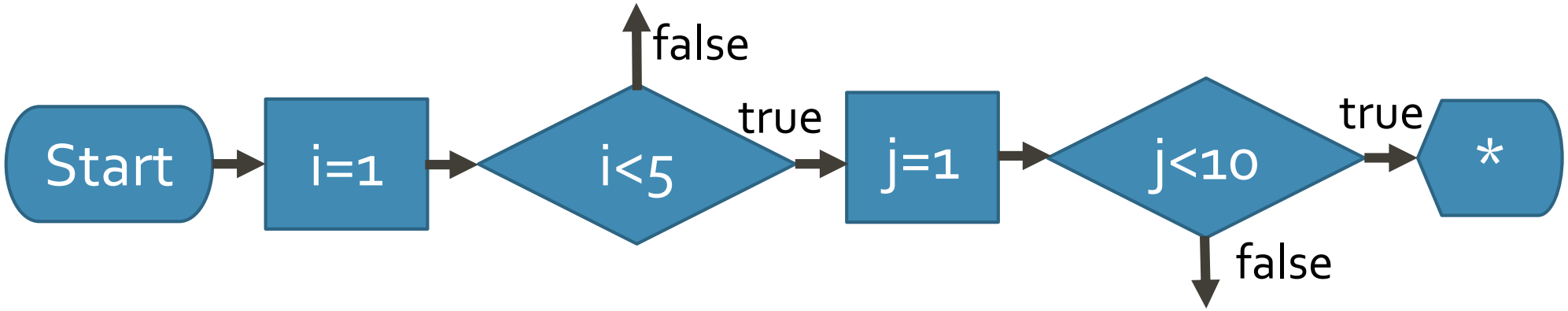
```
for (int i = 1 ; i < 5 ; i++)
{
  for (int j = 1 ; j < 10 ; j++)
  {
     System.out.print ("*");
  }
  System.out.println ("");
}
```

Add the second loop's stopping condition.



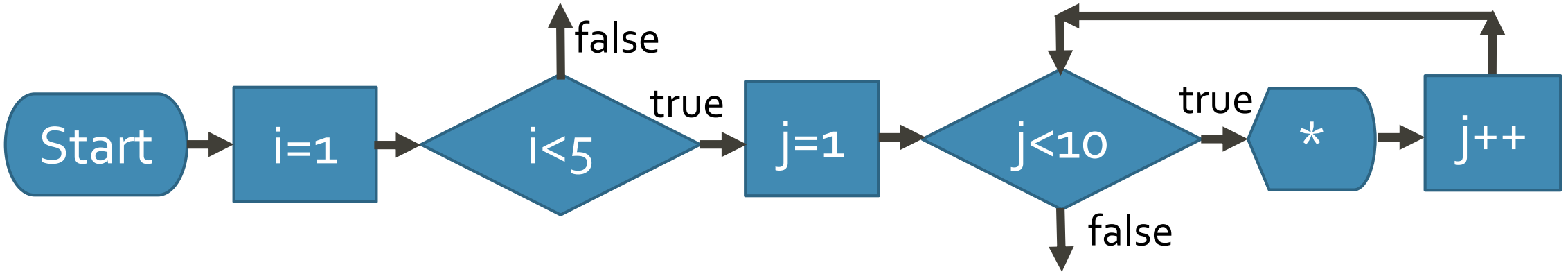
```
for (int i = 1 ; i < 5 ; i++)
{
   for (int j = 1 ; j < 10 ; j++)
   {
      System.out.print ("*");
   }
   System.out.println ("");
}
```

Add the second loop's steps to repeat.



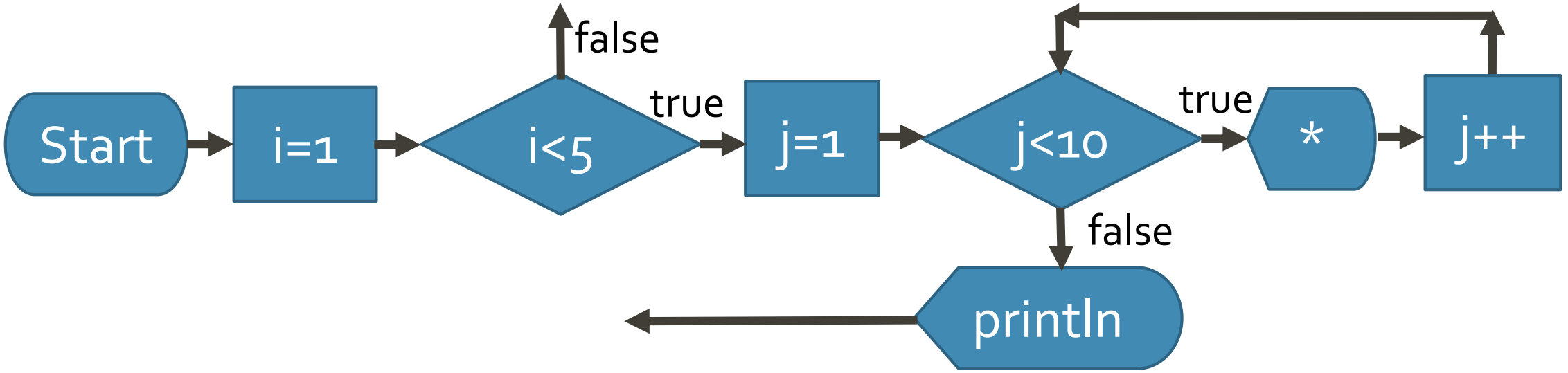
```
for (int i = 1 ; i < 5 ; i++)
{
   for (int j = 1 ; j < 10 ; j++)
   {
      System.out.print ("*");
   }
   System.out.println ("");
}
```

Add the second loop's progress to the stopping condition.



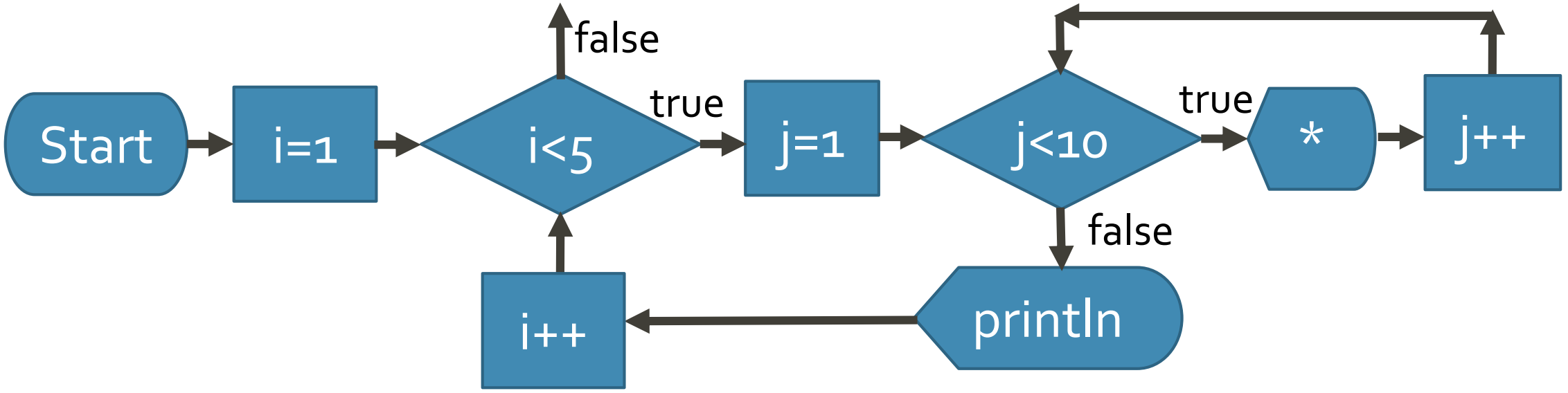
```
for (int i = 1 ; i < 5 ; i++)
{
   for (int j = 1 ; j < 10 ; j++)
   {
      System.out.print ("*");
   }
   System.out.println ("");
}
```

Add the last part of the first loop's steps to repeat



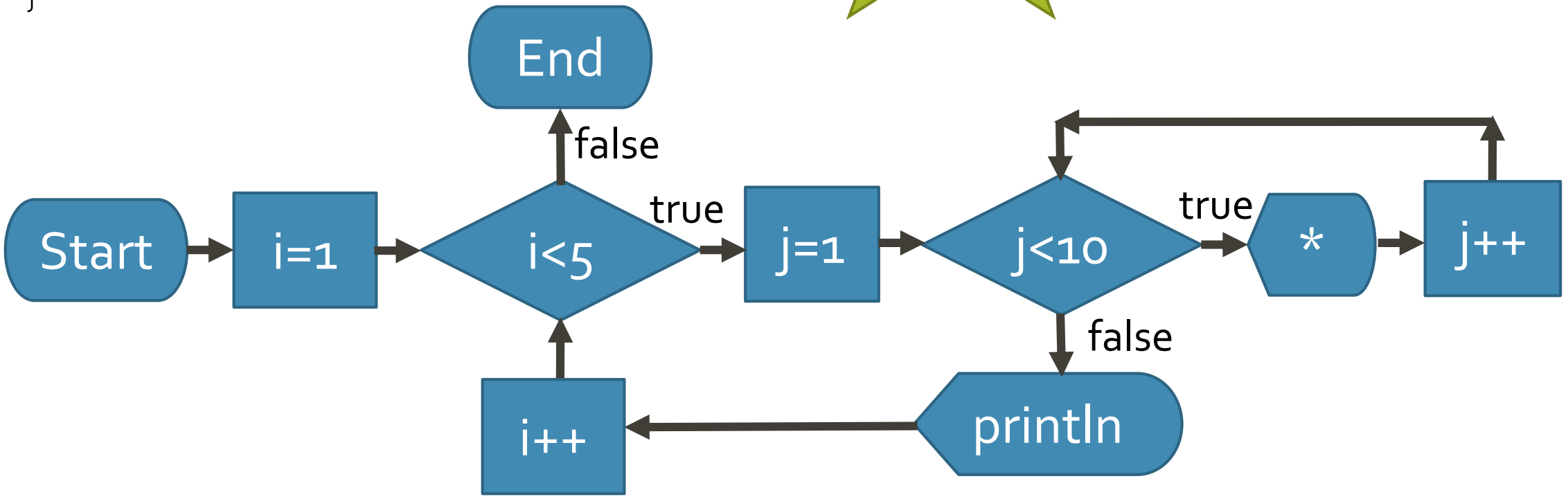
```
for (int i = 1 ; i < 5 ; i++)
{
   for (int j = 1 ; j < 10 ; j++)
   {
      System.out.print ("*");
   }
   System.out.println ("");
}
```

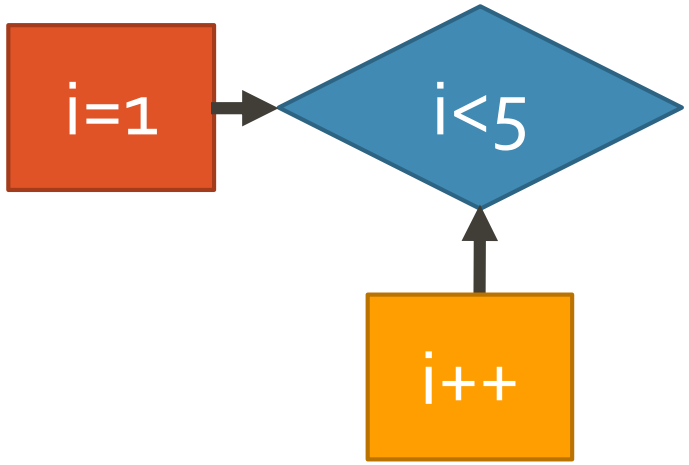
Add the first loop's progress to the loop stopping condition.



```
for (int i = 1 ; i < 5 ; i++)
{
  for (int j = 1 ; j < 10 ; j++)
  {
     System.out.print ("*");
  }
  System.out.println ("");
}
```

End it.





A note on finding things in a flowchart.

- Locate the stopping condition (diamond)
- If you trace backwards, to outside the loop, you get the initialize loop stopping condition.
- If you trace backwards, inside the loop), you get the progress to the loop stopping condition.

```
for (var i = 0; i < 95; i++) {
  moveForward();
}
turnLeft();
for (var j = 0; j < 23; j++) {
  moveForward();
}
makeHoney();
```

A note on finding things in a flowchart.

- Locate the stopping condition (diamond)
- If you trace backwards, to outside the loop, you get the initialize loop stopping condition.
- If you trace backwards, inside the loop), you get the progress to the loop stopping condition.