A Full Program

```
public class FactorialExample extends ConsoleProgram {

  private static final int MAX_NUM = 4;

  public void run() {
      for(int i = 0; i < MAX_NUM; i++) {
            println(i + "! = " + factorial(i));
      }
  }

  private int factorial(int n) {
      int result = 1;
      for (int i = 1; i <= n; i++) {
            result *= i;
      }
      return result;
  }
}
```

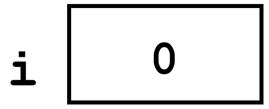
A Full Program

```
public class FactorialExample extends ConsoleProgram {

  private static final int MAX_NUM = 4;

  public void run() {
      for(int i = 0; i < MAX_NUM; i++) {
            println(i + "! = " + factorial(i));
      }
  }

  private int factorial(int n) {
      int result = 1;
      for (int i = 1; i <= n; i++) {
            result *= i;
      }
      return result;
  }
}
```

```
public void run() {
   for(int i = 0; i < MAX_NUM; i++) {
      println(i + "! = " + factorial(i));
   }
}
```

i

```
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        println(i + "! = " + factorial(i));
    }
}
```



```
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        println(i + "! = " + factorial(i));
    }
}
```

```
i   0
```

```
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        println(i + "! = " + factorial(i));
    }
}
```

i 0

```
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        println(i + "! = " + factorial(i));
    }
}
```

i 0

```
private int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

n `0`   result `  `   i `  `

```
private int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

n 0 result 1 i

```
private int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

```
n  0    result  1    i  1
```

```
private int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

n $\boxed{0}$ result $\boxed{1}$ i $\boxed{1}$

```
private int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

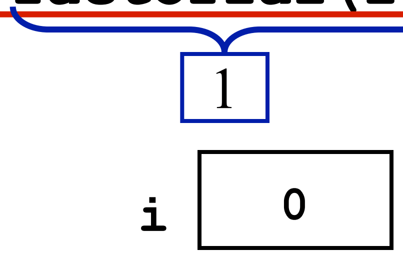
n	result	i
0	1	1

```
public void run() {
   for(int i = 0; i < MAX_NUM; i++) {
      println(i + "! = " + factorial(i));
   }
}
```

$\underbrace{\texttt{factorial(i)}}_{\boxed{1}}$

$\texttt{i}\ \boxed{\;0\;}$

```
public void run() {
   for(int i = 0; i < MAX_NUM; i++) {
      println(i + "! = " + factorial(i));
   }
}
```



```
0! = 1
```

```
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        println(i + "! = " + factorial(i));
    }
}
```

```
i   1
```

```
0! = 1
```

```
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        println(i + "! = " + factorial(i));
    }
}
```

i $\boxed{1}$

```
0! = 1
```

```
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        println(i + "! = " + factorial(i));
    }
}
```

i $\boxed{1}$

```
0! = 1
```

```
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        println(i + "! = " + factorial(i));
    }
}
```

i $\boxed{1}$

```
0! = 1
```

```
private int factorial(int n) {
   int result = 1;
   for (int i = 1; i <= n; i++) {
      result *= i;
   }
   return result;
}
```

```
n  1    result      i
```

```
0! = 1
```

```
private int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

```
          n [ 1 ]    result [ 1 ]    i [   ]
```

```
0! = 1
```

```
private int factorial(int n) {
   int result = 1;
   for (int i = 1; i <= n; i++) {
      result *= i;
   }
   return result;
}
```

```
         n  1       result  1       i  1
```

```
0! = 1
```

```
private int factorial(int n) {
   int result = 1;
   for (int i = 1; i <= n; i++) {
      result *= i;
   }
   return result;
}
```

```
            n   1      result   1      i   1
```

```
0! = 1
```

```
private int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

```
n  1     result  1     i  1
```

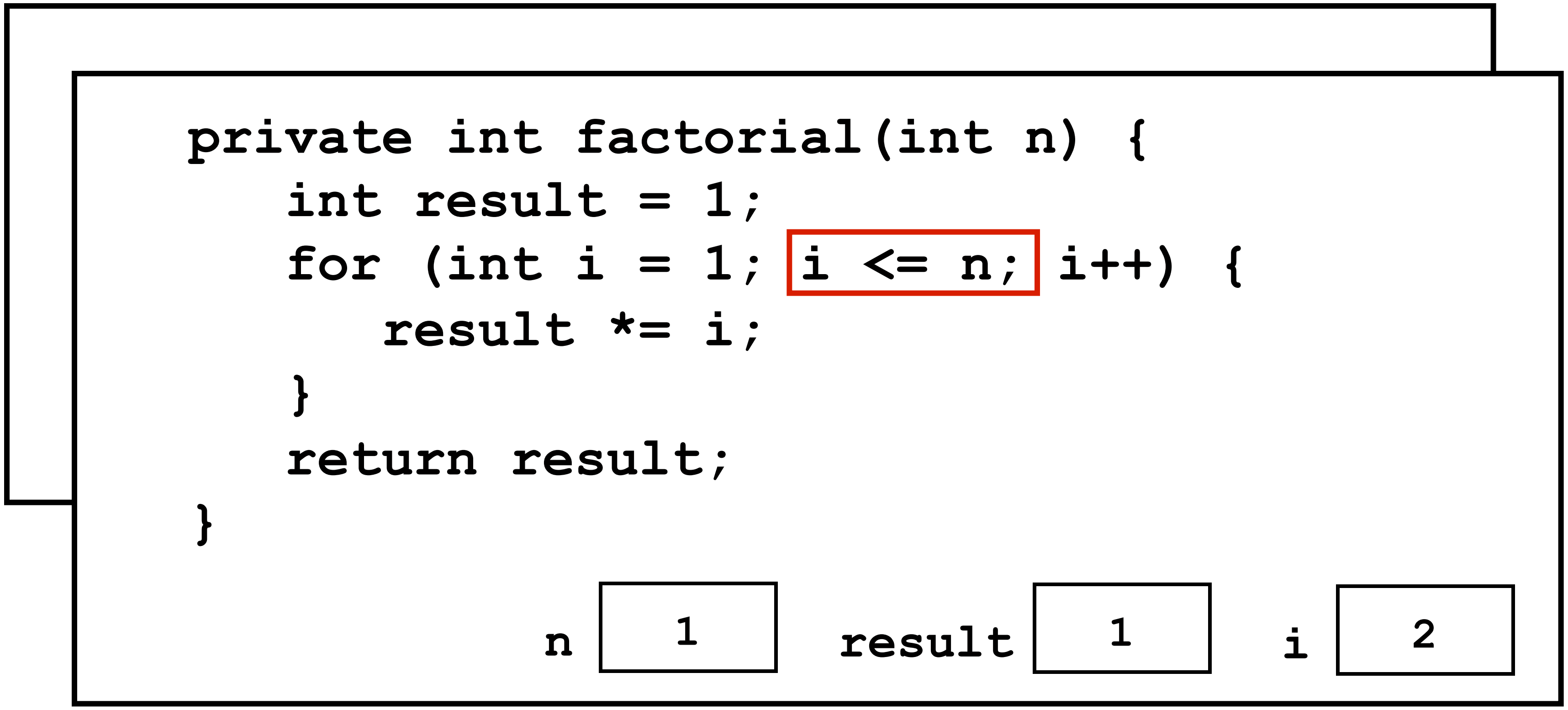
```
0! = 1
```

```
private int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

n `1`   result `1`   i `2`

```
0! = 1
```

```
private int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```



```
0! = 1
```

```
private int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

n $\boxed{1}$ result $\boxed{1}$ i $\boxed{2}$

```
0! = 1
```

```
public void run() {
   for(int i = 0; i < MAX_NUM; i++) {
      println(i + "! = " + factorial(i));
   }
}
```

$\underbrace{\texttt{factorial(i)}}_{\boxed{1}}$

i $\boxed{1}$

```
0! = 1
```

```
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        println(i + "! = " + factorial(i));
    }
}
```

$\underbrace{\texttt{factorial(i)}}_{\boxed{1}}$

i $\boxed{1}$

```
0! = 1
1! = 1
```

```
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        println(i + "! = " + factorial(i));
    }
}
```

i 2

```
0! = 1
1! = 1
```

```
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        println(i + "! = " + factorial(i));
    }
}
```

```
i  2
```

```
0! = 1
1! = 1
```

```
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        println(i + "! = " + factorial(i));
    }
}
```

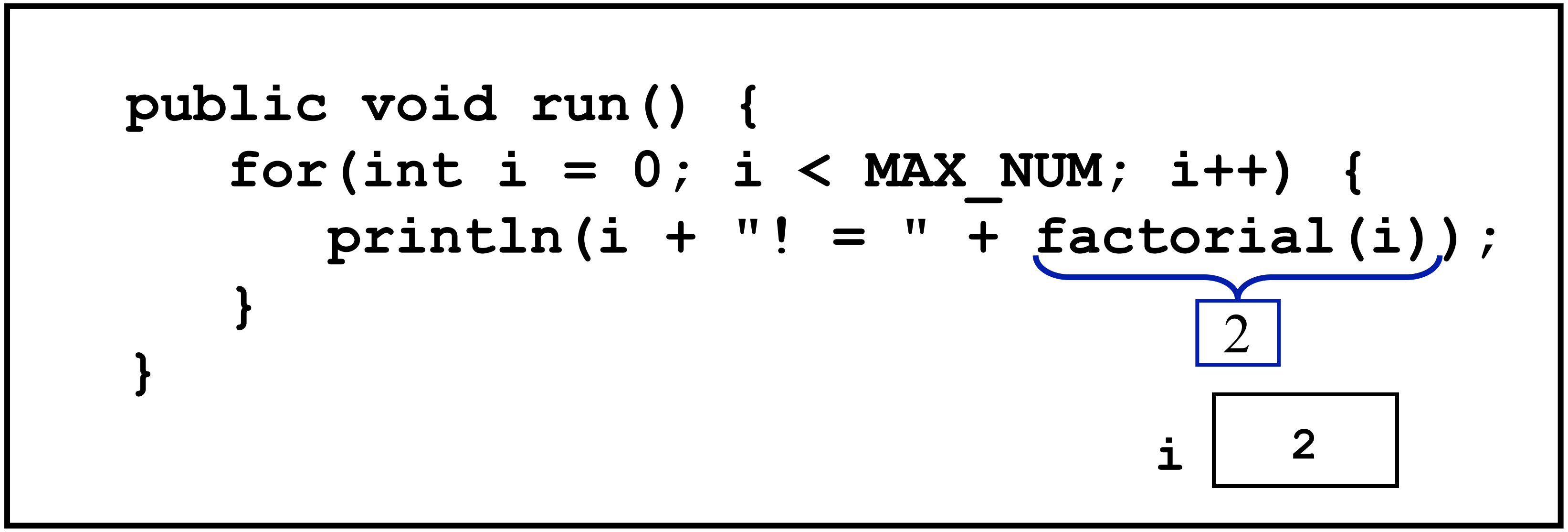
i 2

```
0! = 1
1! = 1
```

```
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        println(i + "! = " + factorial(i));
    }
}
```

i 2

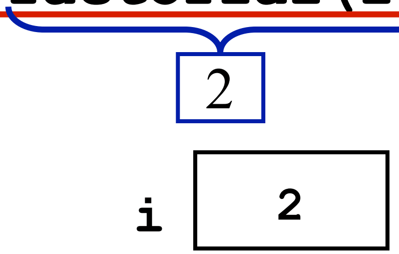
```
0! = 1
1! = 1
```



```
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        println(i + "! = " + factorial(i));
    }
}
```

```
0! = 1
1! = 1
```

```
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        println(i + "! = " + factorial(i));
    }
}
```



```
0! = 1
1! = 1
2! = 2
```

```
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        println(i + "! = " + factorial(i));
    }
}
```

```
i  3
```

```
0! = 1
1! = 1
2! = 2
```

```
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        println(i + "! = " + factorial(i));
    }
}
```

```
i  3
```

```
0! = 1
1! = 1
2! = 2
```

```
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        println(i + "! = " + factorial(i));
    }
}
```

i $\boxed{3}$

```
0! = 1
1! = 1
2! = 2
```

```
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        println(i + "! = " + factorial(i));
    }
}
```

```
i  3
```

```
0! = 1
1! = 1
2! = 2
```

```
public void run() {
   for(int i = 0; i < MAX_NUM; i++) {
      println(i + "! = " + factorial(i));
   }
}
```

$\underbrace{\texttt{factorial(i)}}_{\boxed{6}}$

i $\boxed{3}$

```
0! = 1
1! = 1
2! = 2
```

```
public void run() {
   for(int i = 0; i < MAX_NUM; i++) {
      println(i + "! = " + factorial(i));
   }                                    6
}
                                     i     3
```

```
0! = 1
1! = 1
2! = 2
3! = 6
```

```
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        println(i + "! = " + factorial(i));
    }
}
```

i 4

```
0! = 1
1! = 1
2! = 2
3! = 6
```

```
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        println(i + "! = " + factorial(i));
    }
}
```

```
i   4
```

```
0! = 1
1! = 1
2! = 2
3! = 6
```

Bad Times with Methods

```
// NOTE: This program is buggy!!

private void addFive(int x) {
  x += 5;
}


public void run() {
  int x = 3;
  addFive(x);
  println("x = " + x);
}
```

Good Times with Methods

```
// NOTE: This program is feeling just fine...

private int  addFive(int x) {
   x += 5;
   return x;
}

public void run() {
   int x = 3;
   x = addFive(x);
   println("x = " + x);
}
```

Instance Variables

```
import acm.program.*;

public class MyProgram extends ConsoleProgram {

  public void run() {
      balance = 0;
      for(int i = 0; i < 5; i++) {
         int value = readInt("Value? ");
         addToBalance(value);
      }
  }

  private void addToBalance(int val) {
      balance += val;
  }

  /* Private instance variables */
  private int balance;
}
```

And now a word from …
the Random Number Generator

RandomGenerator

```
import acm.program.*;
import acm.util.*;

public class SimpleRandom extends ConsoleProgram {


  public void run() {
     // Will fill in shortly
  }

  /* Private instance variables */
  private RandomGenerator rgen =
                  RandomGenerator.getInstance();
}
```

Methods to Generate Random Values

The **`RandomGenerator`** class defines the following methods:

`int nextInt(int low, int high)` Returns a random **`int`** between **`low`** and **`high`**, inclusive.
`int nextInt(int n)` Returns a random **`int`** between 0 and **`n`** – 1.
`double nextDouble(double low, double high)` Returns a random **`double`** d in the range **`low`** $\leq d <$ **`high`**.
`double nextDouble()` Returns a random **`double`** d in the range $0 \leq d < 1$.
`boolean nextBoolean()` Returns a random **`boolean`** value, which is **`true`** 50 percent of the time.
`boolean nextBoolean(double p)` Returns a random **`boolean`**, which is **`true`** with probability **`p`**, where $0 \leq$ **`p`** ≤ 1.
`Color nextColor()` Returns a random color.

Simple Random Example

```
import acm.program.*;
import acm.util.*;

public class SimpleRandom extends ConsoleProgram {

  public void run() {
      int dieRoll = rgen.nextInt(1, 6);
      println("You rolled " + dieRoll);
  }

  /* Private instance variables */
  private RandomGenerator rgen =
                  RandomGenerator.getInstance();
}
```