Animation

Announcements

- Assignment 3 (**Problem-Solving in Java**) is due next Monday at 3:15PM.
  - Our recommendation: try to have five of the six problems completed by Friday.
- DiversityBase kick-off event is tonight at 6PM in the Mackenzie Boardroom in the Huang Engineering Center.

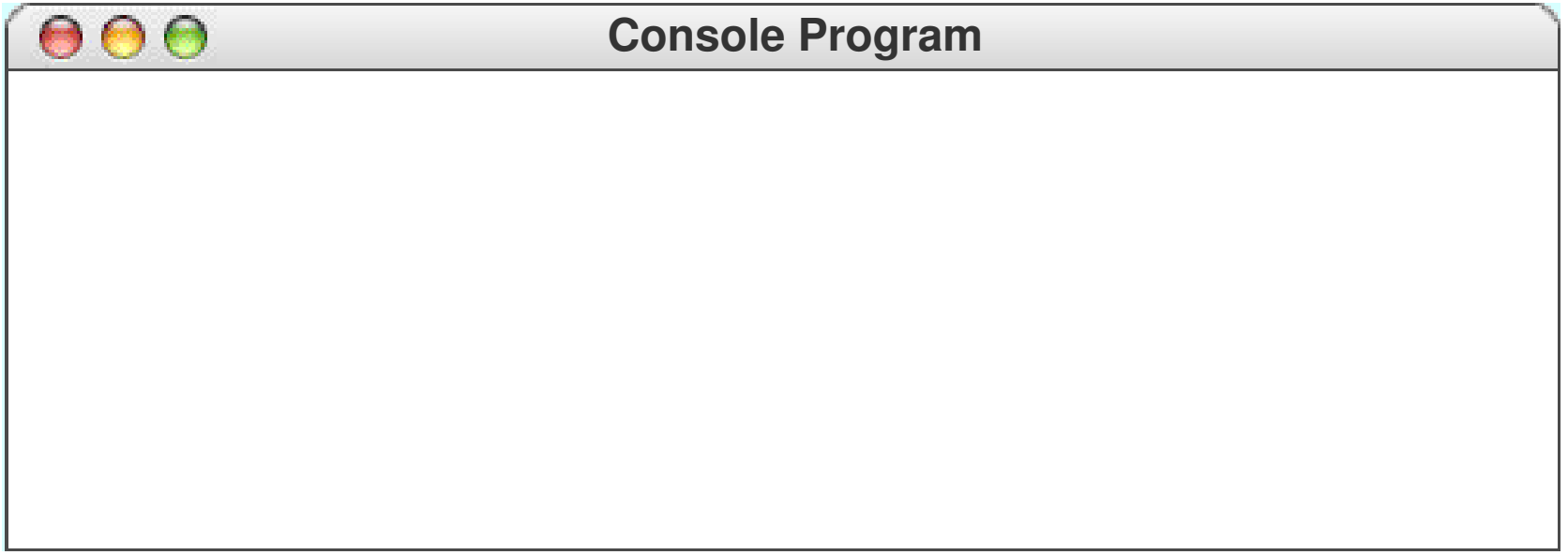
Outline for Today

- **Return Values Revisited**
  - Communicating information out of methods.
- **Animation**
  - How do we move objects around the screen?

Returning Values

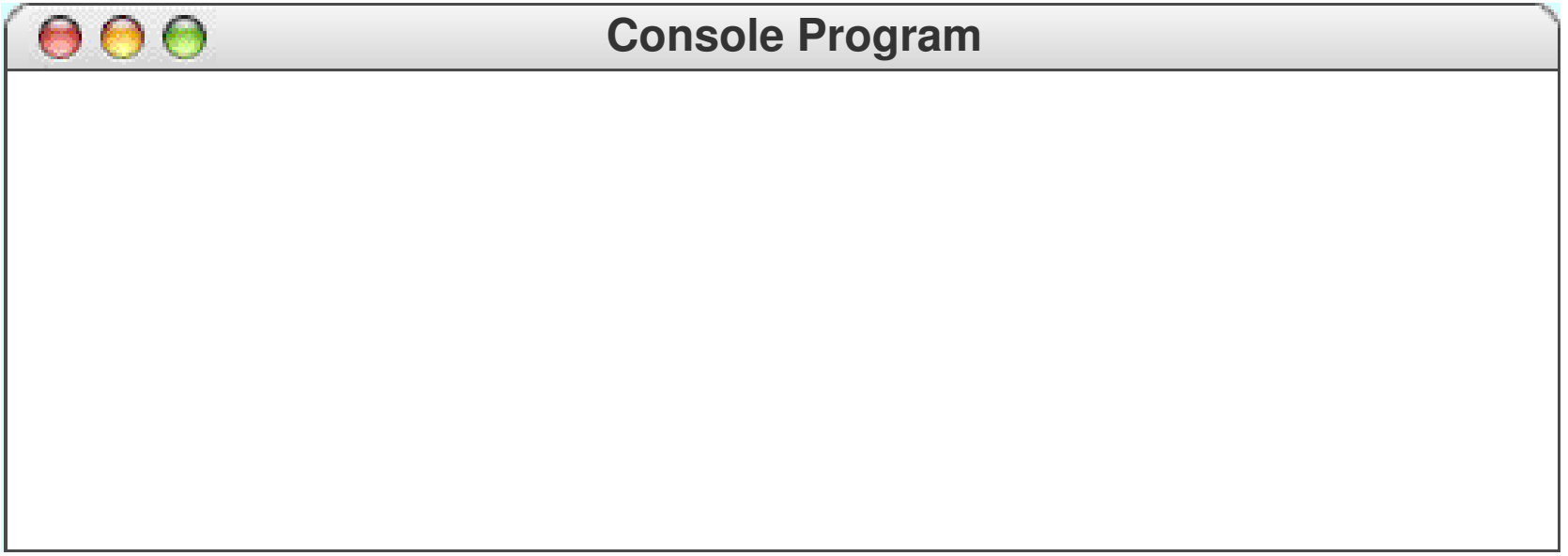
```
public void run() {
   for(int i = 0; i < MAX_NUM; i++) {
      println(i + "! = " + factorial(i));
   }
}
```

i 



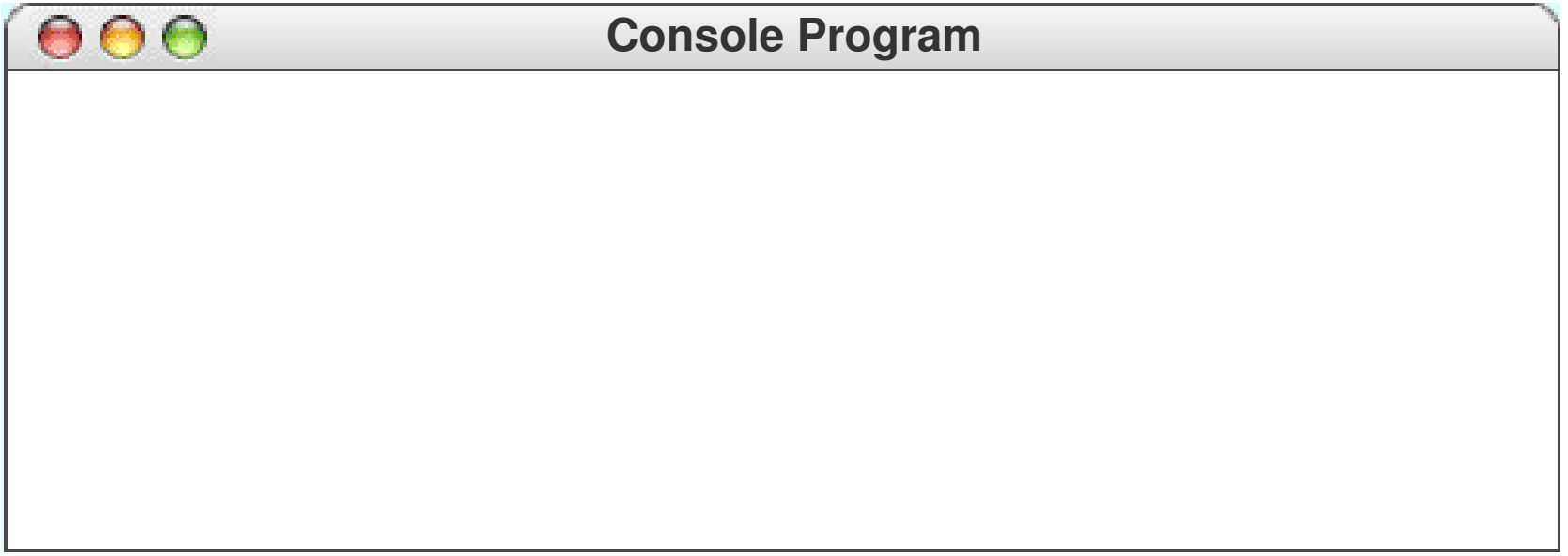
```
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        println(i + "! = " + factorial(i));
    }
}
```

i | 0



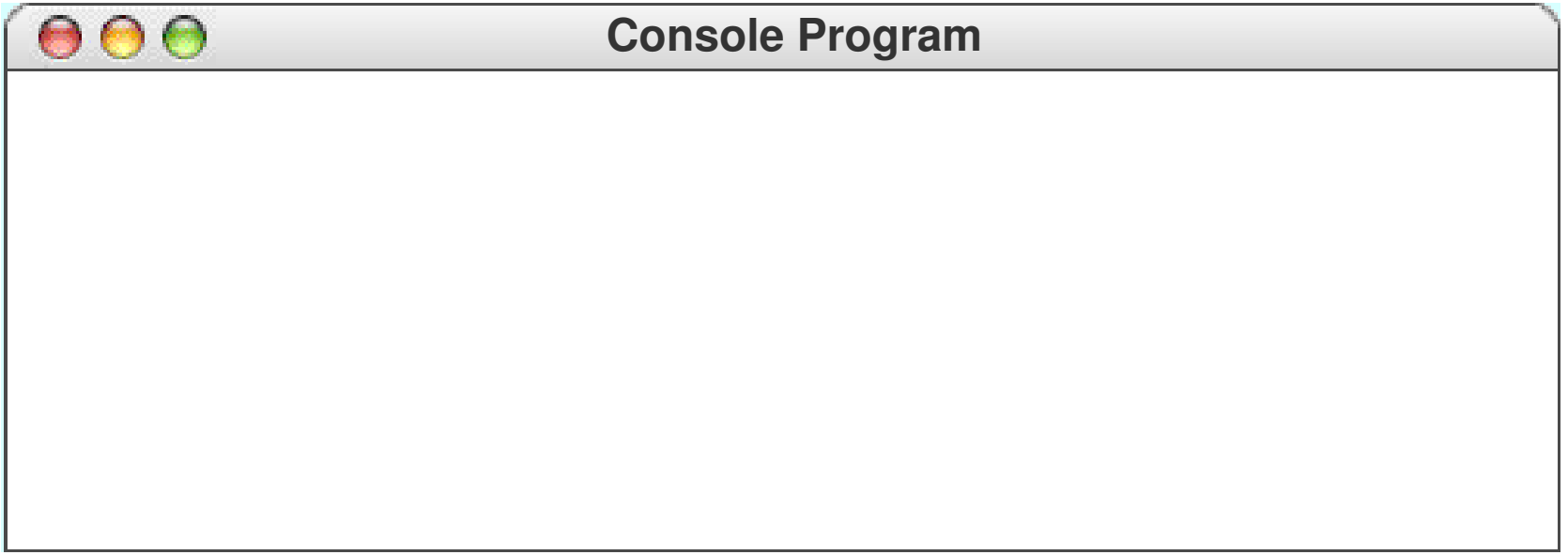
```
public void run() {
   for(int i = 0; i < MAX_NUM; i++) {
      println(i + "! = " + factorial(i));
   }
}
```

i | 0



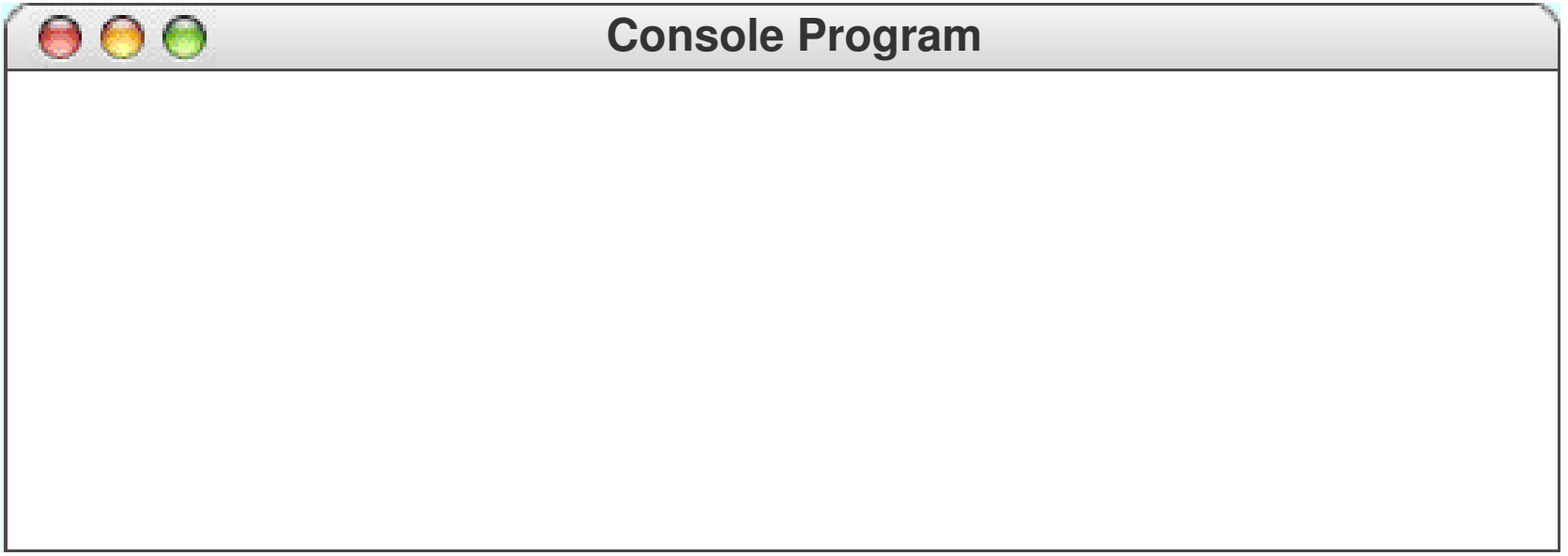
```
public void run() {
   for(int i = 0; i < MAX_NUM; i++) {
      println(i + "! = " + factorial(i));
   }
}
```

i | 0



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

i | 0



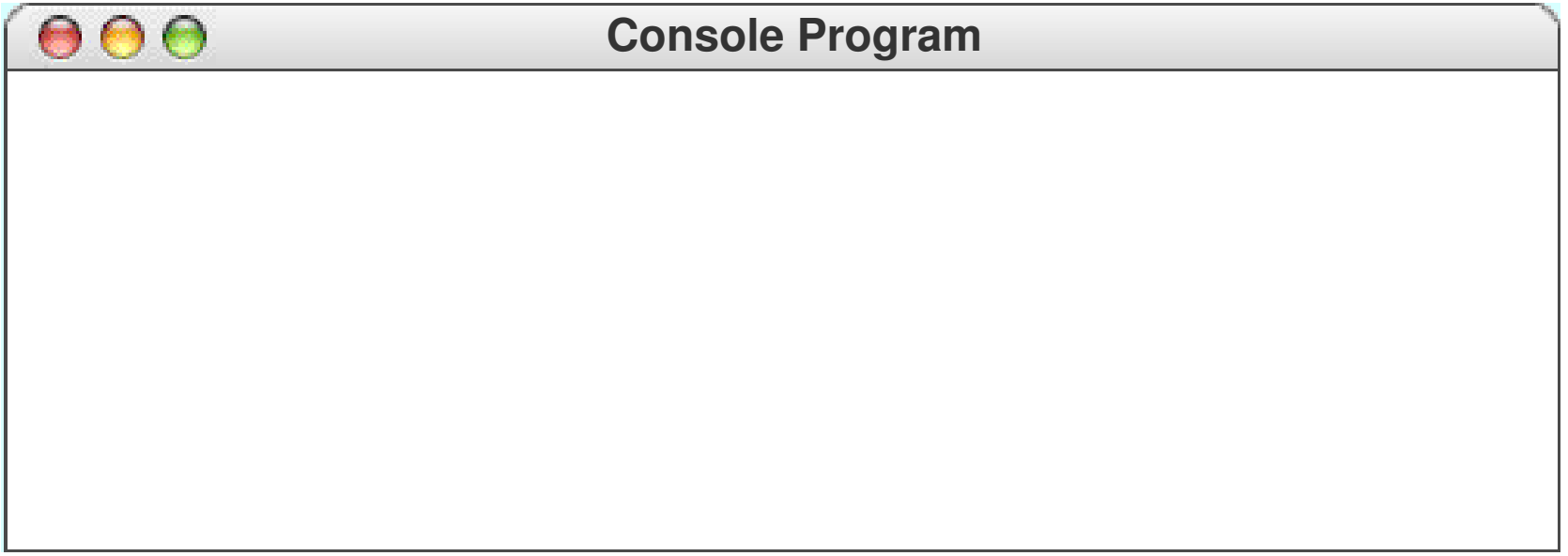
```
private int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

n 0 result i

Console Program

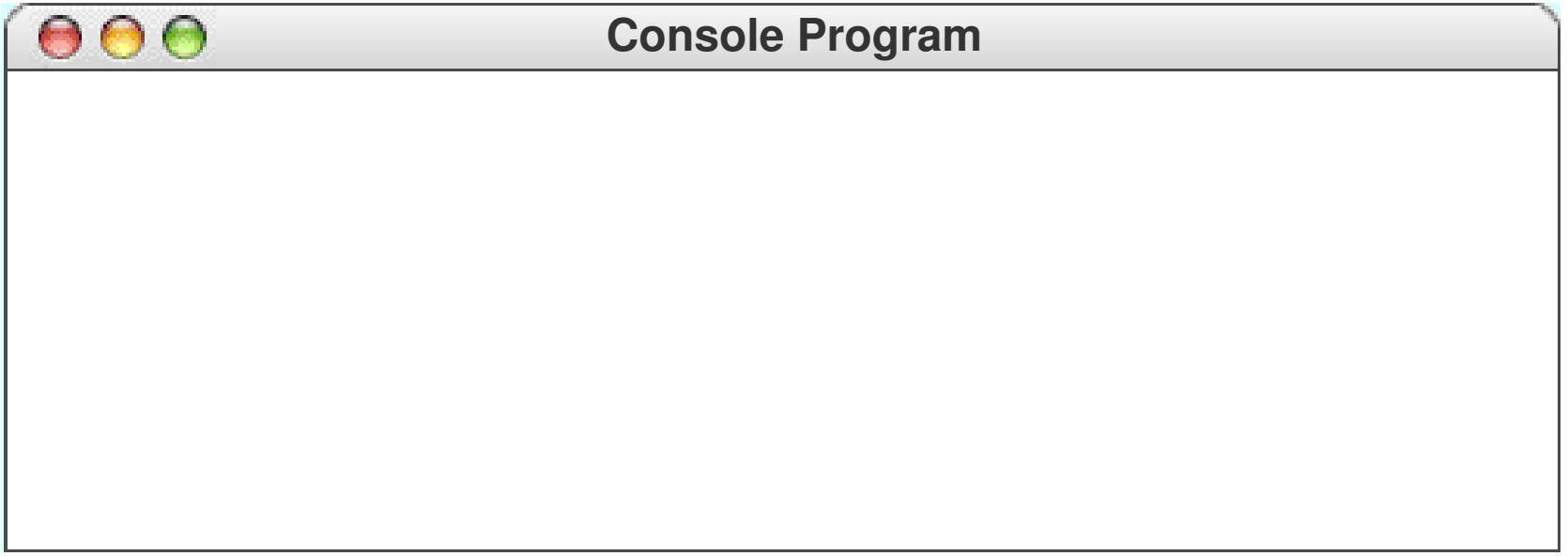
```
private int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

n	0	result	1	i	



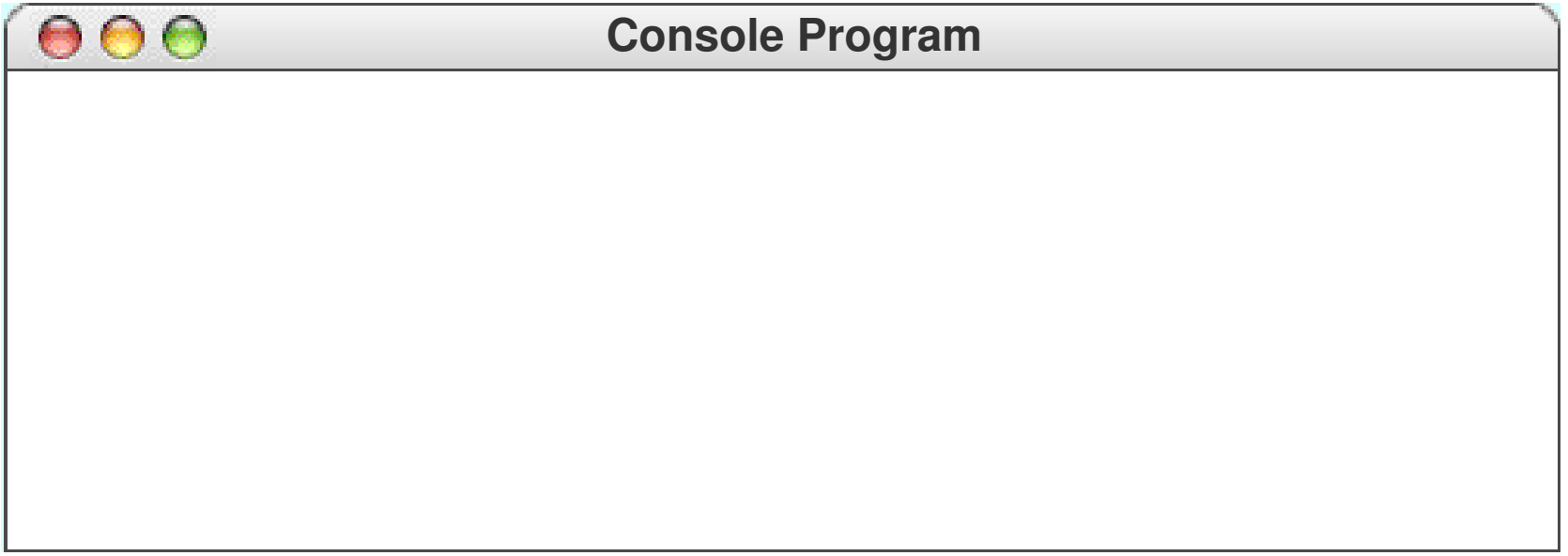
```
private int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

n	0	result	1	i	1



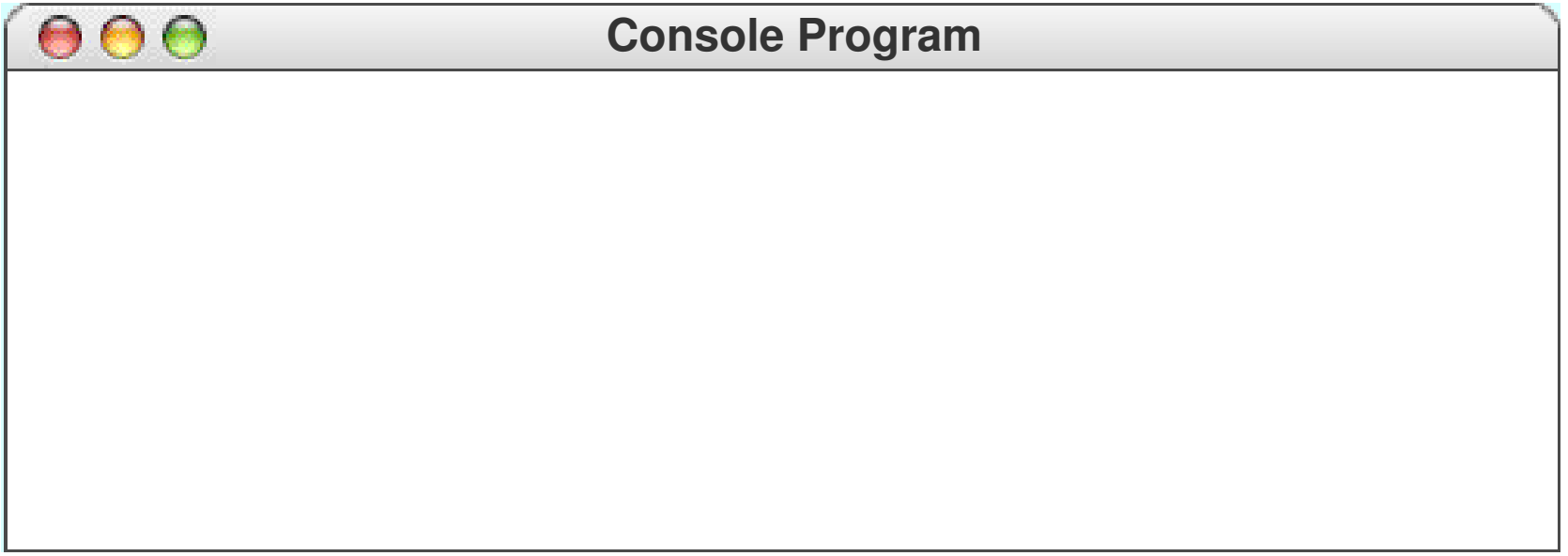
```
private int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

n	0	result	1	i	1

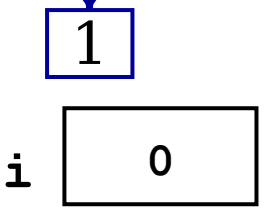


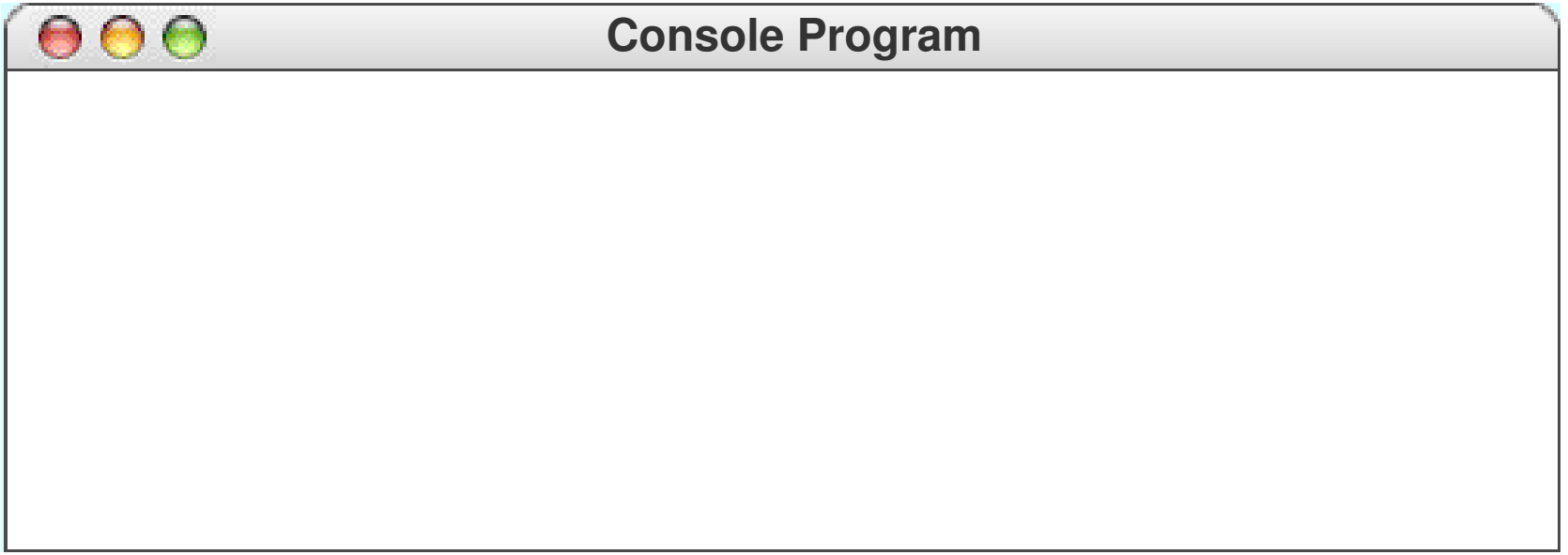
```
private int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

n	0	result	1	i	1

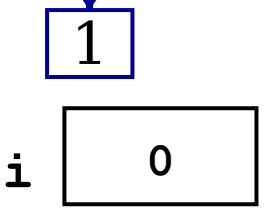


```
public void run() {
   for(int i = 0; i < MAX_NUM; i++) {
      println(i + "! = " + factorial(i));
   }
}
```





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



Console Program

0! = 1

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

i | 1

Console Program

0! = 1

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

i | 1

Console Program

0! = 1

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

i | 1

Console Program

0! = 1

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

i | 1

Console Program

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

Console Program

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	

Console Program

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

Console Program

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

Console Program

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

Console Program

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

Console Program

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

Console Program

0! = 1

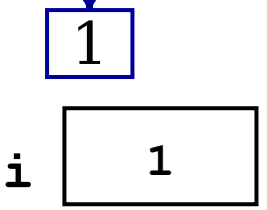
```
private int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

n	1	result	1	i	2

Console Program

0! = 1

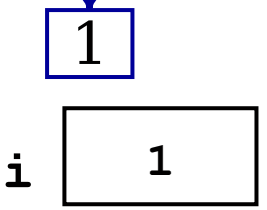
```
public void run() {
   for(int i = 0; i < MAX_NUM; i++) {
      println(i + "! = " + factorial(i));
   }
}
```



Console Program

0! = 1

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



Console Program

0! = 1
1! = 1

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

i | 2

Console Program

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

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

i | 2

Console Program

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

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

i | 2

Console Program

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

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

i | 2

Console Program

0! = 1
1! = 1

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

n 2 result i

Console Program

0! = 1
1! = 1

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

n	2	result	1	i	

Console Program

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

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

n	2	result	1	i	1

Console Program

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

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

n	2	result	1	i	1

Console Program

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

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

n	2	result	1	i	1

Console Program

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

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

n	2	result	1	i	2

Console Program

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

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

n	2	result	1	i	2

Console Program

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

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

n	2	result	2	i	2

Console Program

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

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

n	2	result	2	i	3

Console Program

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

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

n	2	result	2	i	3

Console Program

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

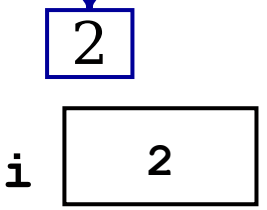
```
private int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

n	2	result	2	i	3

Console Program

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

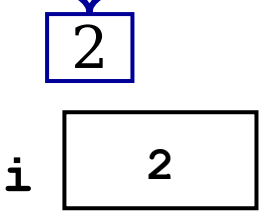
```
public void run() {
   for(int i = 0; i < MAX_NUM; i++) {
      println(i + "! = " + factorial(i));
   }
}
```



Console Program

0! = 1
1! = 1

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



Console Program

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

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

i | 3

Console Program

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

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

i | 3

Console Program

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

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

i | 3

Console Program

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

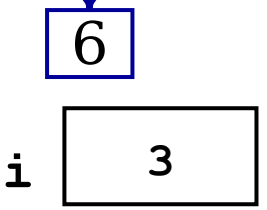
```
public void run() {
   for(int i = 0; i < MAX_NUM; i++) {
      println(i + "! = " + factorial(i));
   }
}
```

i | 3

Console Program

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

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



Console Program

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

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

6

i 3

Console Program

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

Console Program

```
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

Console Program

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

Retiring Young

Pass-by-Value

- Method parameters act as their own variables. They are independent of any similarly-named variable in the calling method.
- This is called ***pass-by-value***.

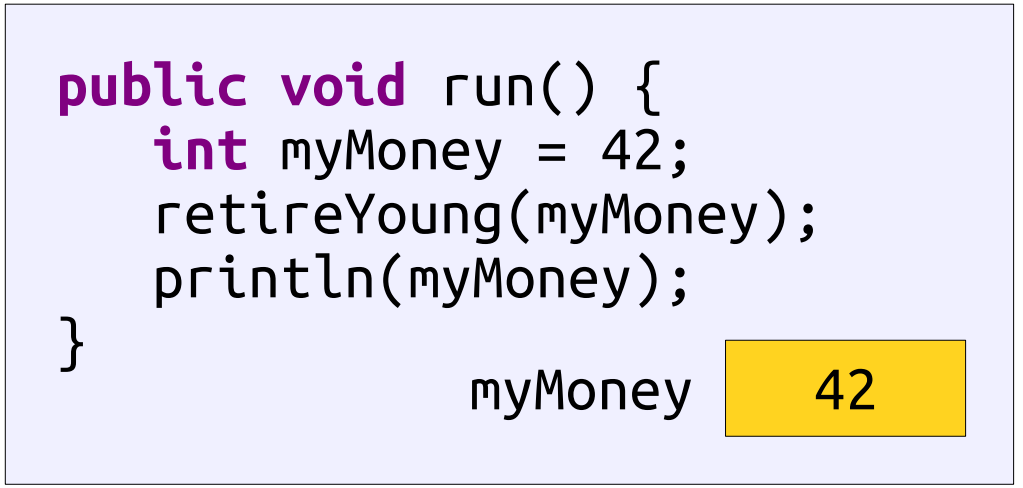
```
private void retireYoung(int myMoney) {
    myMoney = 1000000000;
}
public void run() {
    int myMoney = 42;
    retireYoung(myMoney);
    println(myMoney);
}
```

Pass-by-Value

- Method parameters act as their own variables. They are independent of any similarly-named variable in the calling method.
- This is called ***pass-by-value***.

```
private void retireYoung(int myMoney) {
    myMoney = 1000000000;
}
public void run() {
    int myMoney = 42;
    retireYoung(myMoney);
    println(myMoney);
}
```

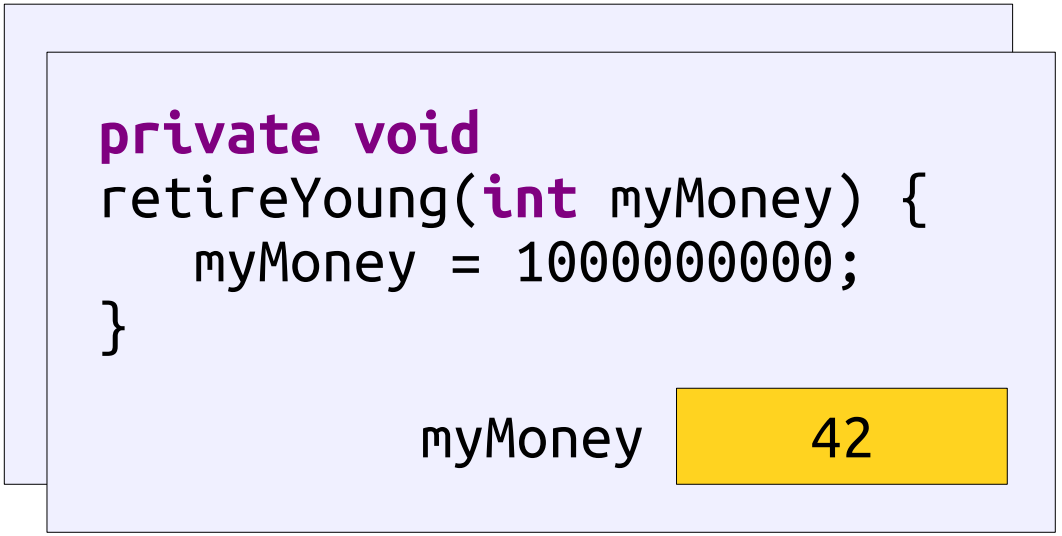
```
public void run() {
    int myMoney = 42;
    retireYoung(myMoney);
    println(myMoney);
}
```



Pass-by-Value

- Method parameters act as their own variables. They are independent of any similarly-named variable in the calling method.
- This is called ***pass-by-value***.

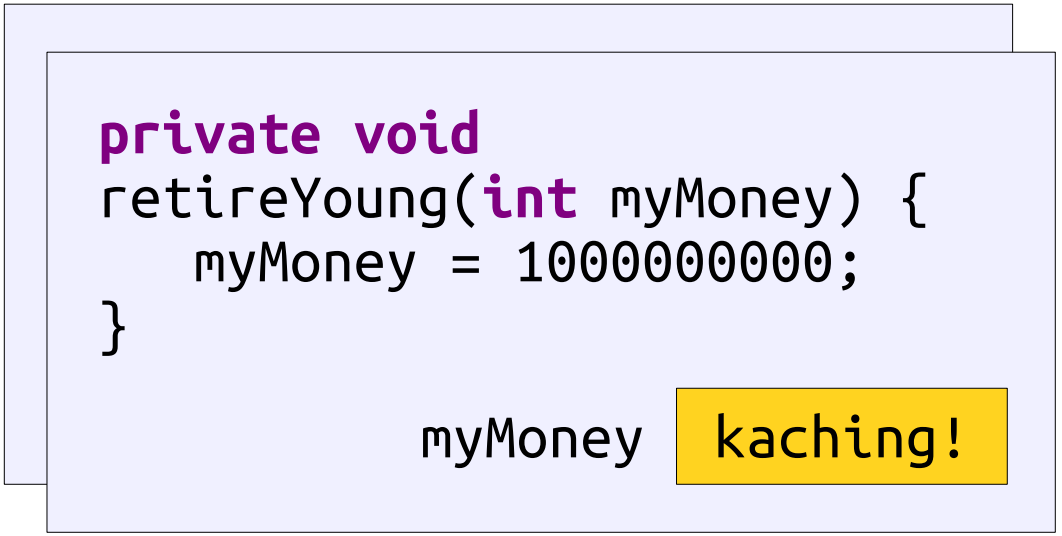
```
private void retireYoung(int myMoney) {
    myMoney = 1000000000;
}
public void run() {
    int myMoney = 42;
    retireYoung(myMoney);
    println(myMoney);
}
```



Pass-by-Value

- Method parameters act as their own variables. They are independent of any similarly-named variable in the calling method.
- This is called ***pass-by-value***.

```
private void retireYoung(int myMoney) {
    myMoney = 1000000000;
}
public void run() {
    int myMoney = 42;
    retireYoung(myMoney);
    println(myMoney);
}
```

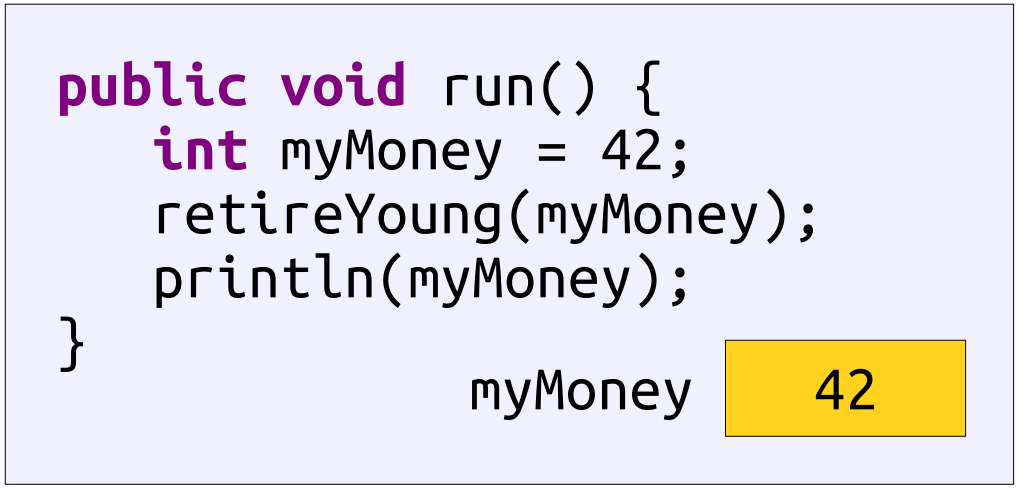


Pass-by-Value

- Method parameters act as their own variables. They are independent of any similarly-named variable in the calling method.
- This is called ***pass-by-value***.

```
private void retireYoung(int myMoney) {
    myMoney = 1000000000;
}
public void run() {
    int myMoney = 42;
    retireYoung(myMoney);
    println(myMoney);
}
```

```
public void run() {
    int myMoney = 42;
    retireYoung(myMoney);
    println(myMoney);
}
```



Many Happy `return`s

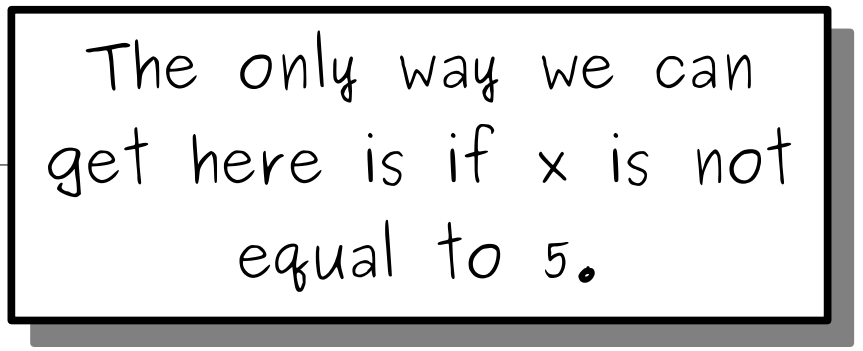
- A method may have multiple return statements. The method ends as soon as `return` is executed.

```
private int thisIsLegal(int x) {
    if (x == 5) {
        return 0;
    } else {
        return 1;
    }
}
```

Many Happy `return`s

- A method may have multiple return statements. The method ends as soon as `return` is executed.

```
private int thisIsLegal(int x) {
    if (x == 5) {
        return 0;
    }
    return 1;
}
```



Prime Numbers

- An integer greater than 1 is called ***prime*** if its only divisors are 1 and itself.
- For example:
  - 5 is prime.
  - 17 is prime.
  - 15 is not prime: it's 3×5
  - 24 is not prime: it's 2×12, 3×8, and 4×6.

A Note about Programming Languages

Java and Python

```java
private void isPrime(int n) {
    if (n <= 1) {
        return false;
    }
    for (int i = 2; i < n; i++) {
        if (n % i == 0) {
            return false;
        }
    }
    return true;
}
```

```python
def isPrime(n):
    if n <= 1:
        return False

    for i in range(2, n):
        if n % i == 0:
            return False

    return True
```

Animation

Operations on the `GObject` Class

The following operations apply to all `GObjects`:

object`.setColor(`*color*`)` Sets the color of the object to the specified color constant.
object`.setLocation(`*x*`, `*y*`)` Changes the location of the object to the point (*x*, *y*).
object`.move(`*dx*`, `*dy*`)` Moves the object on the screen by adding *dx* and *dy* to its current coordinates.

Standard color names defined in the `java.awt` package:

`Color.BLACK`	`Color.RED`	`Color.BLUE`
`Color.DARK_GRAY`	`Color.YELLOW`	`Color.MAGENTA`
`Color.GRAY`	`Color.GREEN`	`Color.ORANGE`
`Color.LIGHT_GRAY`	`Color.CYAN`	`Color.PINK`
`Color.WHITE`		

Graphic courtesy of Eric Roberts

Operations on the `GObject` Class

The following operations apply to all `GObjects`:

object`.setColor(`*color*`)`
Sets the color of the object to the specified color constant.

object`.setLocation(`*x*`, `*y*`)`
Changes the location of the object to the point (*x*, *y*).

object`.move(`*dx*`, `*dy*`)`
Moves the object on the screen by adding *dx* and *dy* to its current coordinates.

Standard color names defined in the `java.awt` package:

`Color.BLACK`	`Color.RED`	`Color.BLUE`
`Color.DARK_GRAY`	`Color.YELLOW`	`Color.MAGENTA`
`Color.GRAY`	`Color.GREEN`	`Color.ORANGE`
`Color.LIGHT_GRAY`	`Color.CYAN`	`Color.PINK`
`Color.WHITE`		

Graphic courtesy of Eric Roberts

Animation

- By repositioning objects after they have been added to the canvas, we can create animations.
- General pattern for animation:

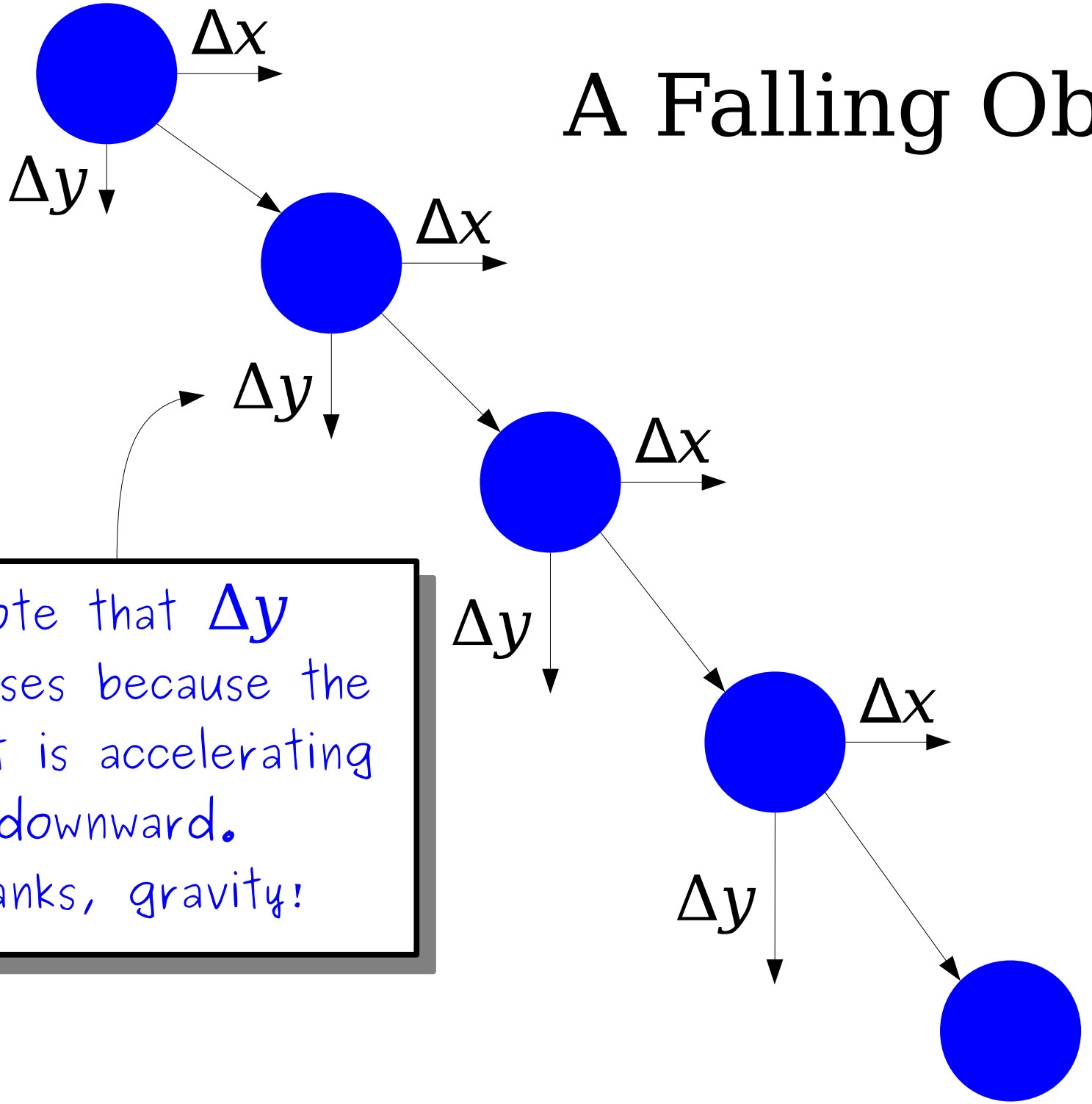
```
while (animation-not-finished) {
    update graphics;
    pause(pause-time);
}
```

Physics Simulation

http://physbam.stanford.edu/~fedkiw/animations/glass00.avi

http://physbam.stanford.edu/~fedkiw/animations/motion_s

A Falling Object



Let's Code It Up!

A Sticky Situation