Animation
An Interesting Website

www.boxcar2d.com
Returning Values

• A method may produce a value that can be read by its caller.

• To indicate that a method returns a value, specify the type returned in the method declaration:

```java
private type name(parameters) {
    /* ... method body ... */
}
```

• A value can be returned with the `return` statement:

```java
return value;
```
Subtleties of return

• If a method has non-\texttt{void} return type, it must always return a value.

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

What do we return if \( x \neq 5 \)?
Subtleties of \texttt{return} \par

- If a method has non-\texttt{void} return type, it must always return a value. 

```java
private int thisIsLegal(int x) {
    if (x == 5) {
        return 0;
    }
    else {
        return 1;
    }
}
```
Many Happy \texttt{returns}

- A method may have multiple return statements. The method ends as soon as \texttt{return} is executed.

```java
private int thisIsLegal(int x) {
    if (x == 5) {
        return 0;
    } else {
        return 1;
    }
}
```
Many Happy \textit{returns}

- A method may have multiple return statements. The method ends as soon as \texttt{return} is executed.

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

The only way we can get here is if \( x \) is not equal to 5.
Scope

- Each variable has a **scope** where it can be accessed and how long it lives.

```java
for (int i = 0; i < 5; i++) {
    int y = i * 4;
}

i = 3; // Error!
y = 2; // Error!
```
Scope of Method Calls

• A variable declared inside a method is called a local variable.

• Local variables can only be accessed inside of the method that declares them.

```java
public void run() {
    int x = 5;
    someOtherMethod();
}

private void someOtherMethod() {
    x = 4; // Error!
}
```
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));
    }
}

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

Slides by Mehran Sahami
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));
    }
}
```java
private int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```
```java
private int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```
```java
private int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```
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
```java
private int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

```
0 1
```

```
Console Program
```

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

Slides by Mehran Sahami
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));
    }
}

0! = 1
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
```java
private int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

```java
0! = 1
```
```java
private int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

```
0! = 1
```
```java
private int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

```
0! = 1
```

Slides by Mehran Sahami
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
```java
private int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

```java
0! = 1
```
```java
private int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

Slides by Mehran Sahami

**Console Program**

```
0! = 1
```
```java
private int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

Console Program

0! = 1
```java
private int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}
```

```
0! = 1
```
```java
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        println(i + "! = " + factorial(i));
    }
}
```

0! = 1

---

Console Program

0! = 1
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        println(i + "! = " + factorial(i));
    }
}

int factorial(int n) {
    if (n == 0) {
        return 1;
    } else {
        return n * factorial(n - 1);
    }
}

Slides by Mehran Sahami

Console Program

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

1! = 1
0! = 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));
    }
}

0! = 1
1! = 1
```java
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        println(i + "! = " + factorial(i));
    }
}
```

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

3

Slides by Mehran Sahami

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

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

```
slides by mehran sahami
```

```
0! = 1
1! = 1
2! = 2
```
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        println(i + "! = " + factorial(i));
    }
}

Slides by Mehran Sahami
```java
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        System.out.println(i + "! = " + factorial(i));
    }
}
```

```
Slides by Mehran Sahami
```

```
3! = 6
```

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

```
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));
    }
}
```java
public void run() {
    for(int i = 0; i < MAX_NUM; i++) {
        println(i + "! = " + factorial(i));
    }
}
```
Retiring Young
Pass-by-Value

- Java methods pass their parameters by **value**.
- The method gets a *copy* of its parameters, not the actual parameters themselves.

```java
private void myMethod(int x) {
    x = 137;
}
```

```java
public void run() {
    int x = 42;
    myMethod(x);
    println("The value of x is " + x);
}
```

This statement prints 42, not 137.
Time-Out For Announcements!
Assignment 2

• Assignment 2 is due on Friday.

• **Recommendation:** Complete all parts of the assignment by Wednesday. Start testing the first five parts of the assignment.

• LaIR open through Thursday, 6PM – Midnight.
Back to CS106A!
Animation
Operations on the GObject Class

The following operations apply to all GObject:

- **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.DARK_GRAY
- Color.GRAY
- Color.LIGHT_GRAY
- Color.WHITE
- Color.RED
- Color.YELLOW
- Color.GREEN
- Color.CYAN
- Color.BLUE
- Color.MAGENTA
- Color.ORANGE
- Color.PINK

Graphic courtesy of Eric Roberts
Operations on the **GObject** Class

The following operations apply to all **GObjects**:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>object.setColor(color)</code></td>
<td>Sets the color of the object to the specified color constant.</td>
</tr>
<tr>
<td><code>object.setLocation(x, y)</code></td>
<td>Changes the location of the object to the point ((x, y)).</td>
</tr>
<tr>
<td><code>object.move(dx, dy)</code></td>
<td>Moves the object on the screen by adding (dx) and (dy) to its current coordinates.</td>
</tr>
</tbody>
</table>

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

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

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:

```plaintext
while (not-done-condition) {
    update graphics
    pause(pause-time);
}
```
Physics Simulation
Note that $\Delta y$ increases because the object is accelerating downward.
Let's Code It Up!
A Sticky Situation

The ball is below the ground, so we reverse its $\Delta y$.

It's still below the ground, so we reverse its $\Delta y$ again.