Methods
Drawing Geometrical Objects
Drawing Geometrical Objects

Constructors

```java
new GRect(x, y, width, height)
```

Creates a rectangle whose upper left corner is at \((x, y)\) of the specified size.
Constructors

**new GRect(x, y, width, height)**
Creates a rectangle whose upper left corner is at \((x, y)\) of the specified size.

**new GOval(x, y, width, height)**
Creates an oval that fits inside the rectangle with the same dimensions.
# Drawing Geometrical Objects

## Constructors

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>new GRect(x, y, width, height)</code></td>
<td>Creates a rectangle whose upper left corner is at ((x, y)) of the specified size.</td>
</tr>
<tr>
<td><code>new GOval(x, y, width, height)</code></td>
<td>Creates an oval that fits inside the rectangle with the same dimensions.</td>
</tr>
<tr>
<td><code>new GLine(x₀, y₀, x₁, y₁)</code></td>
<td>Creates a line extending from ((x₀, y₀)) to ((x₁, y₁)).</td>
</tr>
</tbody>
</table>

[Graphic courtesy of Eric Roberts](#)
Each point $k$ is connected to point $k + 2$, after wrapping around.
Each point $k$ is connected to point $k + 2$, after wrapping around.

Point $k$ is at $\frac{k}{\text{numSides}} \times 360^\circ$
Each point $k$ is connected to point $k + 2$, after wrapping around.

Point $k$ is at $\frac{k}{\text{numSides}} \times 360^\circ$

$1 \ (51.43^\circ)\\
(x + r \cos \theta, \ y + r \sin \theta)$
Each point $k$ is connected to point $k + 2$, after wrapping around.

Point $k$ is at $\frac{k}{\text{numSides}} \times 360^\circ$
Passing Parameters

- A method can accept **parameters** when it is called.

- Syntax:

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

- The values of the parameters inside the method are set when the method is called.

- The values of the parameters can vary between calls.
For more on the geometry and properties of stars:

**Vi Hart on Stars:**
http://youtu.be/CfJzrmS9UfY

**Wikipedia on Stars:**
Time-Out For Announcements!
Friday Four Square!
Today at 4:15PM, Outside Gates
Assignment 2

• Assignment 2 is due a week from today.
• **Recommendation:** Have the first five problems completed by Monday.
• Have questions? Stop by the LaIR, Keith's/Vikas's office hours, or email your section leader!
Back to CS106A!
Factorials

- The number *n* factorial, denoted *n*!, is
  \[1 \times 2 \times 3 \times \ldots \times (n - 1) \times n\]

- For example:
  - 3! = 1 × 2 × 3 = 6.
  - 5! = 1 × 2 × 3 × 4 × 5 = 120
  - 0! = 1 (by definition)

- Factorials show up everywhere:
  - Taylor series.
  - Counting ways to shuffle a deck of cards.
  - Determining how quickly computers can sort values.
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-`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 != 5`?
Subtleties of `return`

- If a method has non-`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 returns

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

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

• A method may have multiple return statements. The method ends as soon as `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.