Expressions, Statements, and Control Structures
Announcements

• Assignment 2 out, due next Wednesday, February 1.
  • Explore the Java concepts we've covered and will be covering.
  • Unleash your creative potential!
YEAH Hours

• Your Early Assignment Help Hours.
• Review session going over major points of the assignment.
• Tonight at 7:00PM in Braun Auditorium.
• Should be available on SCPD tomorrow.
Highlights from Emails
CS is not lame,
Too many essays are lame,
Prove I'm not just fuzz.

I play Temple Run,
And like to watch the sky and,
Waste time with haikus.
Sending Messages

• To call a method on an object stored in a variable, use the syntax

  `object . method(parameters)`

• For example:

  `label.setFont("Comic Sans-32");`
  `label.setColor(Color.ORANGE);`
Operations on the **GObject** Class

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

<table>
<thead>
<tr>
<th><strong>operation</strong></th>
<th><strong>description</strong></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
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
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.

```java
new GOval(x, y, width, height)
```
Creates an oval that fits inside the rectangle with the same dimensions.

Graphic courtesy of Eric Roberts
## 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(x_0, y_0, x_1, y_1)</code></td>
<td>Creates a line extending from ((x_0, y_0)) to ((x_1, y_1)).</td>
</tr>
</tbody>
</table>

![Graphics Program Diagram](image)

*Graphic courtesy of Eric Roberts*
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>

Methods shared by the GRect and GOval classes

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>object.setFilled(fill)</code></td>
<td>If fill is <code>true</code>, fills in the interior of the object; if <code>false</code>, shows only the outline.</td>
</tr>
<tr>
<td><code>object.setFillColor(color)</code></td>
<td>Sets the color used to fill the interior, which can be different from the border.</td>
</tr>
</tbody>
</table>
The Collage Model
The Collage Model
Constants

- Not all variables actually *vary*.
- A **constant** is a name for a value that never changes.
- Syntax (defined outside of any method):
  
  ```java
  private static final type name = value;
  ```
- By convention, constants are named in `UPPER_CASE_WITH_UNDERSCORES` to differentiate them from variables.
Magic Numbers

• A **magic number** is a number written in a piece of code whose meaning cannot easily be deduced from context.

  ```java
  double weight = 9.8 * (m - 14);
  ```

• Constants make it easier to read code:

  ```java
  double weight = GRAVITY * (m - TARE_MASS);
  ```

• Avoid magic numbers in your code by using constants.
Expressions
class Add2Integers extends ConsoleProgram {
    public void run() {
        println("This program adds two numbers.");
        int n1 = readInt("Enter n1: ");
        int n2 = readInt("Enter n2: ");
        int total = n1 + n2;
        println("The total is " + total + ".");
    }
}
class Add2Integers extends ConsoleProgram {
    public void run() {
        println("This program adds two numbers.");
        int n1 = readInt("Enter n1: ");
        int n2 = readInt("Enter n2: ");
        int total = n1 + n2;
        println("The total is " + total + ".");
    }
}
Expressions

• Variables and other values can be used in **expressions**.

• Some familiar mathematical operators:
  • + (addition)
  • − (subtraction)
  • * (multiplication)
  • / (division)
Fun with Division
Size of the Graphics Window

Methods provided by **GraphicsProgram** class

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getWidth()</td>
<td>Returns the width of the graphics window.</td>
</tr>
<tr>
<td>getHeight()</td>
<td>Returns the height of the graphics window.</td>
</tr>
</tbody>
</table>

Note: receiver of these calls is the **GraphicsProgram** itself, so we don’t need to specify a separate object as receiver.
Centering an Object

\[ x = \frac{\text{getWidth}()}{2.0} \] 
\[ x = \frac{\text{getWidth}() - W}{2.0} \]
The Remainder Operator

- The special operator % computes the **remainder** of one value divided by another.

- For example:
  - $15 \div 3 = 0$
  - $14 \div 8 = 6$
  - $21 \div 2 = 1$
  - $14 \div 17 = 14$
Operator Precedence

- Java's mathematical operators have the following precedence:
  - ()  (highest)
  - * / %
  - + -  (lowest)
- Operators of equal precedence are evaluated left-to-right.
A Useful Shorthand


- Commonly, programs contain code like this:

  \[
  \begin{align*}
  x &= x + 1; \\
  y &= y \times 137; \\
  z &= z / 14; \\
  w &= w - 3;
  \end{align*}
  \]
A Useful Shorthand

- Commonly, programs contain code like this:

  \[
  \begin{align*}
  x &= x + 1; \\
  y &= y * 137; \\
  z &= z / 14; \\
  w &= w - 3;
  \end{align*}
  \]

- The statement

  \[
  \text{variable} = \text{variable op value};
  \]

  can be rewritten as

  \[
  \text{variable op= value;}
  \]
A Useful Shorthand

- Commonly, programs contain code like this:
  \[
  x += 1; \quad y *= 137; \\
  z /= 14; \quad w -= 3;
  \]

- The statement
  \[
  \text{variable} = \text{variable op value} ;
  \]
  can be rewritten as
  \[
  \text{variable op= value} ;
  \]
Another Useful Shorthand

- In the special case of writing
  \[ \text{variable} = \text{variable} + 1; \]
  we can instead write
  \[ \text{variable} ++; \]
- In the special case of writing
  \[ \text{variable} = \text{variable} - 1; \]
  we can instead write
  \[ \text{variable} --; \]
Boolean Expressions

• A **boolean expression** is a test for a condition (it is either **true** or **false**).

• Value comparisons:

  ```
  ==  “equals” (note: not single =)
  !=  “not equals” (cannot say <>)
  >   “greater than”
  <   “less than”
  >=  “greater than or equal to”
  <=  “less than or equal to”
  ```
Logical Operators

- We can apply \textbf{logical operators} to boolean values to produce new values.

- Logical \textbf{NOT}: !p
  - !p is \texttt{true} if p is \texttt{false}; !p is \texttt{false} if p is \texttt{true}.

- Logical \textbf{AND}: p && q
  - p && q is \texttt{true} when both p and q are true.

- Logical \textbf{OR}: p || q
  - p || q is \texttt{true} when p is true, q is true, or both p and q are true.

- Order of precedence given above.
Short-Circuit Evaluation

- Cute observations:
  - `true || p` is always `true`.
  - `false && p` is always `false`.

- The logical operators **short-circuit**: if the answer is known from the left operand, the right side is not computed.

- Example: The code
  
  ```java
  boolean b = (x == 0) || ((y / x) < 20)
  ```
  
  will never divide by zero.

Based on slides by Mehran Sahami
Control Statements Revisited
Control Structures in Karel

for
if
while
Control Structures in Karel

for

if

while
This is called the initialization statement and is performed before the loop starts.

```
for (int i = 0; i < 3; i++) {
    ...
}
```

This is called the step or increment and is performed at the end of each loop iteration.

This is called the loop condition or termination condition. The loop will check whether this statement is true before each execution.
Nyan nyan nyan nyan, nyan nyan nyan nyan nyan nyan, nyan, nyan, nyan nyan nyan nyan ...
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}

int i  0
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}

int i 0
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}

int i 0
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}

int i  0

Nyan!
for (int i = 0; i < 4; i++) {
    println("Nyan! ");
}

int i 0

Nyan!
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}

int i  1

Nyan!
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}

int i  1

Nyan!
for (int i = 0; i < 4; i++) {
    println("Nyan! ");
}

int i 1
for (int i = 0; i < 4; i++) {
    println("Nyan! ");
}

int i 1

Console Program
Nyan!
Nyan!
for (int i = 0; i < 4; i++) {
    println("Nyan! ");
}

int i 1

Nyan!
Nyan!
for (int i = 0; i < 4; i++) {
    println("Nyan! ");
}

int i 2
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}

int i 2

Nyan!
Nyan!
```java
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}
```

```
int i 2
```

```
Nyan!
Nyan!
```

Console Program

```
Nyan!
Nyan!
```
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}

int i = 2

Console Program

Nyan!
Nyan!
Nyan!
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}

int i 2

Nyan!
Nyan!
Nyan!
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}

int i 3

Nyan!
Nyan!
Nyan!
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}

int i 3

Console Program
Nyan!
Nyan!
Nyan!
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}

int i 3
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}

int i 3

Nyan!
Nyan!
Nyan!
Nyan!
Nyan!
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}

int i 3

Console Program
Nyan!
Nyan!
Nyan!
Nyan!
Nyan!
```java
for (int i = 0; i < 4; i++) {
    println("Nyan! ");
}
```

```
int i  4
```

Console Program

Nyan!
Nyan!
Nyan!
Nyan!
Nyan!
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}

int i 4

Console Program
Nyan!
Nyan!
Nyan!
Nyan!
Nyan!
for (int i = 0; i < 4; i++) {
    println("Nyan!");
}