Software Adaptability

List some of the limitations of DrawingGizmo. (i.e. what methods would you like to see added to the class?)

Such limitations result from design decisions that were made by the DrawingGizmo creators.

<table>
<thead>
<tr>
<th>DrawingGizmo</th>
</tr>
</thead>
<tbody>
<tr>
<td>«constructor»</td>
</tr>
<tr>
<td>DrawingGizmo()</td>
</tr>
<tr>
<td>«update»</td>
</tr>
<tr>
<td>void moveForward()</td>
</tr>
<tr>
<td>void turnClockwise()</td>
</tr>
<tr>
<td>void turnCounterClockwise()</td>
</tr>
<tr>
<td>void dontDraw()</td>
</tr>
<tr>
<td>void draw()</td>
</tr>
<tr>
<td>void delay2Sec()</td>
</tr>
</tbody>
</table>

One way to improve software adaptability is...
Parameter Passage

Parameter passage permits one method to vary its behavior in response to argument values that are supplied by the call instruction.

This class diagram reveals five additional methods moveBy, turnBy, delayBy, setBackground and setForeground that contain parameters.

In Java “int” means integer.

The _______ parameter indicates the move distance in pixels.

The _______ parameter indicates the rotation amount in degrees.

The _______ parameter indicates the delay amount in milliseconds.

When calling a method with a parameter(s) the call instruction must supply the same number and type of arguments.

pen = new DrawingGizmo();
pen.turnBy(30);
pen.moveBy(-12);
pen.delayBy(1);

Why Parameters?

Parameter passage can support new possibilities.

pen.turnBy(2);

Parameter passage can allow better forms of expression.

(Consider how to write this more succinctly...)

pen.turnClockwise();
pen.turnClockwise();
pen.turnClockwise();

Which DrawingGizmo methods are redundant, given turnBy?

© 2006 Pearson Addison-Wesley. All rights reserved
DrawingTool Class Specifications (additional methods)

Update Methods

public void moveBy(int d)
    pre: A minimum of d pixels remain between the arrow and the edge of
        Drawing Canvas in the direction of travel.
    post: If this object's is moved forward by d pixels from its previous location, and
        if this object is in drawing mode, then a line segment is drawn across
        the path just traversed. (In moving mode nothing is drawn.)

public void turnBy(int r)
    post: This object's direction of travel is rotated by r degrees clockwise from its
        previous direction.

public void delayBy(int t)
    post: All drawing activity for the drawing canvas of this DrawingGizmo
        is suspended for t milliseconds, then resumed.

Q: What are the setBackground and setForeground methods?
A: setBackground establishes the canvas color
    setForeground establishes line DrawingGizmo color for drawing

"______________________" is
the standard Java class for color
objects.

Example
pen = new DrawingGizmo();
pen.setBackground(java.awt.Color.blue);
pen.setForeground(java.awt.Color.green);
pen.moveBy(30);
pen.turnBy(90);
pen.setBackground(java.awt.Color.black);
pen.moveBy(30);
More About Color

Supported Colors

- java.awt.Color.black
- java.awt.Color.blue
- java.awt.Color.cyan
- java.awt.Color.darkGray
- java.awt.Color.gray
- java.awt.Color.green
- java.awt.Color.lightGray
- java.awt.Color.magenta
- java.awt.Color.orange
- java.awt.Color.pink
- java.awt.Color.red
- java.awt.Color.white
- java.awt.Color.yellow

Import

An import declaration permits abbreviated notation for library members.

Without import

```java
public class Driver {
    private DrawingGizmo pen;

    public Driver() {
        pen = new DrawingGizmo();
        pen.setBackground(java.awt.Color.blue);
        pen.setForeground(java.awt.Color.yellow);
        pen.moveBy(30);
        pen.turnBy(90);
        pen.setForeground(java.awt.Color.gray);
        pen.moveBy(30);
    }
}
```

With import

```java
import java.awt.Color;

public class Driver {
    private DrawingGizmo pen;

    public Driver() {
        pen = new DrawingGizmo();
        pen.setBackground(Color.blue);
        pen.setForeground(Color.yellow);
        pen.moveBy(30);
        pen.turnBy(90);
        pen.moveBy(30);
    }
}
```

```java
import java.awt.Color;  // grants abbreviated access to java.awt.Color
import java.awt.*;    // grants abbreviated access to _____ classes in java.awt
```