Patterns

The task of computer programming requires knowledge of patterns.

A pattern is a general template that can be applied to solve specific problems.

**Example Code Pattern: Using a variable**

This pattern explains the order of events that is required to utilize a variable.

1) Declare the variable (name & type).
2) Initialize the variable.
3) Use the variable.
4) Dispose of the variable. (optional)
**Swap Pattern**

**Example Code Pattern: Swap**

This pattern is a code segment that interchanges the binding (for reference data) or value (for primitive data) of two variables.

**preAssertion**

- \(var1\) and \(var2\) are any two variables of the same type,
- call this type \(someType\)

**postAssertion**

\[
\begin{align*}
var1 &= var2@pre \\
and \quad var2 &= var1@pre
\end{align*}
\]

("\(varX@pre\)" denotes the value/binding of \(varX\) before the pattern executes.

---

**Example Swap**

**Problem**

A program to place two circles. The LEFT button should make one of the two circles grow, and the RIGHT button should select which is to grow. (Each RIGHT click should select the alternate circle.)

```java
private Oval growingCircle, fixedCircle;

public void leftAction() {
    growingCircle.setSize((int)(1.2 * growingCircle.getWidth()),
                           (int)(1.2 * growingCircle.getHeight()));
    growingCircle.repaint();
}

public void rightAction() {
    Oval tmpCircle;
    tmpCircle = growingCircle;
    growingCircle = fixedCircle;
    fixedCircle = tmpCircle;
}
```

© 2006 Pearson Addison-Wesley. All rights reserved
Sometimes a pattern can be as simple as a single expression.

### Accurate Integer Division Expression Pattern

Integer division (quotient) leads to a loss of accuracy. For maximum accuracy when dividing `intExpr1` by `intExpr2` the following double expression can be used.

### Conversion to int Expression Pattern

This pattern assigns the **truncated** value of `doubleExpr` to `intVar`.

\[
\text{intVar} = \text{(int)}(\text{doubleExpr});
\]

This pattern assigns the **rounds** value of `intExpr` to `doubleVar`.

\[
\text{OR}
\]

### The Math Class

<table>
<thead>
<tr>
<th>Math</th>
<th>doubleVar = Math.pow(6, 3);</th>
</tr>
</thead>
<tbody>
<tr>
<td>double final PI</td>
<td>doubleVar = Math.sqrt(10);</td>
</tr>
</tbody>
</table>

- `int abs(int )`
- `double abs(double)`
- `double cos(double)`
- `double sin(double)`
- `double tan(double)`
- `double pow(double, double)`
- `double random()`
- `long round(double)`
- `double sqrt(double)`
- `double toDegrees(double)`
- `double toRadians(double)`

doubleVar = Math.cos(3.14159);

doubleVar = Math.sin(Math.toRadians(30));

intVar = Math.abs(-3);
Incrementing Pattern

Frequently, variables need to be increased/decreased by some fixed value.

Increment by n Expression Pattern

This pattern increases the value of \texttt{intVar} by the value \texttt{n}

Example

```java
/** post  xPosition == xPosition@pre + 10
*   and  dot.getX() == xPosition */

public void leftAction() {
    xPosition = xPosition + 10;
    dot.setLocation( xPosition, dot.getY() );
}
```

© 2006 Pearson Addison-Wesley. All rights reserved

Incrementing Variations

Increment by 1 Expression Pattern

This pattern increases the value of \texttt{intVar}, by the value 1

\texttt{intVar}++;

Decrement by 1 Expression Pattern

This pattern decreases the value of \texttt{intVar}, by the value 1

\texttt{intVar}--;

Running Total Update Expression Pattern

This pattern updates a running total, called \texttt{intTotal}, by the value \texttt{n}

\texttt{intTotal = intTotal + n;}

Example

Add the diameter of dot to the total of all dot diameters.
\texttt{diameterTot = diameterTot + dot.getWidth();}

\texttt{System.out.println("Total of all diameters is " + diameterTot );}

© 2006 Pearson Addison-Wesley. All rights reserved
General Counting Expression Pattern

This pattern uses the increment pattern to count one more occurrence of some situation.

```
intCounter ++;
```

Example
Count the number of times LEFT is clicked.
```
public void leftAction() {
    
}
```

Modulus Counting Expression Pattern

This pattern counts from 0 to \( n \) then back to zero \( 0, 1, 2, \ldots, n, 0, 1, 2, \ldots \).

```
intCounter = (intCounter + 1) % (n + 1);
```

Example
Count the number of times that LEFT modulo 4. (i.e., 0, 1, 2, 3, 0, 1, 2, 3, ...)
```
public void leftAction() {
    leftCounter = (leftCounter + 1) % 4;
}
```

Random Number Generation

Random double Expression Pattern

This results in a double randomly selected from the range from \( \text{minDouble} \) through \( \text{maxDouble} \).

```
\text{minDouble} + \text{Math.random()} * (\text{maxDouble} - \text{minDouble})
```

Examples
Pick a number from 10.0 to 25.0
```
System.out.println("I pick " + (10 + Math.random() * 15));
```

Random int Expression Pattern

This results in an integer randomly selected from the range from \( \text{minInt} \) through \( \text{maxInt} \).

```
(int)((\text{minInt} + \text{Math.random()} * (\text{maxInt} - \text{minInt}) + 1)
```

Examples
Generate a random roll of one die
Select a dot diameter from the range of 10 through 100.
```
dot.setSize( diameter, diameter);
```