A Java application organized according to the IPO model would be structured as follows:

```java
public class MyApp
{
    public static void main( String [] args )
    {
        // STATEMENTS THAT DECLARE DATA
        // STATEMENTS THAT INPUT DATA
        // STATEMENTS THAT PROCESS DATA
        // STATEMENTS THAT OUTPUT DATA
    }
}
```

**Example**
This Java application reads the radius of circle and prints its diameter.

```java
import java.util.Scanner;
public class Circle
{
    public static void main( String [] args )
    {
        // declare data
        double radius, diameter;
        // input data
        Scanner in = new Scanner( System.in );
        System.out.print( "Enter circle's radius: " );
        radius = in.nextDouble( );
        // calculate diameter
        diameter = 2 * radius;
        // print results
        System.out.print( "A circle with radius " + radius );
        System.out.println( " has diameter " + diameter );
    }
}
```
Statements That Declare Data
Java requires that a variable be declared before using it. You do this with a variable declaration that informs the compiler of the variable’s data type and identifier. It has this form:

```
data type  variable identifier
```

**Example**
The Java statements below declare two variables identified as `radius` and `diameter`. Both have data type `double`, which means that they can hold floating-point values (i.e. numbers with a fractional part).

```
double radius;
double diameter;
```

In Java, it is always an error to use a variable without having previously declared it.

**Example**
This Java application does not declare its variables `radius` and `diameter`. The Java compiler cites an error on line 5, saying:

```
MyApp.java:5: error: cannot find symbol
radius = 6.25;
```

The message “cannot find symbol” refers to the fact that it cannot find the declaration of symbol `radius`. It cites a similar error for `diameter` on line 6.

```
1 public class MyApp
2 {
3     public static void main( String [] args )
4         {
5             radius = 6.25;
6             diameter = 2 * radius;
7             System.out.println( radius + " " + diameter );
8         }
9     }
```
But it is also an error to declare the same variable twice.

**Example**
In this Java application `diameter` is declared on line 5 and again on line 7. The Java compiler cites an error, saying:

```
MyApp.java:7: error: diameter is already defined
double diameter = 2 * radius;
```

A variable can be initialized within its declaration statement. To *initialize* a variable means to give it a starting value.

**Example**
The statements below declare and initialize variables `radius` and `diameter`. The conceptual picture at right shows each variable holding its value.

```
double radius = 6.25;
double diameter = 2 * radius;
```

<table>
<thead>
<tr>
<th>radius</th>
<th>diameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.25</td>
<td>12.5</td>
</tr>
</tbody>
</table>
Java has the additional requirement that you must initialize a variable before being allowed to use it in a computation.

**Example**
This Java application does not initialize the variable `radius` before attempting to use it to calculate `diameter`. The Java compiler cites an error on line 6, saying:

```
  MyApp.java:6: error: variable radius might not have been initialized
double diameter = 2 * radius;
```

```
public class MyApp {
  public static void main( String [] args ) {
    double radius;
    double diameter = 2 * radius;
    System.out.println( radius + " " + diameter );
  }
}
```

Using the comma (,) as a delimiter, several variables of the same data type can be declared &or initialized within a single declaration statement.

**Example**

<table>
<thead>
<tr>
<th>double radius, diameter;</th>
<th>radius</th>
<th>diameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>double radius = 6.25, diameter = 2 * radius;</td>
<td>radius</td>
<td>6.25</td>
</tr>
<tr>
<td></td>
<td>diameter</td>
<td>12.5</td>
</tr>
</tbody>
</table>
**Statements that Input Data**

A Java application can input data from a human user through the *standard input object*, named `System.in`, which is ready and waiting whenever the application is running. The standard input object collects input as alphanumeric characters, which you must scan into binary using an object of class `java.util.Scanner`.

```
Example
This Java statement builds a Scanner object.

Scanner in = new Scanner( System.in );
```

Since Scanner is a class identifier that is not declared within the Java file, you must use an `import` statement to inform the compiler where its declaration resides.

```
Example
This Java statement informs the compiler that the Scanner class is declared in the java.util package.

import java.util.Scanner;
```

The scanner object has internal methods that input and scan data. One such method is named `nextDouble`, which can be used to input a floating-point value.

```
Example
This statement inputs a value for the double variable radius from the standard input object.

radius = in.nextDouble( );
```

It is common practice to inform the user that you expect input by printing an *input prompt* before waiting for input:

```
Example
Here’s the complete input operation. The print statement warns the user that input is expected.

Scanner in = new Scanner( System.in );
System.out.print( "Enter circle's radius: " );
radius = in.nextDouble( );
```
**Statements that Process Data**

Statements that process data include assignment statements and expressions. An *assignment statement* stores a value into a variable. In Java, it has the form:

```
variable identifier assignment operator value
```

There are several assignment operators, but the most common is the equal sign ( = ). The *value* to the right of the assignment operator can be a constant, another variable or the result of a computation.

**Example**

```java
double x, y, z;
x = 3.5;
y = x;
z = 4 * (x + y);
```

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>3.5</td>
</tr>
<tr>
<td>y</td>
<td>3.5</td>
</tr>
<tr>
<td>z</td>
<td>28.0</td>
</tr>
</tbody>
</table>

An *expression* tells the computer to perform calculations such as addition (+), subtraction (−), multiplication (×) and division (/). The computer performs the calculations by retrieving the data stored in the variables and applying the specified operators.

**Example**

```java
double x, y;
y = 3.5;
x = ( y + 6.5 ) / 2;
```

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>5.0</td>
</tr>
<tr>
<td>y</td>
<td>3.5</td>
</tr>
</tbody>
</table>
Statements that Output Data
A Java application can send output meant for a human user to the standard output object, named `System.out`, which is automatically ready and waiting whenever the application is executing. To do this, `System.out` has a method named `print`, which you can call using the syntax:

```
System.out.print( item to print )
```

<table>
<thead>
<tr>
<th>Example</th>
<th>outputs</th>
</tr>
</thead>
</table>
| double height = 4.5;  
System.out.print( height );  
System.out.print( " feet" ); | 4.5 feet |

`System.out` also has a method named `println` (standing for print line), which works just like `print` but terminates the output line so that what follows is on the next line down.

<table>
<thead>
<tr>
<th>Example</th>
<th>outputs</th>
</tr>
</thead>
</table>
| double height = 4.5;  
System.out.println( height );  
System.out.println( " feet" ); | 4.5 feet |

The string concatenation operator (+) appends one alphanumeric string onto the end of another. If one operand is a string and the other a numeric data type then Java automatically converts the number to string form. This allows you to output several items within a single `print` or `println`.

<table>
<thead>
<tr>
<th>Example</th>
<th>output is</th>
</tr>
</thead>
</table>
| double height = 4.5;  
System.out.println( height + " feet" ); | 4.5 feet |
## Exercises

For each Java application below, circle what's wrong and explain. None of them is correct.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td><code>public class MyApp</code>&lt;br&gt;<code>{</code>&lt;br&gt;<code>   public static void main ( String [] args )</code>&lt;br&gt;<code>   {</code>&lt;br&gt;<code>     feet = 3;</code>&lt;br&gt;<code>     inches = 12 * feet;</code>&lt;br&gt;<code>     . . .</code>&lt;br&gt;<code>   }</code>&lt;br&gt;<code>}</code></td>
</tr>
<tr>
<td>2.</td>
<td><code>public class MyApp</code>&lt;br&gt;<code>{</code>&lt;br&gt;<code>   public static void main ( String [] args )</code>&lt;br&gt;<code>   {</code>&lt;br&gt;<code>     double feet;&lt;br&gt;</code>     double inches;&lt;br&gt;<code>     Feet = 3;&lt;br&gt;</code>     Inches = 12 * Feet;&lt;br&gt;<code>     . . .</code>&lt;br&gt;<code>   }</code>&lt;br&gt;<code>}</code></td>
</tr>
<tr>
<td>3.</td>
<td><code>public class MyApp</code>&lt;br&gt;<code>{</code>&lt;br&gt;<code>   public static void main ( String [] args )</code>&lt;br&gt;<code>   {</code>&lt;br&gt;<code>     double feet;&lt;br&gt;</code>     double inches;&lt;br&gt;<code>     feet = 3;&lt;br&gt;</code>     double inches = 12 * feet;&lt;br&gt;<code>     . . .</code>&lt;br&gt;<code>   }</code>&lt;br&gt;<code>}</code></td>
</tr>
</tbody>
</table>
For each Java application below, circle what's wrong and explain. None of them is correct.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
</table>
| 4. | `public class MyApp`  
   | `{`  
   | `public static void main ( String [] args )`  
   | `{`  
   | `double feet;`  
   | `double inches;`  
   | `inches = 12 * feet;`  
   | `...`  
   | `}`  
   | `}`  

5. `public class MyApp`  
   `{`  
   `public static void main ( String [] args )`  
   `{`  
   `double feet`  
   `double inches`  
   `feet = 3`  
   `inches = 12 * feet`  
   `...`  
   `}`  
   `}`

For each code fragment below, circle what's wrong and explain. None of them is correct.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
</table>
| 6. | `Scanner in = new Scanner( );`  
   | `double x = in.nextDouble( );`  

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
</table>
| 7. | `scanner in = new scanner( System.in );`  
   | `double x = in.nextDouble( );`  

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
</table>
| 8. | `Scanner in = Scanner( System.in );`  
   | `double x = nextDouble( );`  

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
</table>
| 9. | `Scanner in = new Scanner( System.in );`  
   | `double x = in.nextDouble;`
Ignoring that the variables aren’t declared, for each code fragment below, circle what's wrong and explain. None of them is correct.

<table>
<thead>
<tr>
<th></th>
<th>Code Fragment</th>
</tr>
</thead>
</table>
|10.| $3 = \text{feet};$
   | $12 \times \text{feet} = \text{inches};$         |
|11.| $\text{feet} = "3";$
   | $\text{inches} = "12" \times \text{feet};$       |
|12.| "\text{feet}" = 3;
   | "\text{inches}" = 12 \times "\text{feet}";      |
|13.| $\text{feet} = 3$
   | $\text{inches} = 12 \times \text{feet}$         |
|14.| $\text{feet} = 3;$
   | $\text{inches} = 12 \times \text{feet} ;$       |

Ignoring that the variables aren’t declared or initialized, for each output statement below, circle what's wrong and explain. None of them is correct.

<table>
<thead>
<tr>
<th></th>
<th>Output Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.</td>
<td><code>System.out.println( 'x = ', x );</code></td>
</tr>
<tr>
<td>16.</td>
<td><code>System.out.println( &quot;He cried, &quot;STOP!&quot;&quot; );</code></td>
</tr>
</tbody>
</table>

For each, give the output. Assume the variable $a$ is declared and holds the value 102.546.

<table>
<thead>
<tr>
<th></th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>17.</td>
<td><code>System.out.println( &quot;Tom&quot; + &quot;Jones&quot; );</code></td>
</tr>
<tr>
<td>18.</td>
<td><code>System.out.println( &quot;Tom &quot; + &quot;Jones&quot; );</code></td>
</tr>
<tr>
<td>19.</td>
<td><code>System.out.println( &quot;a = &quot; );</code></td>
</tr>
<tr>
<td><code>System.out.println( &quot;a&quot; );</code></td>
<td></td>
</tr>
<tr>
<td>20.</td>
<td><code>System.out.println( &quot;a = &quot; );</code></td>
</tr>
<tr>
<td><code>System.out.println( a );</code></td>
<td></td>
</tr>
<tr>
<td>21.</td>
<td><code>System.out.print( &quot;a = &quot; );</code></td>
</tr>
<tr>
<td><code>System.out.print( a );</code></td>
<td></td>
</tr>
<tr>
<td>22.</td>
<td><code>System.out.print( &quot;a = &quot; );</code></td>
</tr>
<tr>
<td><code>System.out.println( );</code></td>
<td></td>
</tr>
</tbody>
</table>
For each, give the output. Assume the variable `a` is declared and holds the value **102.546**.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>23.</td>
<td><code>System.out.println( &quot;a = &quot; + a );</code></td>
</tr>
<tr>
<td>24.</td>
<td><code>System.out.println( &quot;a = &quot; + &quot;a&quot; );</code></td>
</tr>
<tr>
<td>25.</td>
<td><code>System.out.println( &quot;a=&quot; + a );</code></td>
</tr>
<tr>
<td>26.</td>
<td><code>System.out.println( &quot;a = &quot; + a );</code></td>
</tr>
</tbody>
</table>

Give the output of each code fragment.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
</table>
| 27. | `double x;  
x = 87.65432;  
System.out.print( x );` |
| 28. | `double x;  
x = 87.65432;  
System.out.print( "x = " );  
System.out.print( x );` |
| 29. | `double x;  
x = 87.65432;  
System.out.println( "x = " );  
System.out.println( x );` |
| 30. | `double x;  
x = 87.65432;  
System.out.println( "x = " + x );` |
| 31. | `double x, y;  
x = 10;  
y = 20;  
System.out.println( "x = " + x );  
System.out.println( "y = " + y );` |
| 32. | `double x, y;  
x = 10;  
y = 20;  
System.out.print( "x =" + x );  
System.out.print( "y =" + y );` |
<table>
<thead>
<tr>
<th></th>
<th><strong>Write the Java statement to declare a double variable identified as salesTaxRate, initialized to the value 0.60.</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>34.</td>
<td>Write the Java statement to declare a double variable identified as pi, initialized to the value 3.14159.</td>
</tr>
<tr>
<td>35.</td>
<td>Write the Java statements to declare a double variable identified as x and read its value from the standard input object (hint: you’ll need to build a Scanner object).</td>
</tr>
<tr>
<td>36.</td>
<td>Suppose variable feet is 5.389 and inches is 4.67. Write print and/or println statements that use these variables to output these lines to the standard output object:</td>
</tr>
<tr>
<td></td>
<td>feet = 5.389</td>
</tr>
<tr>
<td></td>
<td>inches = 4.67</td>
</tr>
</tbody>
</table>