A DrawingGizmo object can be moved around a window to draw lines. It appears like an arrow. (See the green arrow to the right.)

The Class Diagram

<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>

Method Call

The method call is the primary instruction in a Java program. When a method call is executed, the operation associated with the method is applied to an object.

Method Call Syntax

where
- objectReference refers to some particular object, and
- methodName is the name of a parameterless method from the class to which objectName belongs.

Examples (assume an object, pen, of type DrawingGizmo)

<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>
Within a Java program instructions are also called ___________.

A sequence of statements (instructions) is executed in order from top to bottom.

Sample Java  (assume an object, *pen*, of type *DrawingGizmo*)

```java
pen.draw();
pen.moveForward();
pen.moveForward();
pen.turnClockwise();
pen.dontDraw();
pen.moveForward();
```

<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>

Objects don’t exist until they are instantiated. Attempting to call methods on objects before they are instantiated is an error.

Syntax to Instantiate & Assign

```java
objectName = _____ constructorName();
```

where

- *objectName* is the name (i.e., variable) of some particular object, and

- *constructorName* is the name of a parameterless constructor method from the class to which *objectName* belongs. (Note: In Java constructors have the same name as the class.)

Example  (assume an object, *pen*, of type *DrawingGizmo*)
Programming languages, like Java, are extremely particular about notation (syntax). Syntax diagrams are a pictorial way to define permissible syntax.

**MethodCall** (abridged version)

- **ObjectReference**
- .
- **MethodName**
- ( )
- ;

Notes:
- **ObjectReference** is a variable name or other acceptable object reference
- **MethodName** is the name of a parameterless method from the class to which **ObjectReference** belongs.
- No white space is permitted before or after the period ( ).

**AssignmentInstruction**

- **Variable**
- =
- **Expression**
- ;

Notes:
- **Variable** is a variable name
- **Expression** must have a type that conforms to the type of **Variable**.

**Expression** (abridged version)

- **Variable**
- new
- **Constructor**

Note:
- **Constructor** names a valid parameterless constructor.

**StatementSequence**

**OneStatement**

**OneStatement** (abridged version)

- **MethodCall**
- **AssignmentInstruction**
**DriverClass** (abridged version of **Class**)

```
public class Driver {
    public Driver() {
        
    }
}
```

**Semantic**
Instantiating a Driver object causes **StatementSequence** from Driver to execute.

**Note**
- **ImportDecls** and **PrivateMethod** are explained later.

---

**Declaring Instance Variables**

**InstanceVarDecls**

```
OneVarDecl

PrivateVarDecl (a version of **OneVarDecl**)  

private ClassName Variable;  
```

**Notes**
- The scope of each **Variable** is the class in which it is declared.
- Each **Variable** must be named by an identifier that is unique for the class.

**Example**
```
private DrawingGizmo pen;
private DrawingGizmo pencil;
```

**Alternative**
```
private DrawingGizmo pen, pencil;
```
A First Java Program

When this program executes, it draws a square.

```java
public class Driver {
    private DrawingGizmo pen;
    public Driver() {
        pen = new DrawingGizmo();
        pen.draw();
        pen.moveForward();
        pen.turnClockwise();
        pen.turnClockwise();
        pen.turnClockwise();
        pen.moveForward();
        pen.turnClockwise();
        pen.turnClockwise();
        pen.turnClockwise();
        pen.moveForward();
        pen.turnClockwise();
        pen.turnClockwise();
        pen.turnClockwise();
        pen.moveForward();
        pen.turnClockwise();
        pen.turnClockwise();
        pen.turnClockwise();
    }
}
```

More Java Syntax

Identifiers

Program entities are named by __________.

Identifiers name...

- each class -- like __________
- each variable -- like __
- each method -- like __________
- and other things (stay tuned)

Syntax

- begin with one alphabetic character (a-z or A-Z)
- followed by zero or more consecutive alphabetic and/or numeric (0-9) characters. (Underscores and dollars sign are reserved for special usage.)

Notes

- Take care to use different names for different entities.
- Java identifiers are case sensitive (i.e., capital letters are different than small letters).
- Identifiers should be meaningful to enhance software readability.
Programmers frequently include notes within their programs. Such notes are called comments.

**Comment**

```
//  anythingOnOneLine
/*  anythingButCommentEnd  */
```

**Notes**

- `anythingOnOneLine` is any sequence of characters up to the end of the text line.
- `anythingButCommentEnd` is any sequence of characters up to the first `*/`.
- `Comment` is permitted anywhere white space is allowed.

**Semantics**

Executing a comment has no effect at run-time.

**Why Comment?**

Comments can assist you and others in understanding a program.

```java
/** Program to draw two 30 degree angles
 *  @author David Riley
 *  @version August, 2005
 */
public class Driver {
    private DrawingGizmo pen, pencil;

    public Driver() {
        pen = new DrawingGizmo();
        pen.draw();
        pen.moveForward();
        pencil = new DrawingGizmo();
        pencil.turnClockwise();
        pencil.dontDraw();
        pencil.moveForward();
        pen.turnClockwise();
        pen.moveForward();
        pencil.draw();
        pencil.moveForward();
        pencil.turnClockwise();
        pencil.moveForward();
    }
}
```