Methods in Java -- The relevant BNF syntax descriptions from Chapter 5.

**BNF 5.6 Method Declaration**

<table>
<thead>
<tr>
<th>(&lt;\text{method decl}&gt; ::= \langle\text{method heading}\rangle \langle\text{method body}\rangle</th>
<th>(\text{Semantics})</th>
</tr>
</thead>
<tbody>
<tr>
<td>A method declaration is never executed. Nevertheless, it does have a meaning, namely: Create a method for the current class with the signature declared in the method heading. The body of the method is executed when the corresponding message is sent.</td>
<td></td>
</tr>
</tbody>
</table>

**BNF 5.7 Method Heading**

<table>
<thead>
<tr>
<th>(&lt;\text{method heading}&gt; ::= [\langle\text{access}\rangle] [\langle\text{return type}\rangle] \langle\text{identifier}\rangle (\langle\text{formal parameters}\rangle)</th>
<th>(\text{Semantics})</th>
</tr>
</thead>
<tbody>
<tr>
<td>A method heading is never executed. It defines the signature of the method. Constructors do not have a return type. For ordinary methods, if the return type is not void, the method body must end with a return statement whose (&lt;\text{expression}\rangle) has a type compatible with the (&lt;\text{return type}\rangle).</td>
<td></td>
</tr>
</tbody>
</table>

**Return Types**

Every method that is not a constructor must have a return type. Any legal type may be a return type. If nothing is returned, the return type must be declared as void.

**BNF 5.8 Method Body**

<table>
<thead>
<tr>
<th>(&lt;\text{method body}&gt; ::= \langle\text{block statement}\rangle</th>
<th>(\text{Semantics})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Same as the block statement.</td>
<td></td>
</tr>
</tbody>
</table>

**BNF 5.9 Block Statement**

<table>
<thead>
<tr>
<th>(&lt;\text{block statement}&gt; ::= { \langle\text{statement}\rangle}^*</th>
<th>(\text{Semantics})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute each statement in the block in order.</td>
<td></td>
</tr>
</tbody>
</table>

**BNF 5.10 Formal Parameters**

<table>
<thead>
<tr>
<th>(&lt;\text{formal parameters}&gt; ::= \langle\text{formal parameter}\rangle , \langle\text{formal parameter}\rangle^*</th>
<th>(\text{Semantics})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Declares one or more local variables which are created when the method is invoked and destroyed when it returns.</td>
<td></td>
</tr>
</tbody>
</table>

**BNF 5.11 Actual Parameters**

<table>
<thead>
<tr>
<th>(&lt;\text{actual parameters}&gt; ::= \langle\text{actual parameter}\rangle , \langle\text{actual parameter}\rangle^*</th>
<th>(\text{Semantics})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Each expression is evaluated and its value copied to the corresponding formal parameter before the message is sent; this is called parameter linkage.</td>
<td></td>
</tr>
</tbody>
</table>
Review Questions (to be handed in if you scored <7 on the quiz on 10/15)

For the Foo class defined below:

1. What are the names of the variables? What are the names of the methods?
2. What are the return types of each method?
3. What are the parameter types of each method?
4. What are the names of the parameters of each method?
5. Draw a square around each method heading.
6. Draw a diamond around each formal parameter.
7. Circle the body of each method.
8. What kind of statement is in each body?
9. Draw a double diamond around each actual parameter.
10. Draw lines connecting each actual parameter to its corresponding formal parameter.
11. What is output when public static void main is executed?

```java
class Foo {
    int x;

    void sayHi() {
        System.out.println("hi!");
    }

    void setX(int nuX) {
        x = nuX;
    }

    int getX() {
        return x;
    }

    int ave(int x, int y, int z) {
        return (x + y + z) / 3;
    }

    public String toString() {
        return "Foo! x=" + x
            + " without this method, Object would say: "
            + super.toString();
    }

    public static void main(String[] args) {
        Foo myFoo = new Foo();
        myFoo.setX(123);
        myFoo.sayHi();
        System.out.println(myFoo);
        myFoo.setX(17);
        System.out.println("now x *should* be 17 " + myFoo
            + " getX returns: " + myFoo.getX());
        System.out.println("the average of 4, 7, and 10 is: "
            + myFoo.ave(4, 7, 10));
    }
}
```