METHOD SPECIFICATIONS

To correctly call a method and use its result, you need to be able to read and understand its specification. A well-written method specification includes the following information.

<table>
<thead>
<tr>
<th>What a Well-Written Method Specification Should Tell You</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. The name of the package and class in which the method is defined.</td>
</tr>
<tr>
<td>2. The method identifier to be used to call it.</td>
</tr>
<tr>
<td>3. Its parameters; i.e. what data you must pass it as arguments.</td>
</tr>
<tr>
<td>4. What it does and its return value, if any.</td>
</tr>
<tr>
<td>5. Whether it is an instance or a class method.</td>
</tr>
</tbody>
</table>

Programmers usually write method specifications in Java code supplemented by comments.

Example
Here is the specification of `currentTimeMillis` found in the Java API class `java.lang.System`.

```
static long currentTimeMillis( )
// Returns the current time counted in milliseconds since
// midnight January 1, 1970 UTC.
```

1. It is found in `java.lang.System`.
2. Its identifier is `currentTimeMillis`.
3. It has no parameters and, therefore, requires no arguments to call it.
4. It returns the current time as a value whose data type is `long`.
5. It is a class method since it is tagged `static`. 
**Example**
Here is the specification of `nextInt` found in the Java API class `java.util.Scanner`.

```java
int nextInt( )
// Scans and returns the next token of the input as an int.
```

1. It is found in `java.util.Scanner`.
2. Its identifier is `nextInt`.
3. It has no parameters and, therefore, requires no arguments to call it.
4. It returns an `int` value taken from the current input object.
5. It is an instance method since it isn’t tagged `static`.

**Example**
Here’s the specification of `pow` found in API class `java.lang.Math`.

```java
static double pow( double a, double b )
// Returns the value of the first argument raised to the
// power of the second argument.
```

1. It is found in `java.lang.Math`.
2. Its identifier is `pow`.
3. It has two parameters both of which must have the data type `double`. If you want to calculate $x^y$ the first argument must be $x$ and the second $y$.
4. It returns a `double` value that equals $x^y$, where $x$ and $y$ are the arguments.
5. It is a class method since it is tagged `static`.

How to use the facts given in the method specification in order to write a correct method call is the subject of the topics that follow.