2.4 Methods

2.4.1 Defining Methods

Here is an example of a typical method declaration:

```
public double calculateAnswer(double wingSpan, int numberOfEngines,
                    double length, double grossTons) {
    //do the calculation here
}
```

The only required elements of a method declaration are the method's return type, name, a pair of parentheses, ( ), and a body between braces, { }.

More generally, method declarations have six components, in order:

1. **Modifiers** — such as public, private, and others you will learn about later.
2. **The return type** — the data type of the value returned by the method, or void if the method does not return a value.
3. **The method name** — the rules for field names apply to method names as well, but the convention is a little different.
4. **The parameter list in parenthesis** — a comma-delimited list of input parameters, preceded by their data types, enclosed by parentheses, (). If there are no parameters, you must use empty parentheses.
5. **An exception list** — to be discussed later.
6. **The method body, enclosed between braces** — the method's code, including the declaration of local variables, goes here.

Modifiers, return types, and parameters will be discussed later in this lesson. Exceptions are discussed in a later lesson.

Definition: Two of the components of a method declaration comprise the *method signature*—the method's name and the parameter types.

The signature of the method declared above is:
calculateAnswer(double, int, double, double)

2.4.2 Naming a Method

- ➢ Although a method name can be any legal identifier, code conventions restrict method names.
- ➢ By convention, method names should be a verb in lowercase or a multi-word name that begins with a verb in lowercase, followed by adjectives, nouns, etc.
- ➢ In multi-word names, the first letter of each of the second and following words should be capitalized.
- ➢ Here are some **examples**: run
  runFast
  getBackground
  getFinalData
  compareTo
  setX
  isEmpty

- ➢ Typically, a method has a unique name within its class.
- ➢ However, a method might have the same name as other methods due to *method overloading.*