Turning Recurring Code into Functions

Writing Method Statements

A method statement consists of a declaration and a body.
The method declaration contains:

- The method name.
- The parameters—parameters are written in a list separated by commas and enclosed in parentheses. If there are no parameters, then the parentheses are empty.
- The return type—the return type is the data type of the value that will be sent out of the method and is preceded by a colon. A return type of **void** means that the method will not return a value.

The method body contains:

- The code to be executed when the method is called. The code is enclosed in curly braces.

Example

The following code creates a method that defines a string parameter and then calls the method using the string "Hello" as the parameter value:

```
public static void showParameter(String aParam){
    System.out.println(aParam);
}

public static void main(String[] args) {
    showParameter("Hello");
}
```

Practice Exercise

1. Write a method called displayAge that defines an integer parameter. The method will display the value in displayAge. Write code to call the method using the variable studentAge as the parameter value.

Returning Values from Methods

To return a value from your method, use the return statement followed by the expression or literal value that you want to return.

Example 2

You will create a method that will double any number. When the method doubleNumber is called, an argument (value) will be sent into the method and stored in the variable baseNumber. This number will then be multiplied by 2 and the product will be returned out of the method as an integer.

```
public static int doubleNumber(int baseNumber){
        return (baseNumber*2);
}

public static void main(String[] args) {
        int answer;
        answer = doubleNumber(10);
        System.out.println(answer);          //displays 20
}
```

Practice Exercise

1. Write a method called calculatePay that defines two input parameters (numberOfHours and hourlyRate) and returns the amount of pay.
   Complete the conversion using the method shown above and then check your work.

Note: The return statement terminates the method; any statements below a return statement will not be executed.

Example 3

```
public static int doubleNumber(int baseNumber){
        return (baseNumber*2);
        System.out.println("after return");        //this statement will not execute
}
```

Passing arguments by value or by reference

Objects that we have, and will continue to be using, belong to the **primitive data types** which include boolean, double, int, and String. Java has special operators that make them behave as if they were passed by **value.** To be passed by **value** means that the value of the argument is **copied** into a local variable for use within the method and the original value is not affected.

Example 4

```
public static void passPrimitives(int xParam, int yParam){
        xParam++;
        yParam++;
        System.out.println(xParam + " " + yParam);
}
public static void main(String[] args) {
        int xValue = 10;
        int yValue = 15;
        System.out.println(xValue + " " + yValue);        // displays 10 15
        passPrimitives(xValue, yValue);                  // displays 11 16
        System.out.println(xValue + " " + yValue);        // displays 10 15
}
```