### **Methods**

* Methods are used to define reusable code.
* They simplify code by removing repetition.
* This is referred to as **modular** software development.

### **Method Definitions**

Methods are defined as follows:

<modifiers> <return value type> <method name> ( <parameters> ) {

}

Method Definition Example:

public static int max(int num1, int num2) {

int result = num1;

if (num1 > num2)

result = num1;

else result = num1;

return result;

}

* **Modifiers** are public static.
* Return value type is int.
* Method name is max.
* **Parameters** are int num1 and int num2.
* Method **Signature** is the method name and the parameter list.

### **Method Invocation**

Method calling example:

int i = max(2, 3);

Here the 2 and 3 are the actual parameters that are called **arguments**.

### **Void vs. Value Returning Methods**

* Methods typically return a specific type.
* But can also have methods that don't return anything; these will have return value type of void.
* Can use "return;" to exit a void returning method but not required.
* Example:

public static void printName(String name) {

System.out.println("Hello " + name);

}

### **Passing by Value**

* Arguments are typically "passed by value".
* This means a copy of the actual data is passed in.
* Example:

public class Increment {

public static void increment(int n) {

n++;

System.out.println(n);

}

public static void main(String[] args) {

int x = 1;

System.out.println(x);

increment(x);

System.out.println(x);

}

}

This prints

1

2

1

### **Overloading**

* Allows the use of the same method name for different methods.
* Can only do this if the signature is different.
* For Example:

public static int max(int num1, int num2) {

if (num1 > num2) return num1;

else return num2;

}

public static double max(double num1, double num2) {

if (num1 > num2) return num1;

else return num2;

}

public static double max(double num1, double num2, double num3) {

return max(max(num1, num2), num3);

}

### **Ambiguous Invocation**

* Sometimes the compiler may not be able to figure out which method to call.
* For Example:

public static double max(int num1, double num2) {

if (num1 > num2) return (double)num1;

else return num2;

}

public static double max(double num1, int num2) {

if (num1 > num2) return num1;

else return (double)num2;

}

* It is unclear which method fits max(1, 2). So, the compiler will return an error.

### **Scope of Variables**

* Scope is the part of the program where the variable can be used.
* For example:

public static int myMethod(int i) {

s1;

int j;

s2;

for (int k = 0; k < 5; k++) {

s3;

s4;

}

}

* i can be using within the whole method myMethod
* j can be used s2 onwards
* k can be used within the for loop
* A variable inside a method is called a **local variable** to the method.