Learning Objectives: Returning Values

- **Use the `return` keyword to return a value**
- **Identify the return value of the print statement**
- **Demonstrate the ability to return several different data types**
- **Create and apply helper methods**

Returning Values

The Return Keyword

Instead of just **printing** data, methods can also **return** data. Think of the `length` and `length()` methods. They return the length (in integer) of an array and string respectively. So the return value of these methods is of type `int`. Both `length` and `length()` do not print anything to the screen, they just return a number. From here on out, user-defined methods will avoid just printing to the screen. Instead, they will return a value. To return a value, simply use the `return` keyword.

```
/**
 * This method adds 5 to an integer
 *
 * @param num An integer
 * @return The integer added to 5
 */
public static int addFive(int num) {
  return(num + 5);
}

public static void main(String args[]) {
  addFive(10);
}
```

You'll notice the program no longer prints anything to the screen. That is because the method only adds `5` to whatever parameter is passed to the method and then returns it *internally*. To *see* the result, explicitly tell Java to print the return value of the method to the screen.

```java
/**
 * This method adds 5 to an integer
 *
 * @param num An integer
 * @return The integer added to 5
 */
public static int addFive(int num) {
  return(num + 5);
}

public static void main(String args[]) {
  int newNum = addFive(10);
  System.out.println(newNum);
}
```

challenge

What happens if you:

- Remove all lines of code within `main()` and replace them with just `System.out.println(addFive(10));`?

▼ What is the return value for methods that use `System.out.println()`?

If each method in Java has a return value, what is the return value for methods that use `System.out.println()`? Actually, you've seen it before and it is `void`. Methods that do not return a value are considered to be `void` methods.

```java
 /**
 * This method prints "Hello"
 *
 * @param No parameter
 * @return No return value
 */
 public static void printHello() { //void method
   System.out.println("Hello");
 }

 public static void main(String args[]) { //void method
   printHello();
 }
```

Returning Values

Methods can return any value in Java — ints, doubles, strings, ArrayLists, etc.

```
/**
 * This method adds two integers together
 *
 * @param x The first integer
 * @param y The second integer
 * @return x added to y
 */
public static int returnInt(int x, int y) { //int method
  return(x + y);
}

/**
 * This method adds two doubles together
 *
 * @param x The first double
 * @param y The second double
 * @return x added to y
 */
public static double returnDouble(double x, double y) { //double
        method
  return(x + y);
}

/**
 * This method adds two strings together
 *
 * @param x The first string
 * @param y The second string
 * @return x added to y
 */
public static String returnString(String x, String y) { //String
        method
  return(x + y);
}

public static void main(String args[]) { //void method
  System.out.println(returnInt(1, 2));
  System.out.println(returnDouble(1, 2));
  System.out.println(returnString("1", "2"));
}
```

challenge

Can you write a method that returns an ArrayList?

If you want to return an ArrayList, one possible approach is to have an ArrayList be passed as a parameter. You can then modify the ArrayList in some way, and then return it to the system.

▼ **Sample Code**

The code below takes an ArrayList of numbers as a parameter for the method `multiplyFive()`. The method creates a new empty ArrayList, multiplies each element of the parameter ArrayList by 5, and then adds those new products to the new ArrayList. Finally, the new ArrayList is returned.

```
public static ArrayList<Integer>
      multiplyFive(ArrayList<Integer> myList) {
  ArrayList<Integer> newList = new ArrayList<Integer>();
  for (Integer i : myList) {
    newList.add(i * 5);
  }
  return newList;
}

public static void main(String args[]) {
  ArrayList<Integer> numbers = new ArrayList<Integer>();
  numbers.add(1);
  numbers.add(2);
  numbers.add(3);
  numbers.add(4);
  numbers.add(5);

  System.out.println(multiplyFive(numbers));
}
```

Helper Methods

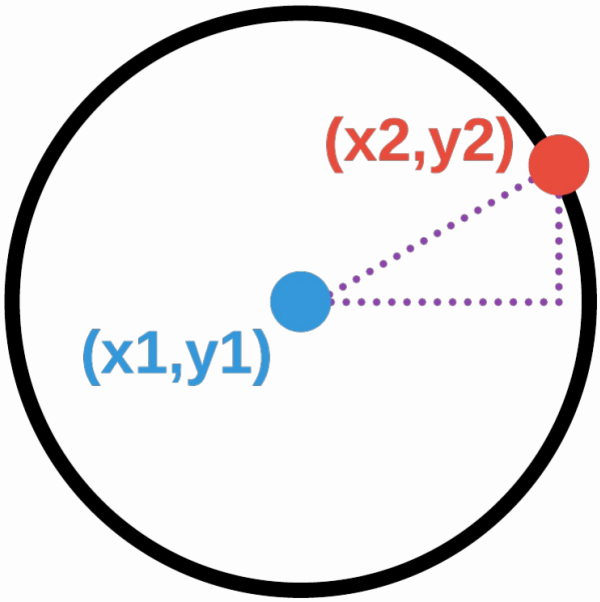
Helper Methods

Helper methods are methods that are called from within other methods. Take, for example, the formula for calculating the area of a circle:

$$A = \pi r^2$$

It would be quite easy to write a Java method to calculate the area of a circle. However, instead of knowing the radius of the circle, you have the X/Y coordinates for a point at the center of the circle and another point on the circle. The distance formula (which is based on the Pythagorean Theorem) can calculate the radius of the circle.

$$\sqrt{(x2 - x1)^2 + (y2 - y1)^2}$$



.guides/img/Radius

The `findRadius()` method uses the distance formula to calculate the distance between 2 pairs of points. The `findArea()` method finds the area of a circle by relying on the `findRadius()` method. Therefore, the

`findRadius()` method is a helper method. Helper methods help shorten how much code is needed to accomplish certain tasks.

```
/**
 * This method finds the radius of a circle given
 * two coordinate points
 *
 * @param x1 A double of the first x-coordinate
 * @param y1 A double of the first y-coordinate
 * @param x2 A double of the second x-coordinate
 * @param y2 A double of the second y-coordinate
 * @return The radius of a circle in double
 */
public static double findRadius(double x1, double y1, double x2,
        double y2) {
  return(Math.sqrt(Math.pow(x2 - x1, 2) + Math.pow(y2 - y1,
        2)));
}

/**
 * This method finds the area of a circle given
 * two coordinate points
 *
 * @param x1 A double of the first x-coordinate
 * @param y1 A double of the first y-coordinate
 * @param x2 A double of the second x-coordinate
 * @param y2 A double of the second y-coordinate
 * @return The area of a circle in double
 */
public static double findArea(double x1, double y1, double x2,
        double y2) {
  return(Math.PI * Math.pow(findRadius(x1, y1, x2, y2), 2));
}

public static void main(String args[]) {
  System.out.println(findArea(0.0, 0.0, 4.0, 4.0));
}
```

info

Math Methods

Note that in order to perform certain methods such as finding a square or an exponent, we imported the `Math` class as specified in the program header by `import java.lang.Math;`. If you remove the program header, the math methods associated with the `Math` class such as `Math.pow()` and `Math.PI()` will no longer work. In essence, these methods also serve as helper methods.