COURSEWARE

Enhanced For Loops

Contents

Overview

In Java, you have two types of `for` loops; basic `for` loops and enhanced `for` loops, otherwise known as `for-each` loops.
Enhanced `for` loops are better suited to iterating through arrays and collections of data.
The reason we use enhanced `for` loops, as opposed to basic `for` loops for looping through arrays and collections, is because it makes our code more readable, therefore making maintaining it much easier.
Enhanced `for` loops are also better used when iterating through an **entire** data set whereas basic `for` loops are better when we want to partially iterate over a data set due to being able to access the index.

Tutorial

Syntax

```
for(DataType item : array) {

}
```

The above example shows the syntax of an enhanced for loop.
We declare the `for` loop the same way we would a basic `for` loop, however the syntax within the parentheses changes slightly.
We first specify the data type that the array or collection holds, and then give it a reference variable name that it can put the value at the current iteration into.
We then use a colon ":" and specify the array or collection that we want to iterate through.

Example

```
public void printArray(String[] stringArray) {
    for(String str : stringArray) {
        System.out.println(str);
    }
}
```

In the above example, we iterate through `stringArray` pass the value to str, and then execute the body of the loop, in this case, print the value stored in str.
The variable str only stores the value at the current iteration of the loop, once the body of the loop has been executed, the next value in `stringArray` will be stored for the next execution of the method body.

Exercises

1. Create an array of strings and iterate through it, printing each value to console, using an enhanced `for` loop.

2. Create an array of integers 1-20 and iterate through it, using an enhanced `for` loop, square, and then print each value.

3. Create a method that returns a boolean and accepts an integer as a parameter, if the integer is even, return true, if not then return false.
4. Using the array of integers from exercise 2 and the method created in exercise 3; iterate through the array using an enhanced for loop, calling the method from exercise 3 in the body.
   - If the value is even, cube it, then print it to console.
   - If the value is odd, square it, then print it to console.