Chapter 5

Loops

A loop is a programming tool that allows developers to repeat the same block of code until some condition is met. We employ loops to easily scale programs, saving time and minimizing mistakes.

We'll go over three types of loops that you'll see everywhere:

- while loops
- for loops
- for-each loops

5.1 While Loop

The if-then statement is the most simple control flow we can write. It tests an expression for truth and executes some code based on it.

Example:

```
// Importing the Random library
import java.util.Random;

class LuckyFive {

  public static void main(String[] args) {
    // Creating a random number generator
    Random randomGenerator = new Random();
    // Generate a number between 1 and 6
    int dieRoll = randomGenerator.nextInt(6) + 1;
    // Repeat while roll isn't 5
    while (dieRoll != 5) {
           System.out.println(dieRoll);
      dieRoll = randomGenerator.nextInt(6) + 1;
       }
  }
}
```

When looping through code, it's common to use a counter variable. A counter (also known as an iterator) is a variable used in the conditional logic of the loop and (usually) incremented in value during each iteration through the code.

Example:

```
class Coffee {

  public static void main(String[] args) {
    // initialize cupsOfCoffee
    int cupsOfCoffee = 1;
    // add while loop with counter
    while (cupsOfCoffee <= 100){
      System.out.println("Free drinks cup of coffee #"+cupsOfCoffee);
      cupsOfCoffee++;
    }
  }
}
```

4.2 For Loop

A for loop brings together the following steps in a single line, separated by semicolons:

- Initializing a counter variable.
- Defining the looping condition.
- Incrementing the counter.

Example:

```
class Coffee {

  public static void main(String[] args) {

    for(int cupsOfCoffee=1;cupsOfCoffee <=100;cupsOfCoffee++) {

      System.out.println("Free drinks cup of coffee #" +
cupsOfCoffee);

    }

  }

}
```

Example of using for loop to loop through array elements:

```
import java.util.ArrayList;

class CalculateTotal {

  public static void main(String[] args) {

    ArrayList<Double> expenses = new ArrayList<Double>();
    expenses.add(74.46);
    expenses.add(63.99);
    expenses.add(10.57);
    expenses.add(81.37);
    double total = 0;
    // Iterate over expenses
    for (int i = 0; i < expenses.size(); i++) {

      total += expenses.get(i);
    }
    System.out.println(total);

 }

}
```

4.3 For-each Loop

For-each loops allow you to directly loop through each item in a list of items (like an array or ArrayList) and perform some action with each item. The syntax looks like this:

```
for (String inventoryItem : inventoryItems) {
        System.out.println(inventoryItem);
}
```

We can read the : as "in" like this: for each inventoryItem (which should be a String) in inventoryItems, print inventoryItem.

Example:

```
import java.util.ArrayList;

class MostExpensive {

  public static void main(String[] args) {

    ArrayList<Double> expenses = new ArrayList<Double>();
    expenses.add(74.46);
    expenses.add(63.99);
    expenses.add(10.57);
    expenses.add(81.37);

    double mostExpensive = 0;

    // Iterate over expenses
    for(double expense : expenses){
     if (expense > mostExpensive)
     {mostExpensive = expense;}
    }

    System.out.println(mostExpensive);
  }
}
```

Note on ArrayList:

The ArrayList class is a resizable array, which can be found in the java.util package.

The difference between a built-in array and an ArrayList in Java, is that the size of an array cannot be modified (if you want to add or remove elements to/from an array, you have to create a new one). While elements can be added and removed from an ArrayList whenever you want.

Example:

```
import java.util.ArrayList;

public class MyCars {
```

```
  public static void main(String[] args) {
    ArrayList<String> cars = new ArrayList<String>();
    cars.add("Volvo");
    cars.add("BMW");
    cars.add("Ford");
    cars.add("Mazda");
    System.out.println(cars.get(0));
  }
}
```

To access an element in the ArrayList, use the get() method and refer to the index number. To remove an element, use the remove() method and refer to the index number. Example:

```
import java.util.ArrayList;
public class MyCars {
  public static void main(String[] args) {
    ArrayList<String> cars = new ArrayList<String>();
    cars.add("Volvo");
    cars.add("BMW");
    cars.add("Ford");
    cars.add("Mazda");
    cars.remove(0);
    System.out.println(cars);
  }
}
```

To remove all the elements in the ArrayList, use the clear() method, cars.clear().

What other things can you do with ArrayList?