while-loops

Learn the syntax of while-loops in Java.

WE'LL COVER THE FOLLOWING

while-loops in Java work exactly the same as in Python, Javascript, and C, although the syntax is different than that of Python.

The examples and exercises are worth working through anyway, to gain comfort with code organization in Java. Here is a simple example of using a while-loop to count in Java:

```java
class WhileExample {
  public static void main(String[] args) {
    int number = 0;
    while(number < 100) {
      System.out.println(number);
      number += 2;
    }
  }
}
```

Exercise: charge!

In Java, while-loops are rarely used for counting, since for-loops provide a briefer syntax. But while-loops are great for doing an action some unspecified number of times until a condition changes. In the code below, write a static method `charge` that causes the robot to drive forwards until it has gone either `maxDist` squares or until it hits a wall, whichever is sooner.

You may call the method `blocked()` on the robot; `blocked()` returns `true` if there is a wall in front of the robot, and false otherwise.

ChargingRobotDemo.java | Sample solution

```java
class ChargingRobotDemo {

  public static void charge(SimpleRobot r, int maxDist) {
    // you write this part
  }

  public static void main(String args[]) {
    SimpleRobot r = new SimpleRobot();
    r.forward();
    r.right();
    charge(r, 3);
  }

}
```

You probably noticed that the syntax for calling `charge()` on a robot is different than the syntax for calling `forward()`, since the robot is passed as the first parameter to `charge`, rather than appearing before the dot. This is because `charge` is a static method of the current class, while `forward` is a method of the `SimpleRobot` class. We'll see how to write non-static methods soon.

Translation exercise: factor

It's good practice to translate code from one language to another. Here is some Python code that computes the integer factors of a number. Write Java code to compute the factors in the second tab; while you are at it, put the code inside a nice `public static void` method `printFactors` that takes an integer parameter and prints the integer factors on the screen. You should be fine even if you do not know Python.

factor.py | PrintFactors.java | Sample solution

```python
number = 42

possible_factor = 1
while possible_factor <= number:
    if number % possible_factor == 0:
        print( str(possible_factor) + " is a factor of " + str(number) + "." )
    possible_factor = possible_factor + 1

print( "And that's all the factors of " + str(number) + "." )
```

It would be even nicer to have a method that returns a list of factors, rather than printing the numbers out on the screen. One way to return the list of numbers would be to create an array; unfortunately, since we don’t know how many factors there will be ahead of time, we’d either have to create an array that was too large, or first count the factors. Later, we’ll see how to use an `ArrayList` object that will serve as an extensible list.