# While loop

"While" loops are used to repeatedly execute some code while a certain condition evaluates to `true`. The most commonly used form of this loop is:

```
while( <continue condition> ) {
    <statements>
}
```

The code fragment below tests if a `shape` is contained in a given `group` either directly or in any of its subgroups. The function `getGroup()` of `Shape` class returns the group, which is the immediate container of the shape. For a top-level shape (a shape that is not contained in any group) it returns `null`. In this loop we start with the immediate container of the `shape` and move one level up in each iteration until we either find the `group` or reach the top level:

```
ShapeGroup container = shape.getGroup(); //start with the immediate container of shape
while( container != null ) { //if container exists
    if( container == group ) //test if it equals the group we are looking for
        return true; //if yes, the shape is contained in the group
    container = container.getGroup(); //otherwise move one level up
}
return false; //if we are here, the shape is definitely not contained in the group
```

The condition in the first form of "while" loop is tested before each iteration; if it initially is false, nothing will be executed. There is also a second form of while loop – `do…while`:

```
do {
    <statements>
} while( <continue condition> );
```

The difference between `do…while` loop and `while` loop is that `do…while` evaluates its condition *after* the iteration, therefore the loop body is executed at least once. This makes sense, for example, if the condition depends on the variables prepared during the iteration.

Consider the following example. The local area map is a square 1000 by 1000 pixels. The city bounds on the map are marked with a closed polyline `citybounds`. We need to find a random point within the city bounds. As long as the form of the polyline can be arbitrary we will use Monte Carlo method: we will be generating random points in the entire area until a point happens to be inside the city. The `do…while` loop can be naturally used here:

```
//declare two variables
double x;
double y;
do {
    //pick a random point within the entire area
    x = uniform( 0, 1000 );
    y = uniform( 0, 1000 );
} while( ! citybounds.contains( x, y ) ); //if the point is not inside the bounds, try again
```