

A photograph of four students in a library setting. A young man in a grey t-shirt is smiling and looking at a laptop. A young woman with glasses is looking at the laptop. Another young woman is looking at a book. A fourth student is partially visible on the right. The background is filled with bookshelves. The image has a semi-transparent blue overlay on the left side and a semi-transparent red overlay on the right side.

Controlling Program Flow

'break' and 'continue' statements

Nested Loops

- Loops can be nested.

```
String[][] directions =  
    { {"North", "South"}, {"East", "West"} };  
// North    South  
// East     West  
for(String[] row:directions){// enhanced-for  
    for(int i=0; i<row.length; i++){// traditional for  
        System.out.print(row[i] + "\t");  
    }  
    System.out.println();  
}
```


break and *continue*

- The *break* and *continue* keywords are used to either exit the entire loop (*break*) or just skip to the next iteration (*continue*).
- *continue* statements must be inside a loop, otherwise you will get a compiler error.
- *break* statements must be used inside either a loop or a *switch* statement.
- If loops are nested, *break* and *continue* refer to the current loop (the one in which the *break* or *continue* statement exists).



labelled *break* and *continue*

- The labelled varieties are only needed in nested loops.
- They are used to indicate which of the nested loops you want to exit from or which of the nested loops you want to skip to the next iteration.
- A label must be placed just before the loop being labelled and it consists of a valid identifier that ends in colon (:). The identifier is case sensitive and must directly precede the loop i.e. no code between the label and the loop.
- Labelled *break* and *continue* statements must be inside the loop that has the same label name, otherwise the code will not compile.

