Controlling Program Flow

'while' and 'do-while' statements

Loops: while

- Java loops come in three flavours: *while*, *do-while* and *for* (as of Java 5, the *for* loop has two variants).

- The *while* loop is good when you don't know how many times a block or statement should repeat, but you want to continue looping as long as some condition is *true*.

Loops: while

- The general syntax is:

```
while(booleanExprIsTrue){
  // do something
}
```

- {} required if the loop controls more than 1 statement.
- A *while* loop might never run i.e. it executes 0 or more times. This is because the boolean expression being tested may be initially *false* and thus the loop will not run at all.

Loops: while

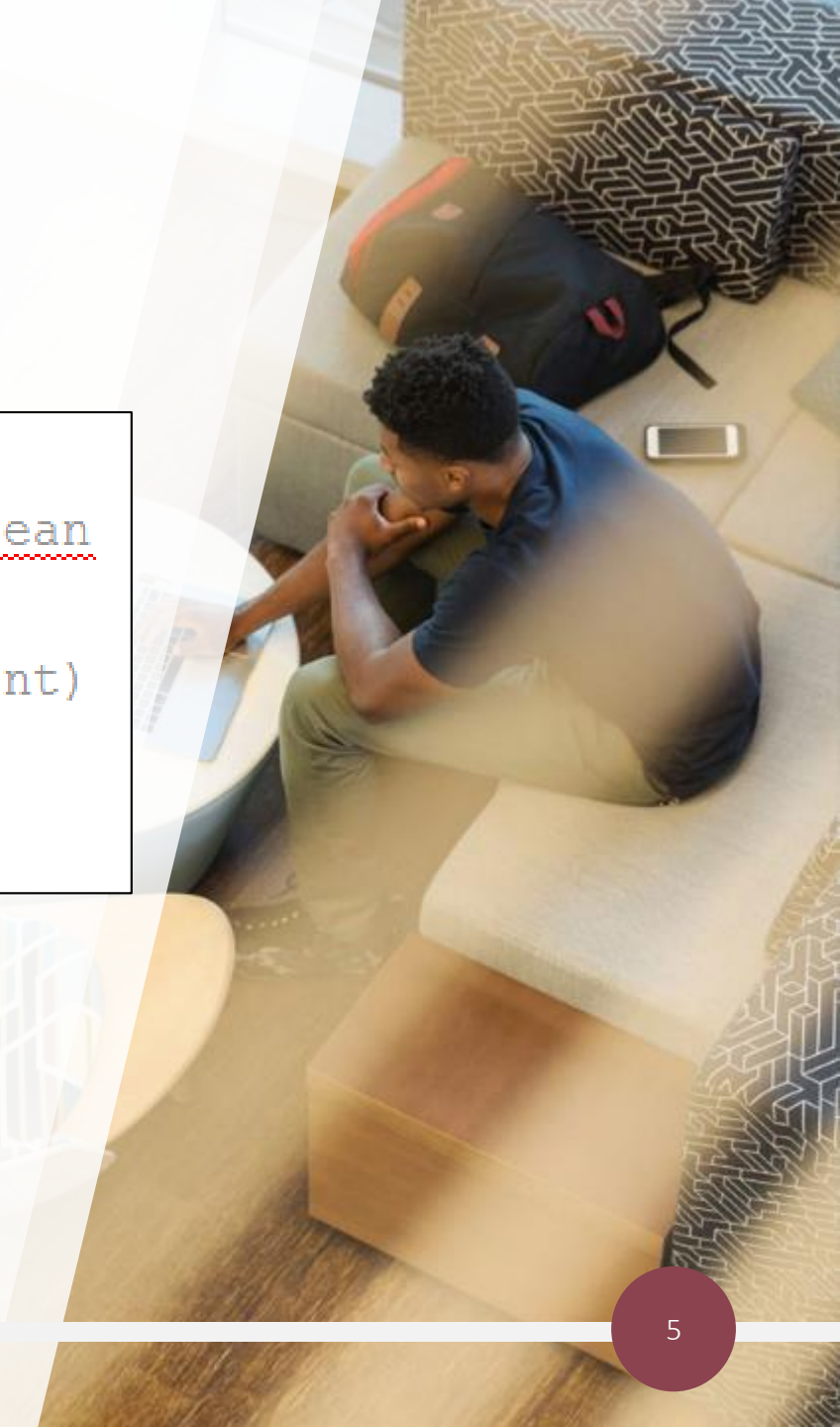
```
int x = 0;

while( x > 0)
    System.out.println("x > 0"); // not output

while(x < 3){
    System.out.println(x); // 0,1,2
    x++;
}
```

Loops: while

```
int x=1;
while(x){}          // int cannot be converted to boolean
while(x=5){}        // resolves to 5 (the value of
                    // the variable after the assignment)
while(x==5){}       // OK
while(true){}       // OK
```



Loops: do-while

- The *do-while* loop is similar to the *while* loop except that the expression is not evaluated until after the *do* loop's code is executed.

```
do{
    // something
} while(booleanExprIsTrue); // Note the ;
```

- A *do-while* loop is guaranteed to execute at least once i.e. it runs 1 or more times. This is because the boolean expression being tested is after the code block for the loop.

Loops: do-while

```
int x = 3;

do
    System.out.println(x);   // 3
while( x > 99);

do{
    System.out.println(x);    // 3, 2, 1
    --x;
}while( x > 0);

do{

}while(x > 0) // ; missing
```