Loops

Loops allow a program to repeat a statement, or a block of statements. In Java there are three basic loop types:

while
do-while
for

Loops can be executed using enumerators (not covered in this class).
while & do-while loops

```c
while ( condition )
{
    statement(s)
}

do
{
    statement(s)
} while ( condition );
```

Execute the statement(s) as long as the `condition` is true.

**Note:** semicolon after “while (condition)” in do-while

**Note:** no semicolon after “do”

**Note:** `condition` cannot be empty
The `break` statement terminates the loop

```java
while ( condition )
{
    statement(s)
    if ( new condition )
        break; // Terminate
    statement(s)
}
```

Infinite loop:

```java
while ( true )
{
    statement(s)
}
```

Empty loop (wait for event):

```java
while ( condition ) ;
```

can be re-written with infinite loop and break:

```java
while ( true )
    if ( condition )
        break;
```
**for loop**

```c
for( initialization; condition; increment )
{
    statement(s)
}
```

All parts are optional (careful)

*Initialization*: executed before start of the loop

*Increment*: executed at the end of the loop

*Condition*: checked after *initialization* and after each *increment*. Loop terminates if *condition* fails.
The classic for loop:

```c
for ( index = 0; index < limit; ++index )
{
    statement(s)
}
```

The `break` statement terminates the loop

```c
for ( index = 0; index < limit; ++index )
{
    statement(s)
    if ( new condition )
        break;            // Terminate
    statement(s)
}
```

Infinite loop:

```c
for ( ; true; )
{
    statement(s)
}
```

Empty loop (wait for event):

```c
for ( ; condition; ) ;
```
Choosing a Loop Type

Any loop can be implemented using either a `for` or a `while` or a `do-while`. The choice of loop type depends on what is easiest to implement and to read.

As a general rule, if the number of steps are known in advance and neither the end condition nor the increment are changed inside the body of the loop, you use `for`, otherwise use `while` or `do-while`.