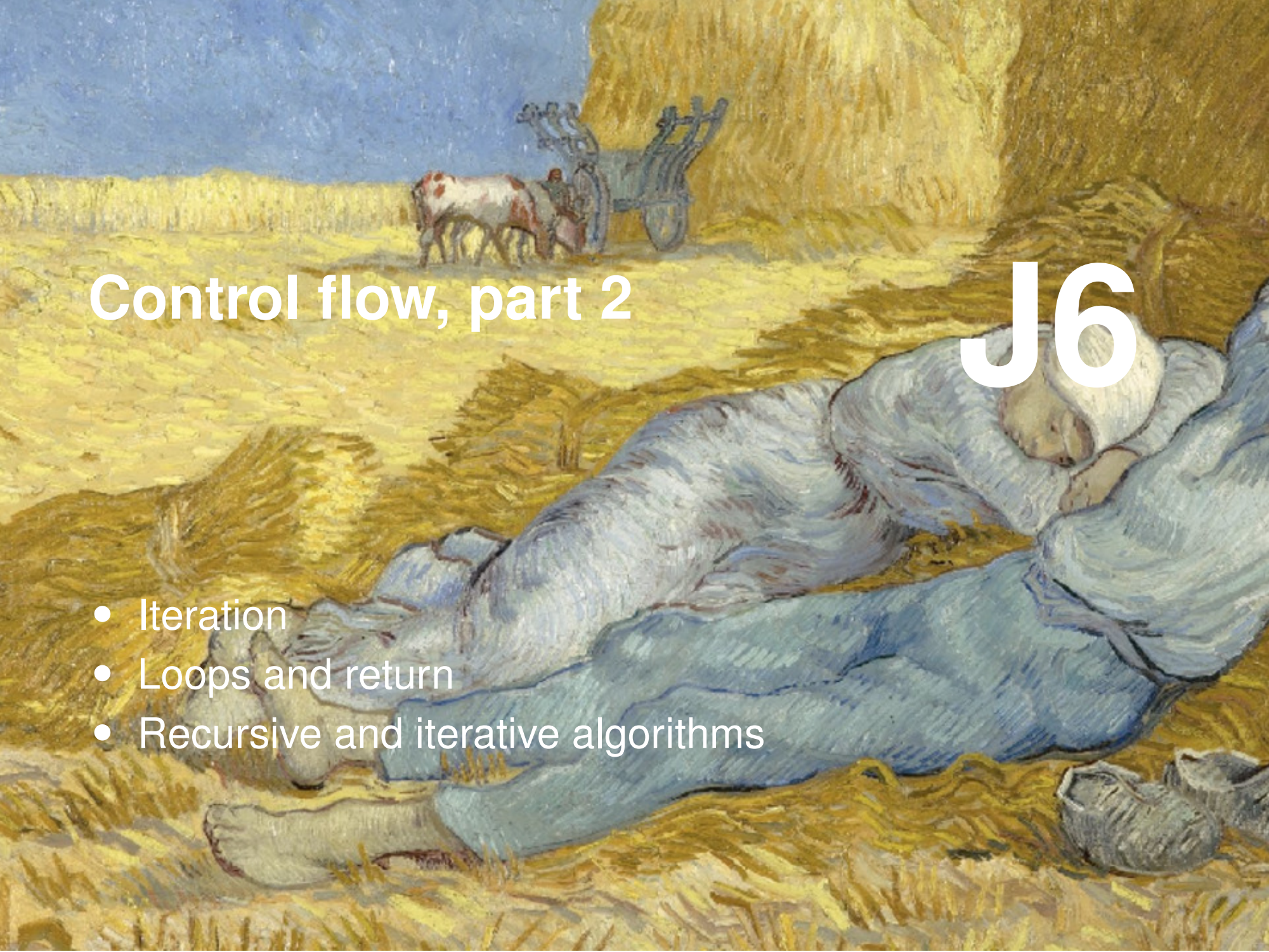
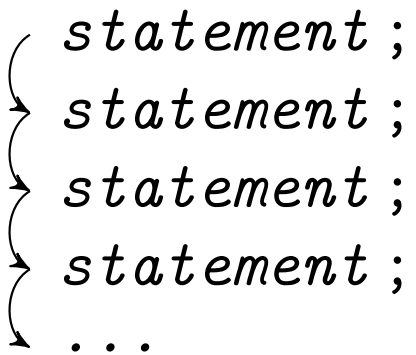


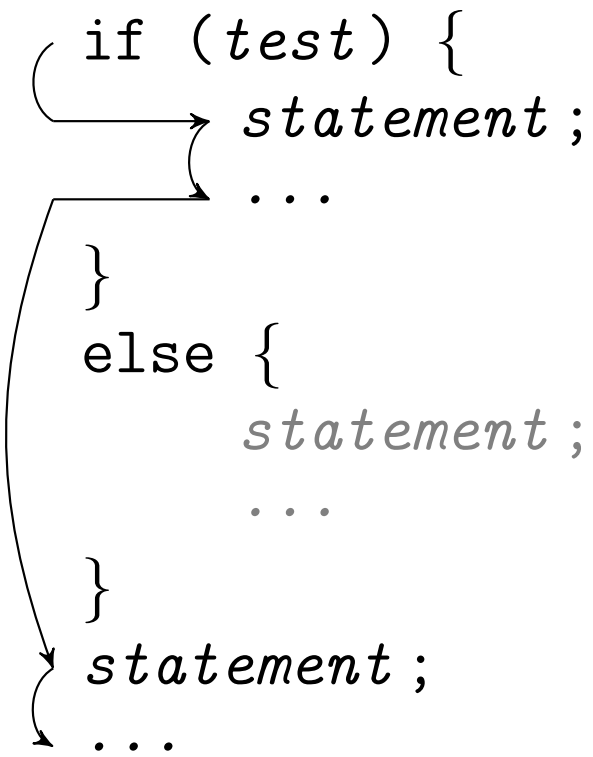
Control flow, part 2

J6

- Iteration
- Loops and return
- Recursive and iterative algorithms

Control flow





- Structured imperative programming: sequence, branching and *iteration*.

Iteration

- Iteration repeats a block of statements.
- A test is evaluated before each iteration, and the block executed (again) if it is true.
- When the loop finishes, execution continues at the next following statement.

The `while` statement

```
while (condition)
  <block>
```

- The condition is an expression of type `boolean`.
- A block is either a statement (ending with `;`) or a sequence of statements in braces (`{ }`).
- A `while` statement can appear inside a block.

[Coding]

`return` from a loop

- The `return` statement, in a function, ends the execution of a function call immediately (returning to where the function was called).
- Hence, a `return` within a loop (that is in a function) will also leave the loop, *even if the loop condition remains true*.
- (This can be useful as well as confusing.)

```
static int gcd(int a, int b) {
  int d = 1;
  int c = 1;
  while (d <= a && d <= b) {
    if (a % d == 0 && b % d == 0)
      c = d;
    d += 1;
  }
  return c;
}
```

```
static int gcd(int a, int b) {
  int c = (a > b ? b : a);
  while (c > 1) {
    if (a % c != 0 || b % c != 0)
      return c;
    c -= 1;
  }
  return 1;
}
```

Recursive & iterative algorithms

```
static int f(int n) {
  if (n == 1)
    return 1;
  else
    return n * f(n-1);
}
```

```
static int f(int n) {
  int r = 1;
  while (n > 1) {
    r = r * n;
    n = n - 1;
  }
  return r;
}
```

- We can always replace iteration with recursion.
- We can transform a recursive algorithm into an iterative, but this may require extra memory (*dynamic programming*).

The `for` statement

```
for (loop var init; loop cond; update)
  <block>
```

- The loop variable is initalised before the loop starts, and updated at the end of each iteration; the loop ends when the loop condition is false.
- For example,

```
for (int i = 0; i < 5; i = i + 1) {
  System.out.println(i)
}
```

- Shorthand syntax for looping over container data structures (later in the course).