extern

extern with Methods

It turns out that when a function is declared or defined, the `extern` keyword is implicitly assumed. For example, when we write:

```
int foo(int arg1, char arg2)
```

The compiler treats it as:

```
extern int foo(int arg1, char arg2)
```

Since the `extern` keyword extends the function's visibility to the whole program, provided those files contain a declaration of the function. With the declaration of the function in place, the compiler knows the definition of the function exists somewhere else and goes ahead and compiles the file.

Now lets say you define a method, if you don't use the keyword `extern` the compiler will not allow external use of that method unless explicitly stated.

extern with Variables

To declare a variable without defining it you would write:

```
extern int var;
```

Here, an integer type variable called var has been declared, but not defined so no memory was allocated for it. To define a variable you would write:

```
int var;
```

In this line, an integer `var` has been both declared and defined. However, with variables C does not implicitly define it as an `extern`.