UCB CS61B

Lecture 1

```java
public class Helloworld {
    public static void main(String[] args){
        System.out.println("Hello, World!");
    }
}
```

- All code in java must be part of a `class`.
- Delimit the begging and ends of code segments with `{` and `}`
- All statements must end with a semicolon `;`
- For code to run, we put it inside of a `public static void main(String[] args)` .

```java
public class HelloNumber {
    public static void main(String[] args) {
        int x = 0;
        while (x < 10) {
            System.out.println(x);
            x = x + 1;
        }
    }
}
```

- java requires that variables `be declared` before they are used. like `int x = 0;`
- java variables have a specific type, that type can never change, `static type`.
- Types are verified before the code runs, `Static type checking`.

Functions

```java
public class LargerDemo {
    public static int larger(int x, int y) {
        if (x > y) {
            return x;
        }
        return y;
    }

    public static void main(String[] args) {
        System.out.println(larger(5, 10));
    }
}
```

- Functions must be part of a class. A function that is part of a class is called a `method`. Therefore, all functions in Jave are methods.
- To define a function, we use (for today) the `public static` .
- All paremeters of a function must have a `declared type`, and the function must `return a specific type`.
- functions in java only return one value.

> **why make a class file at all?**
> class file has been type checked. Distributed code is safer.
> class file are 'simpler' for machines to excute. Distributed code is faster.

> **Static Type**
> *The Goods*
>
> > Always what things are - easier to reason about code.
> > User facing application, less opportunity for users input to cause errors.
> >
> > > Many fewer bugs possible.
> >
> > Early error detection. You can fix it right when you write it.
>
> *The Bads*
>
> > Less flexibility.
> > Makes the language more verbose.
> > Tricky syntax. The syntax gets very weird for methods that need to deal with very specific types with certain properties.

Homework

```
public class NumberTotal {
    public static void main(String[] args) {
        int total = 25;
        for (int number = 1; number <= (total / 2); number++) {
            total = total - number;
            System.out.println(total + " " + number);
        }
    }
}
```

print:

```
24 1
22 2
19 3
15 4
10 5
```

- In Java's `for loop`, the update expression is executed after all the contents of the loop body have been executed but before checking the next loop condition.