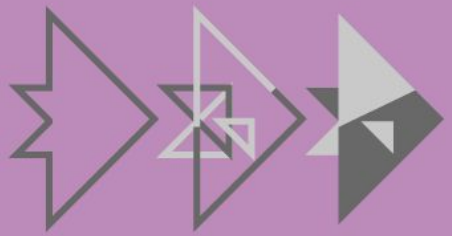
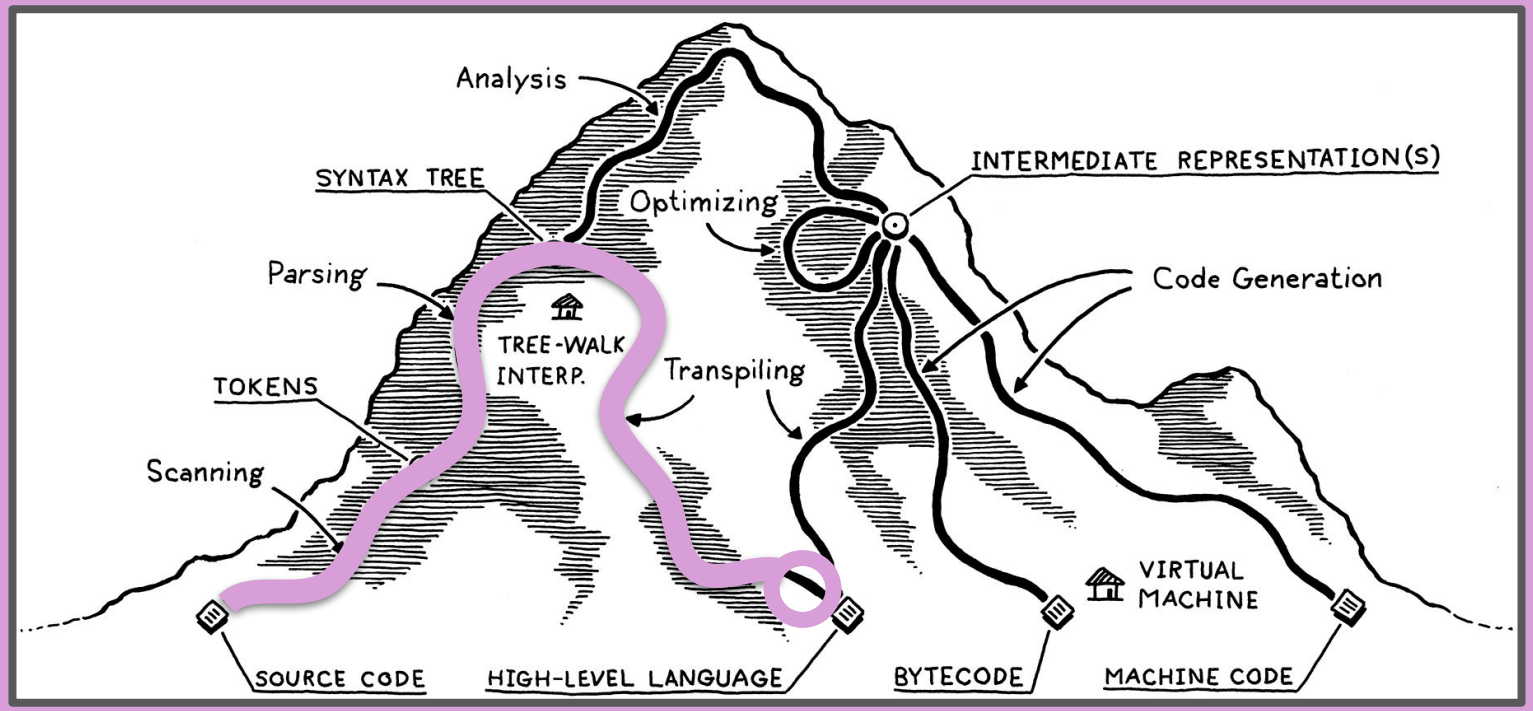
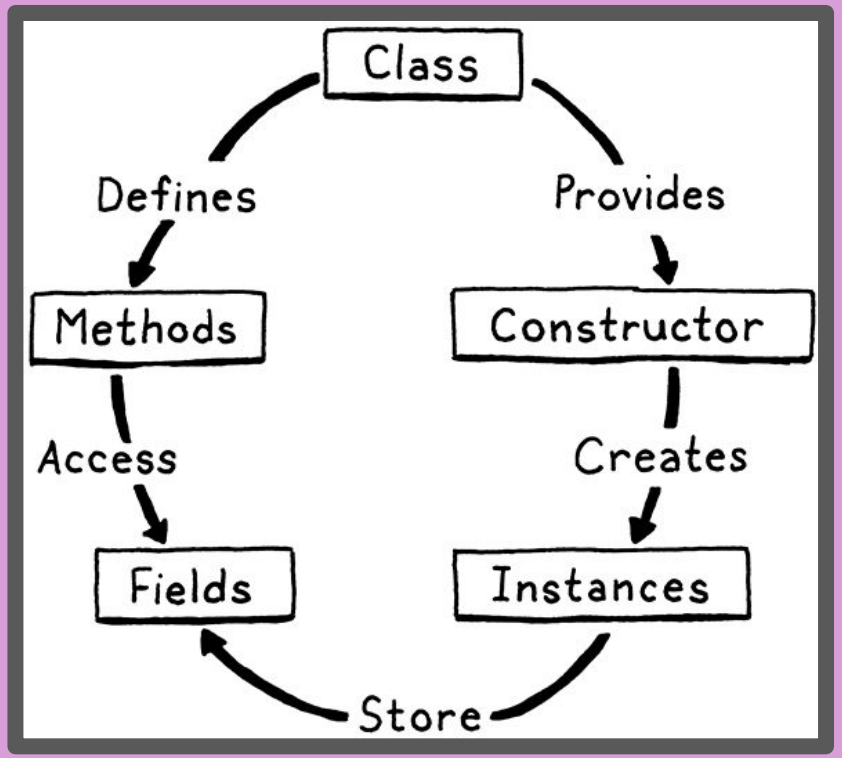


CMPSC 201: PROGRAMMING LANGUAGES


Analysis
SYNTAX TREE
Optimizing
INTERMEDIATE REPRESENTATION(S)
Parsing
Code Generation
TREE-WALK
INTERP.
TOKENS
Transpiling
Scanning
VIRTUAL
MACHINE
SOURCE CODE
HIGH-LEVEL LANGUAGE
BYTECODE
MACHINE CODE

Being class-y



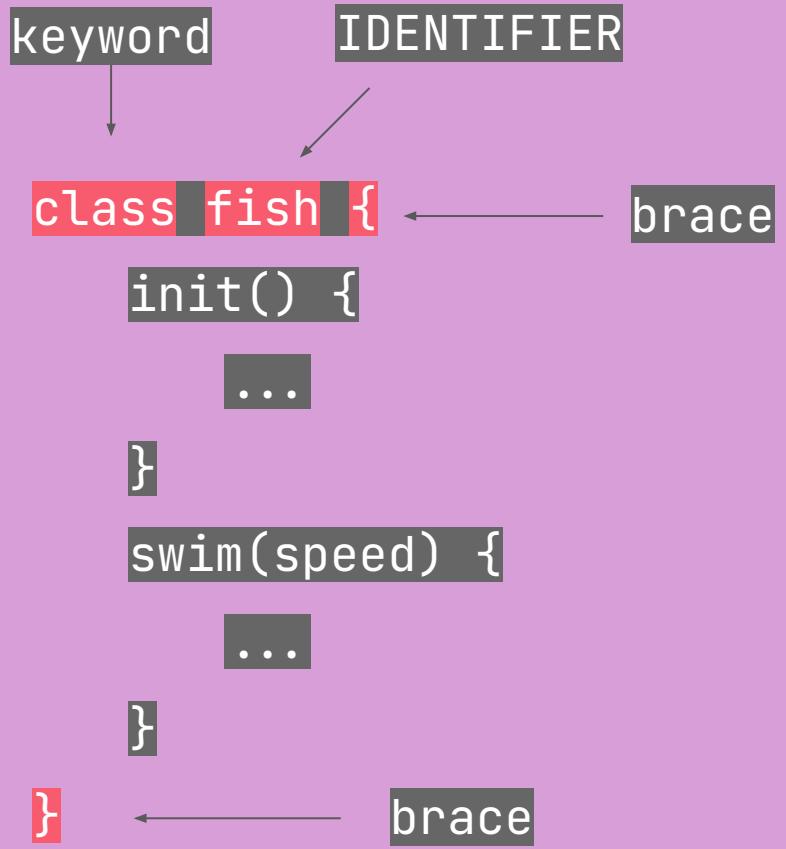
Class: a construct that bundles data and code that operates on it.

Made up of:

- a constructor
- fields to store data
- State-changing methods shared by instances of the class

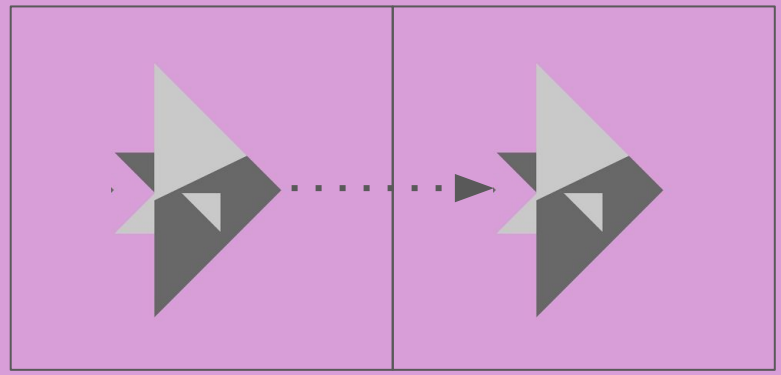
Being class-y

```
classDecl → "class" IDENTIFIER "{"
... "}" ;

function* → IDENTIFIER "("
parameters? ")" block ;

parameters → IDENTIFIER ( ","
IDENTIFIER )* ;
```



Methods

A procedure bound to a class, creating an interface enabling interaction with a given class' instance data.



```
fun swim(posX, speed) {
```

```
...
swim(speed) {
```

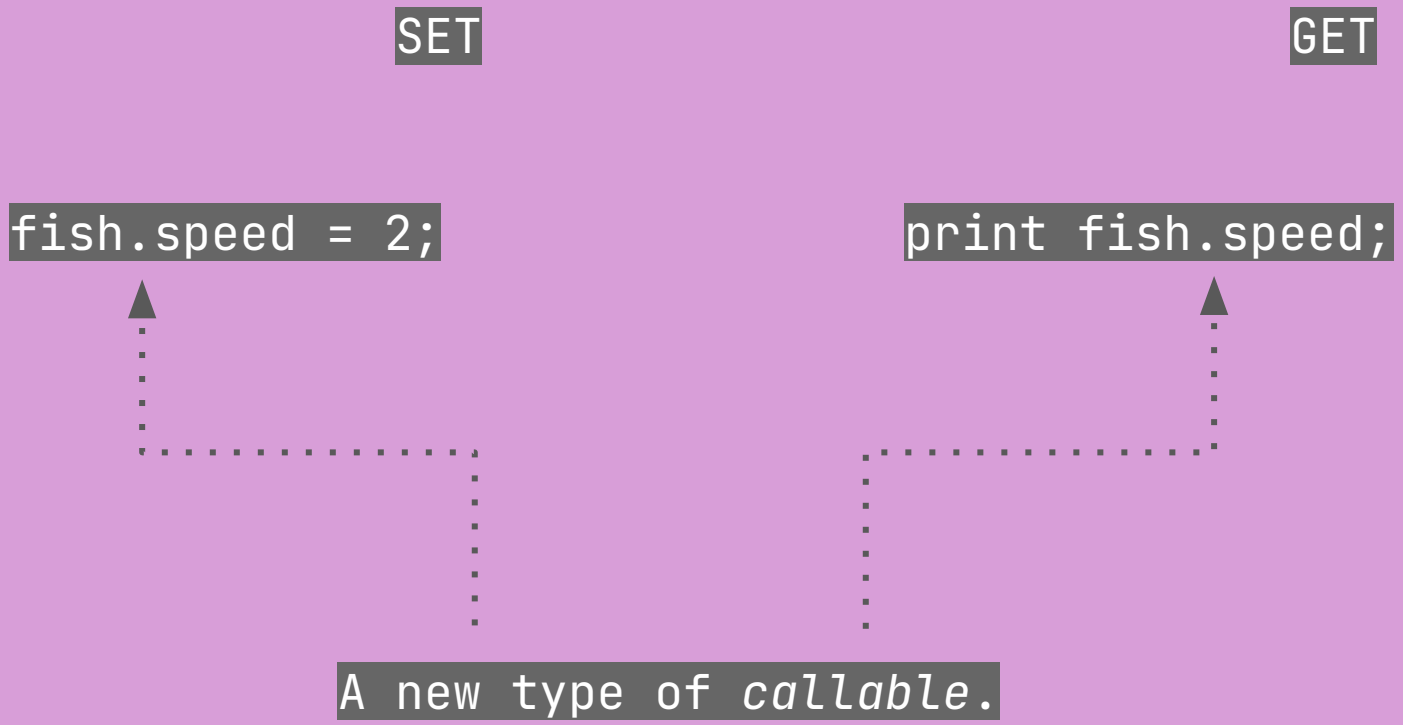
For sake of argument, let's just say that methods are a *special kind of function*.

```
}
```

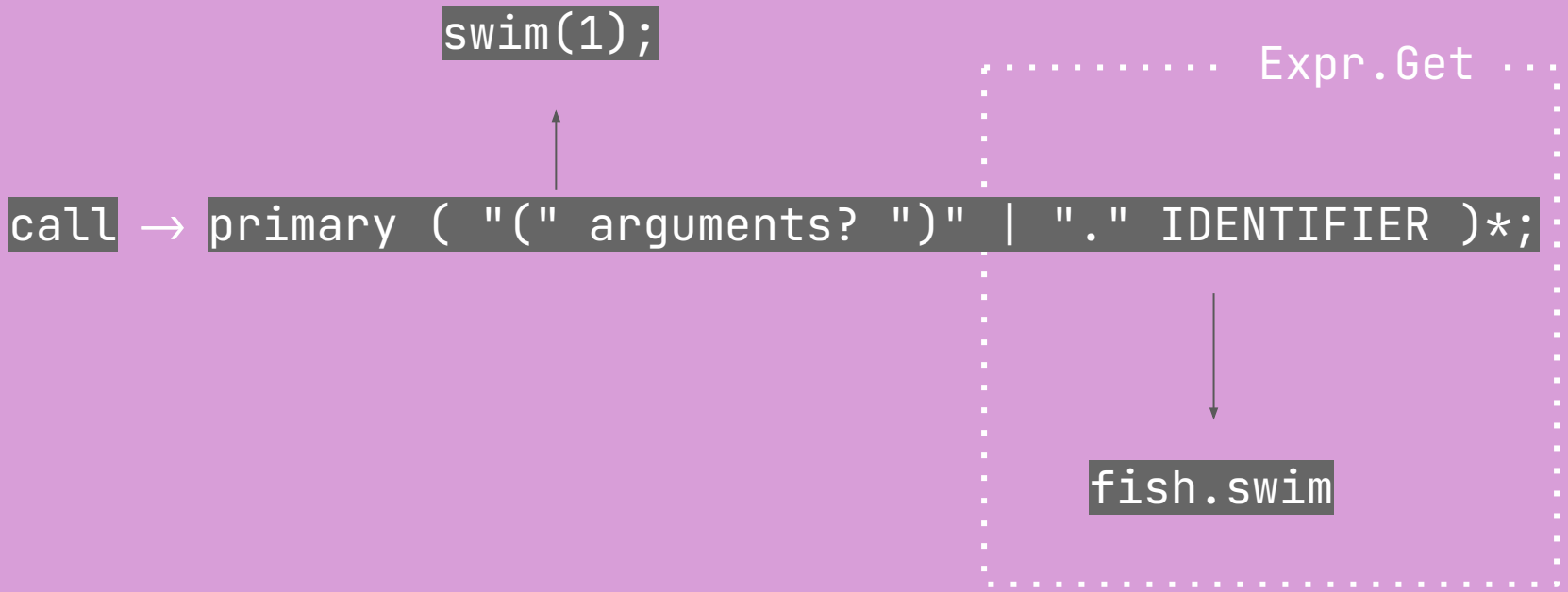
```
swim(1);
```

```
fish.swim(1);
```

A detour into SET and GET



A detour into SET and GET



A detour into SET and GET

```
assignment → (call "." ) ? IDENTIFIER "=" assignment | logic_or;

call → primary ( "(" arguments? ")" | "." IDENTIFIER )*;

fish.type.jump.speed(1)
```

Continuing with methods

```
class fish {
    init() {
        this.pos = 0;
    }
    swim(speed) {
        this.pos = this.pos + speed;
    }
}
```

When we interpret the class, we gather all of its methods in a list of functions named specially.

What's this?

```
class fish {
    init() {
        this.pos = 0;
    }
    swim(speed) {
        this.pos = this.pos + speed;
    }
}
```

Like Python's `self`, `this` binds to a given *instance*, of an object. No two fish are alike!

What's this?

```
class fish {
  init(){
    this.pos = 0;
    this.speed = 0;
  }
  swim() {
    this.pos = this.pos
        + this.speed;
  }
}
```

```
bass = fish();
bass.speed = 1;
bass.swim();
```

>> pos is 1

```
trout = fish();
trout.speed = 2;
trout.swim();
```

>> pos is 2;