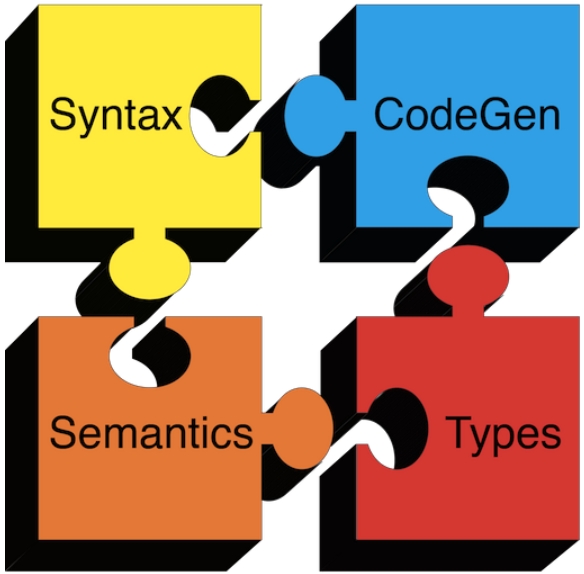

Compilers

Static vs. Dynamic Typing

Static vs. Dynamic

- Static type systems detect common errors
- But some correct programs are disallowed
  - Some argue for dynamic type checking instead
  - Others want more expressive static type checking
- But more expressive type systems are more complex

Static vs. Dynamic

- The dynamic type of an object is the class C that is used in the "new C" expression that created it
  - A run-time notion
  - Even languages that are not statically typed have the notion of dynamic type

- The static type of an expression captures all dynamic types the expression could have
  - A compile-time notion

Static vs. Dynamic

- Soundness theorem: for all expressions E

  dynamic_type(E) = static_type(E)

In all executions, E evaluates to values of the type inferred by the compiler.

Static vs. Dynamic

```
class A { … }
class B inherits A {…}
class Main {
   x:A ← new A;
   …
   x ← new B;
   …
}
```

Static vs. Dynamic

Choose the static/dynamic type pairs that are correct. For dynamic type, assume execution has halted at line 14.

	Var	Static Type	Dynamic Type
☐	w	Animal	Lion
☐	x	Animal	Pet
☐	y	Pet	Dog
☐	z	Pet	Pet

```
1   class Animal { … }
2   class Pet inherits Animal { … }
3   class Cat inherits Pet { … }
4   class Dog inherits Pet { … }
5   class Lion inherits Animal { … }
6   class Main {
7     w:Animal <- new Animal;
8     x:Animal <- new Pet;
9     y:Animal <- new Pet;
10    z:Pet <- new Pet;
11    w <- new Lion;
12    y <- new Dog;
13    z <- new Cat;
14 →  …
15  }
```

Static vs. Dynamic

Soundness theorem for the Cool type system:

$$\forall \text{ E.} \quad \text{dynamic\_type(E)} \leq \text{static\_type(E)}$$

- All operations that can be used on an object of type C can also be used on an object of type C' ≤ C
  - Such as fetching the value of an attribute
  - Or invoking a method on the object
- Subclasses only add attributes or methods
- Methods can be redefined but with same type!