Modular Programming

Nate Foster
Spring 2019

Today’s music: "Giorgio By Moroder" by Daft Punk
Moog modular synthesizer

Based in Trumansburg, NY, 1953-1971

*Game changing!* picked up by the Beatles, the Rolling Stones...
Review

Previously in 3110:
• how to build small programs

Today:
• language features for building large programs: structures, signatures, modules
Scale

- Staff solution to A1: 100 LoC
- OCaml: 200,000 LoC
- Unreal engine 3: 2,000,000 LoC
- Windows Vista: 50,000,000 LoC


...can’t be done by one person
...no individual programmer can understand all the details
...too complex to build with OCaml we’ve seen so far
Modularity

**Modular programming:** code comprises independent *modules*

- developed separately
- understand behavior of module in isolation
- reason locally, not globally
Java features for modularity

• **classes, packages:** organize identifiers (classes, methods, fields, etc.) into namespaces

• **interfaces:** describe related classes

• **public, protected, private:** control what is visible outside a namespace

• **subtyping, inheritance:** enables code reuse
OCaml features for modularity

• **structures**: organize identifiers (functions, values, etc.) into namespaces

• **signatures**: describe related modules

• **abstract types**: control what is visible outside a namespace

• **functors, includes**: enable code reuse

...the OCaml *module system*
STRUCTURES
Structures

- Collections of definitions
- Evaluated in order
- Structure value can be bound to module name
- Structure values are second class
Signatures

• Collections of declarations (and some definitions)
• Not evaluated; just type checked
• Signature type can be bound to module type name
Type checking

If you give a module a type...

```ocaml
module Mod : Sig = struct ... end
```

Then type checker ensures...

1. **Signature matching:** everything declared in `Sig` must be defined in `Mod` (OK to add new definitions to `Mod` that aren't declared in `Sig`)

2. **Encapsulation:** nothing other than what’s declared in `Sig` can be accessed from outside `Mod`
ABSTRACT TYPES
Exposure is bad

- Client code shouldn't need to know what the representation type is
- Rule of thumb: clients will exploit knowledge of representation if you let them
- Client code shouldn't get to know what the representation type is
COMPILATION UNITS
OCaml features for modularity

- **structures**: organize identifiers (functions, values, etc.) into namespaces
- **signatures**: describe related modules
- **abstract types**: control what is visible outside a namespace
- **functors, includes**: enable code reuse
Upcoming events

• [Now] Team #1 boot-up!
  • Complete survey announced on Discourse
  • Short written assignment due Sunday
• [Wed] A1 due
• [Wed] A2 out

This is game changing.

THIS IS 3110