Ross Pate
rpate@my.athens.edu
Assignment 07
00084234

Strategy Selection:

- **OOP (e.g., C++)**
    - Strategies are objects implementing a common interface.
    - Context stores a reference/pointer and invokes via runtime polymorphism (vtables).
    - Switching strategies = replacing the stored object reference.
    - Strength: supports stateful strategies and clear contracts.
    - Tradeoff: requires boilerplate (classes, inheritance).
- **FP (e.g., OCaml)**
- Strategies are first-class functions passed directly.
- Context = higher-order function applying the chosen strategy.
- Switching strategies = supplying a different function argument.
- Strength: lightweight, no dispatch overhead, easy composition.
- Tradeoff: state must be threaded explicitly or via closures.

Core distinction:
OOP selects behavior by choosing an object (identity + state), while FP selects behavior by choosing a function (computation only)**.**

Functions-as-Values vs Objects-as-Behaviors:

Aspect	**OOP**	**FP**
Unit of behavior	Object bundling state + methods	Function as pure computation
Advantages	Encapsulation, stateful strategies, discoverable via type hierarchy	Minimal ceremony, composable, inline definitions
Tradeoffs	Boilerplate, instantiation overhead	Harder to group helpers, explicit state handling

OCaml Modules and Functors and how they can serve similar roles to classes:

- **Modules** = group of related functions/types (like classes).
- **Module types** = abstract interfaces.
- **Functors** = parameterized modules (like generic classes/templates).
    - Provide compile-time specialization and type safety.
    - Enable reusable contexts with interchangeable implementations.
- **First-class modules** bridge both worlds: structured modules with runtime flexibility