Designing with Interfaces

OO A&D Principle

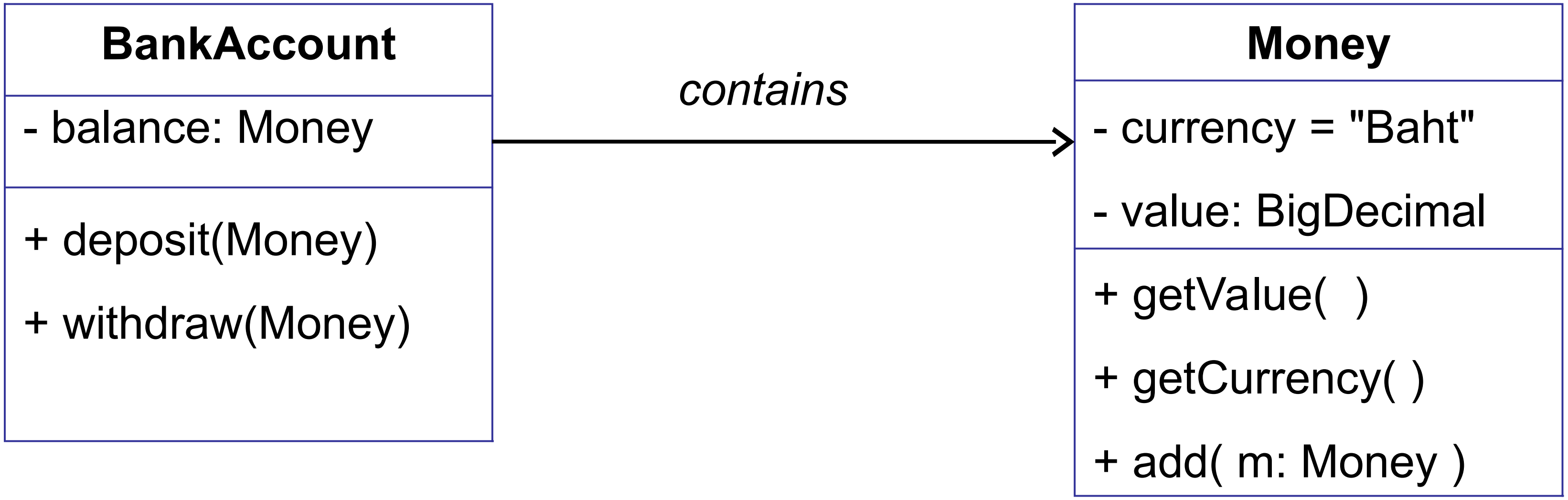
"*Program to an interface, not an implementation*"

meaning:

"Program to the *specification* (of an object's behavior), don't depend on its *implementation* (which may change)".

Designing with Interfaces (1)

1. Use interfaces to "protect" one class from another class whose implementation may change.
2. Reduces *coupling* between classes.
3. Define what *behavior* a class must provide.



BankAccount

- balance: Money

+ deposit(Money)

+ withdraw(Money)

contains

Money

- currency = "Baht"

- value: BigDecimal

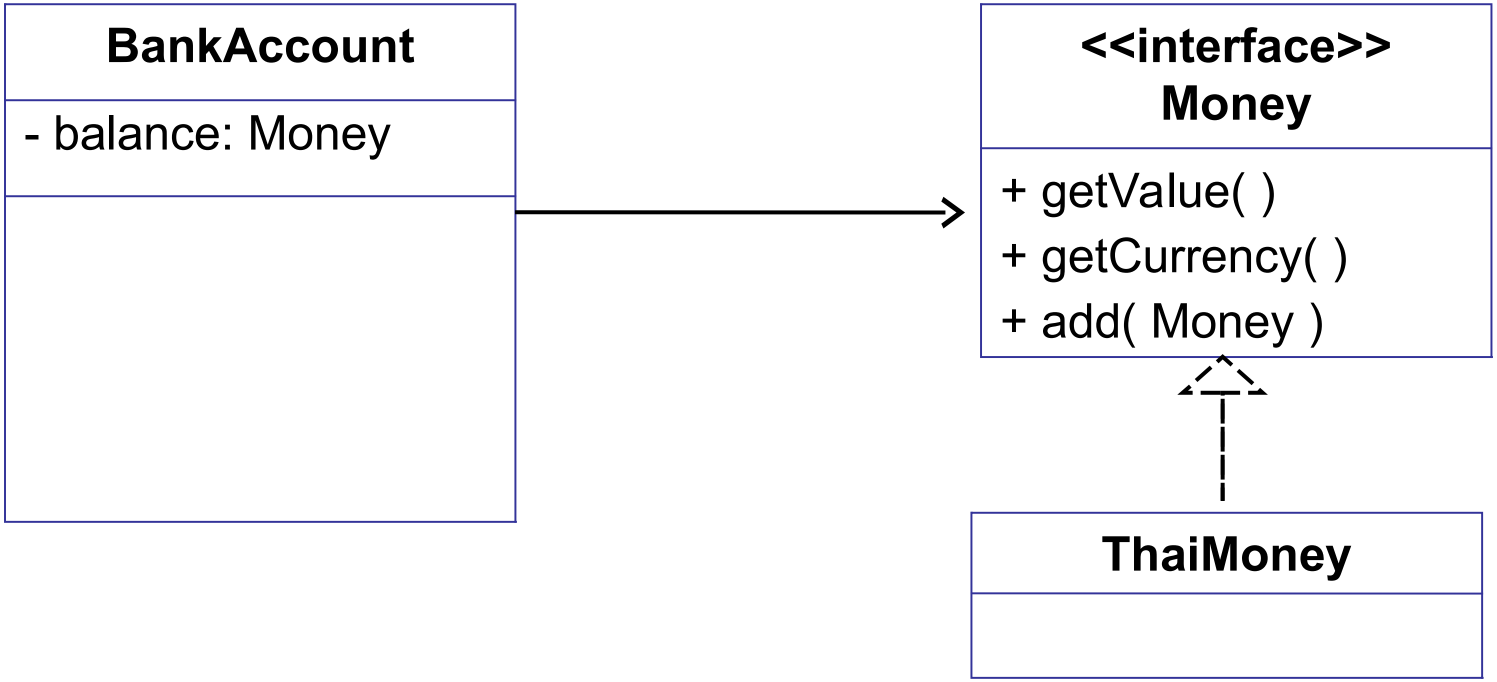
+ getValue( )

+ getCurrency( )

+ add(m: Money)

Designing with Interfaces (2)

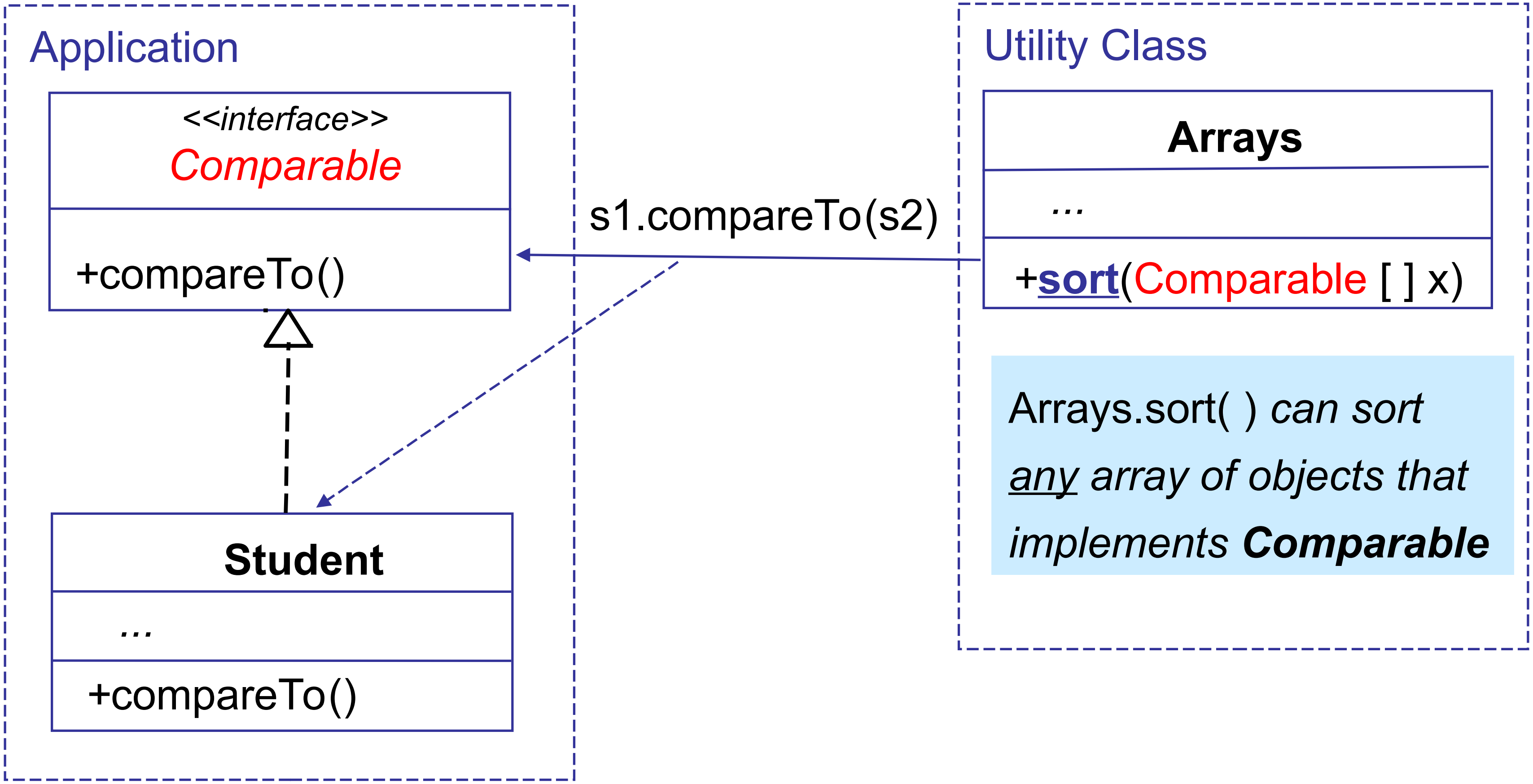
4. Create an interface for the required behavior.
5. Clients use the Interface type, not the actual type.
6. Providers *implement* the interface.



Designing with Interfaces

- Interface can be used to define required behavior of a *client.*

```
Arrays.sort( students )  // array of Student objects
```

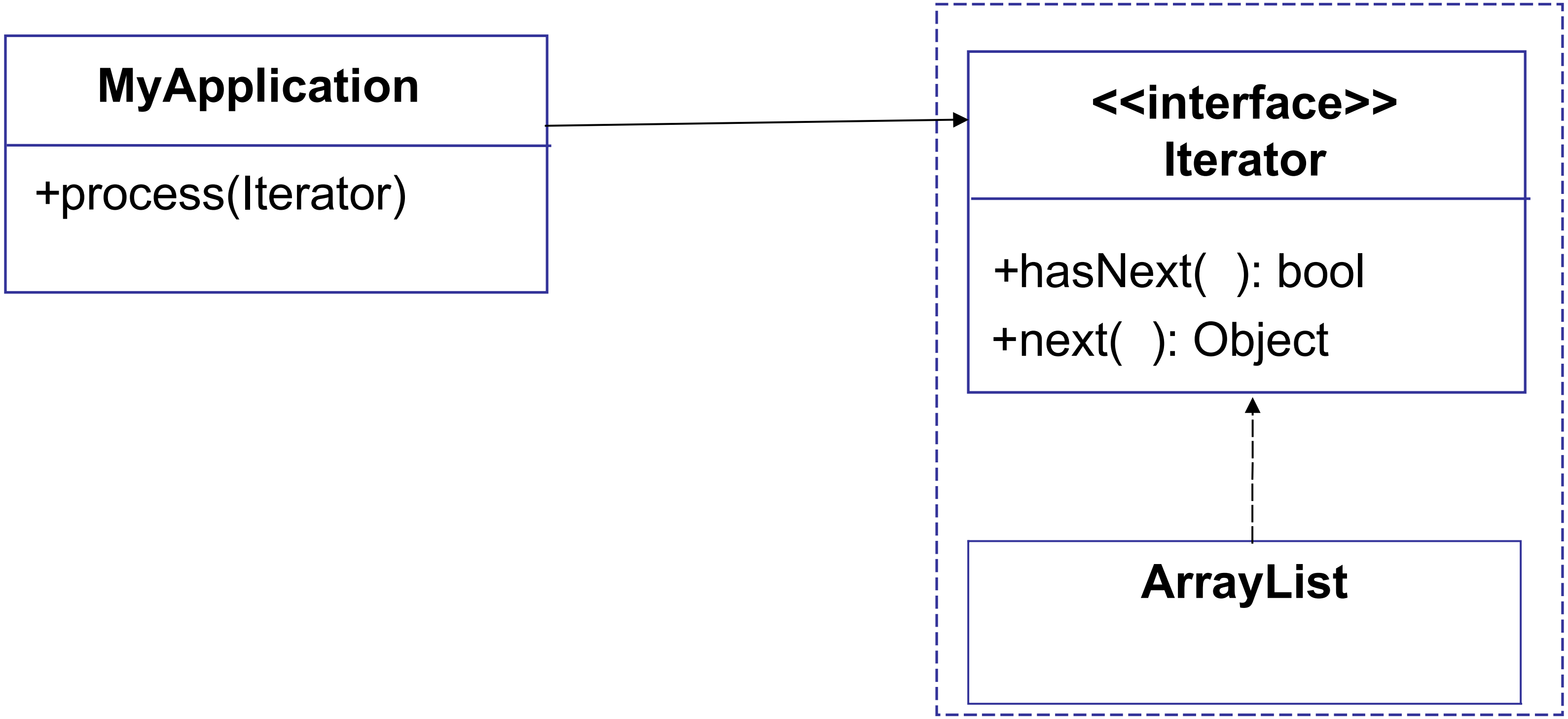


Arrays.sort( ) *can sort* *<u>any</u> array of objects that implements* ***Comparable***

Iterator Interface

Pattern: we want to visit every member of a collection, and we want this to work for *any kind of collection*.

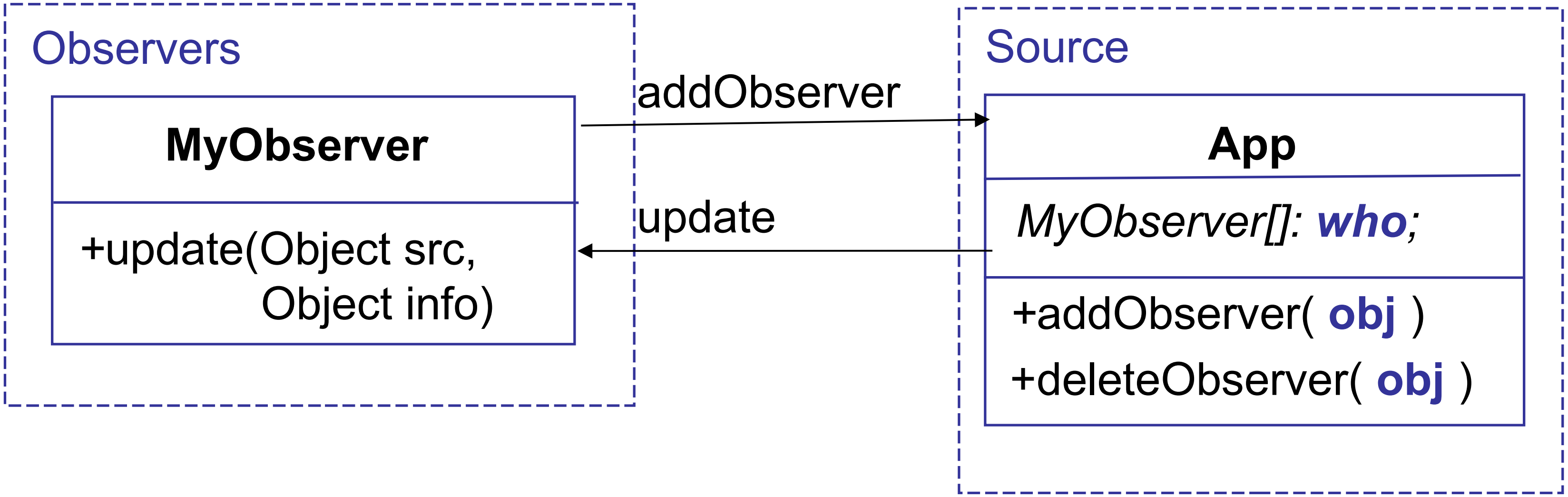
Solution: design an *interface* for the behavior we want. Require that all collections implement this interface.



Interface for the Observer Pattern

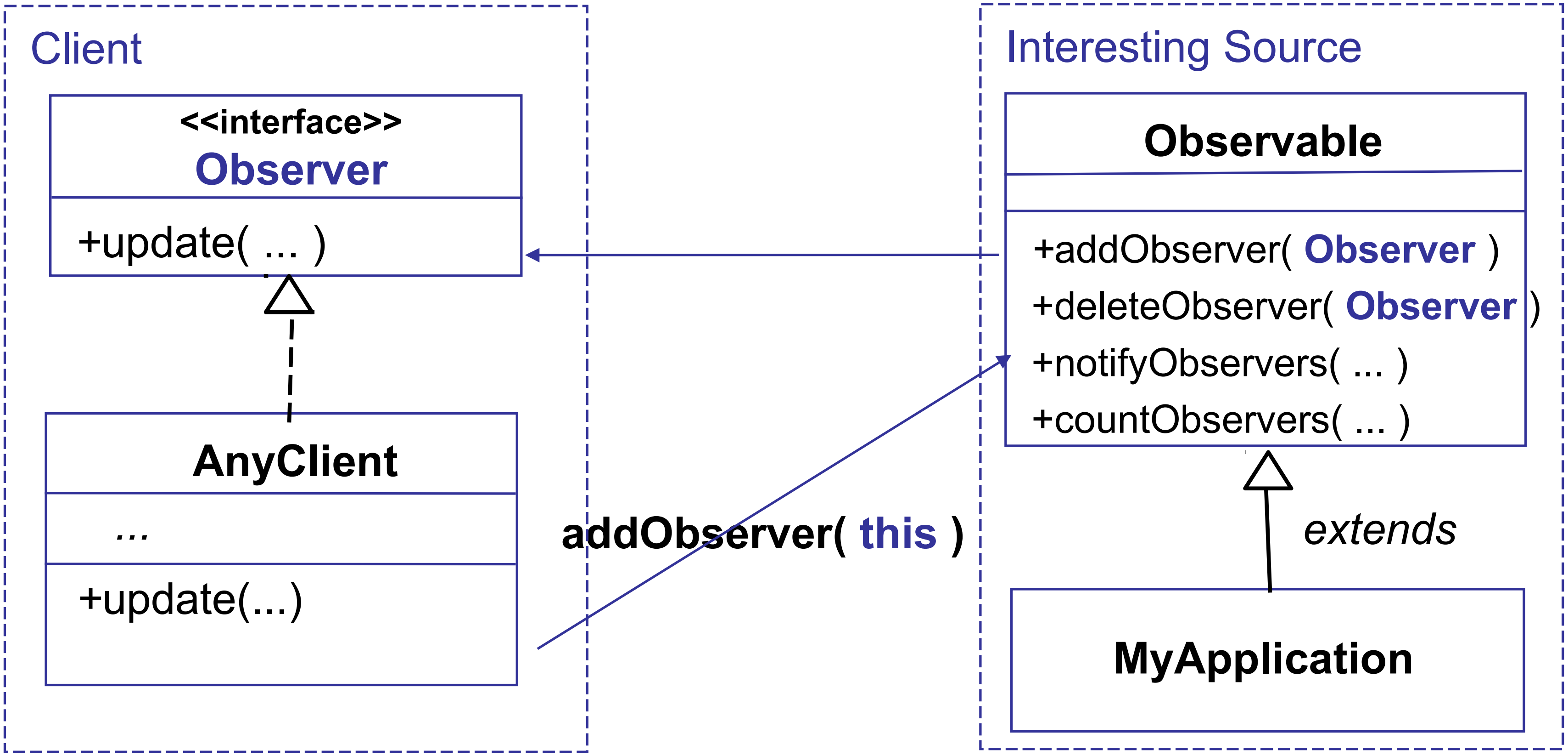
Pattern: one object is the source if "interesting" events. Other objects want to be notified when an interesting event occurs.

Solution: objects ***register*** themselves as Observers. Then the "interesting" event occurs, the source calls the Observers' **`update( )`** method.



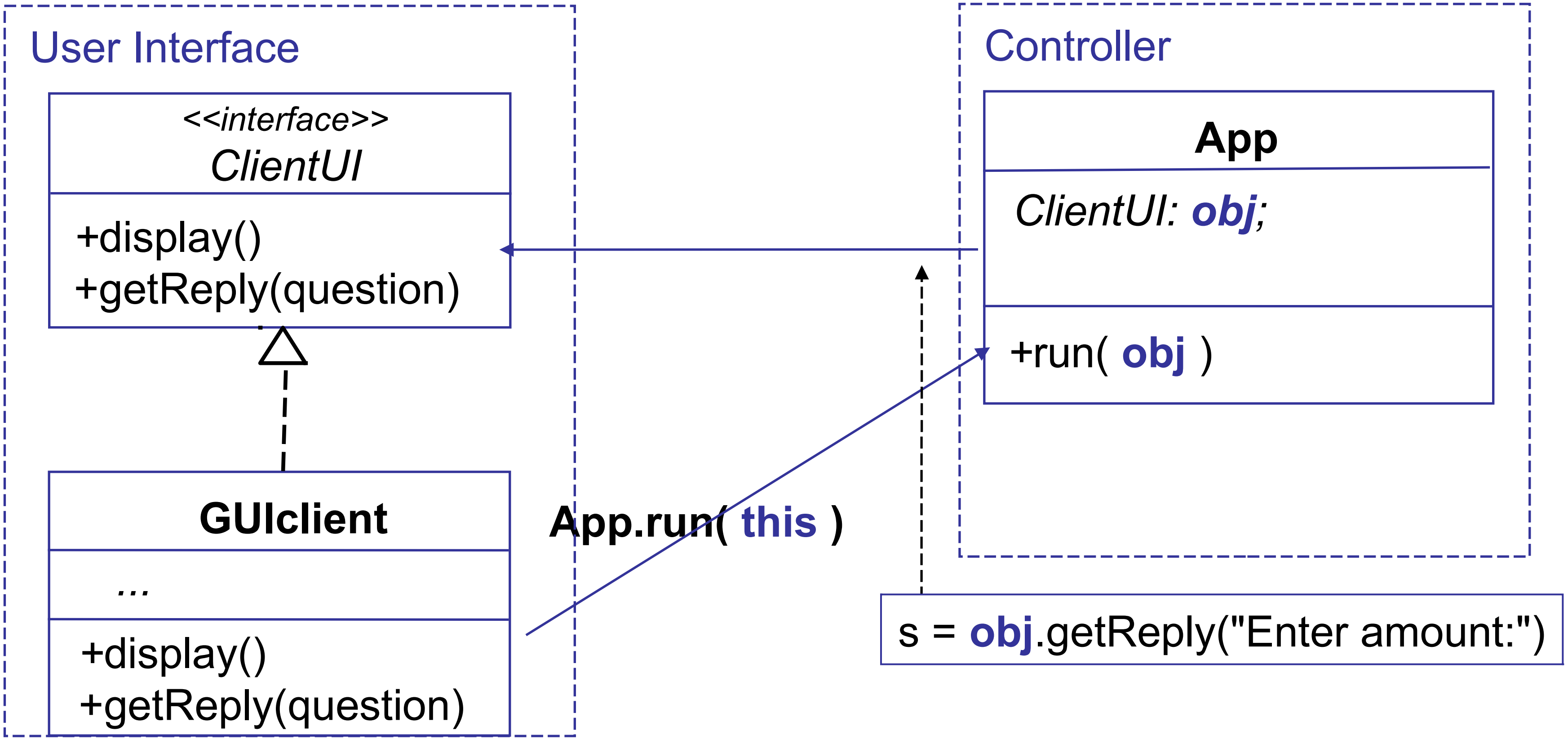
Interface for Observer Pattern

- The Java Observer interface specifies client behavior
- Observable abstract class provides the server side.



Interface for View-Controller Pattern

- Interfaces are used to separate an application's "user interface" from the "logic engine" of the application.
- Interface reduces dependency between classes.



Interfaces You Should Know

Interface	What it specifies
`Runnable`	run( ) method
`Comparable<T>`	compareTo(T other)
`Comparator<T>`	compare(T x, T y)
`Iterator<T>`	hasNext() and next()
	iterating over collections
`Iterable<T>`	iterator() - create an Iterator
	a way of creating iterators
`Cloneable`	safe to call clone()