Lecture 20

ECE 1145: Software Construction and Evolution

Framework Theory (CH 32)

Announcements

- Iteration 7: Blackbox Testing and Pattern Hunting due Nov. 14
  - Bonus: EtaCiv due Dec. 12
- Next week: Code Review 2
  - Code Swap due by start of class Wednesday Nov. 17
  - Report due Sun. Nov. 21
- Midterm survey on Canvas
- Schedule updates:

11	11/8	19	More Patterns	CH 26, 28, 29, 31	PI7: Blackbox Testing and Pattern Hunting
	11/10	20	Framework Theory	CH 32	
12	11/15	21	MiniDraw	CH 30	Code Review 2
	11/17		Time In-Class for Code Review		
13	11/22		No Class - Thanksgiving Recess		
	11/24		No Class - Thanksgiving Recess		
14	11/29	22	HotCiv GUI	CH 36.7	
	12/1	23	Exception Handling		
15	12/6	24	Final Review		PI8: Frameworks
	12/8		Time In-Class for Iteration 8		
16	12/13		No Class - Finals Week		Final
	12/15		No Class - Finals Week		

Questions for Today

How can the concepts we've learned so far be applied to develop customizable software?

What is a Framework?

A framework is:

- a) a reusable design of an application of subsystem; b) represented by a set of abstract classes and the way objects in these classes collaborate (Fraser et al. 1997)

What is a Framework?

A framework is:

- a) a reusable design of an application of subsystem; b) represented by a set of abstract classes and the way objects in these classes collaborate (Fraser et al. 1997)
- A set of classes that embodies an abstract design for solutions to a family of related problems (Johnson and Foote 1988)

What is a Framework?

A framework is:

- a) a reusable design of an application of subsystem; b) represented by a set of abstract classes and the way objects in these classes collaborate (Fraser et al. 1997)
- A set of classes that embodies an abstract design for solutions to a family of related problems (Johnson and Foote 1988)
- A set of cooperating classes that make up a reusable design for a specific class of software (Gamma et al. 1995)

What is a Framework?

A framework is:

- a) a reusable design of an application of subsystem; b) represented by a set of abstract classes and the way objects in these classes collaborate (Fraser et al. 1997)
- A set of classes that embodies an abstract design for solutions to a family of related problems (Johnson and Foote 1988)
- A set of cooperating classes that make up a reusable design for a specific class of software (Gamma et al. 1995)
- The skeleton of an application that can be customized by an application developer (Fayad et al. 1999)

What is a Framework?

A framework is:

- a) a reusable design of an application of subsystem; b) represented by a set of abstract classes and the way objects in these classes collaborate (Fraser et al. 1997)
- A set of classes that embodies an abstract design for solutions to a family of related problems (Johnson and Foote 1988)
- A set of cooperating classes that make up a reusable design for a specific class of software (Gamma et al. 1995)
- The skeleton of an application that can be customized by an application developer (Fayad et al. 1999)
- A definition of a high-level language with which applications within a domain are created through specialization (Pree 1999)

What is a Framework?

A framework is:

- a) a reusable design of an application of subsystem; b) represented by a set of abstract classes and the way objects in these classes collaborate (Fraser et al. 1997)
- A set of classes that embodies an abstract design for solutions to a family of related problems (Johnson and Foote 1988)
- A set of cooperating classes that make up a reusable design for a specific class of software (Gamma et al. 1995)
- The skeleton of an application that can be customized by an application developer (Fayad et al. 1999)
- A definition of a high-level language with which applications within a domain are created through specialization (Pree 1999)
- An architectural pattern that provides an extensible template for applications within a domain (Booch et al. 1999)

What is a Framework?

A framework is:

- a) a **reusable design** of an application of subsystem; b) represented by a set of abstract classes and the way objects in these classes collaborate (Fraser et al. 1997)
- A set of classes that embodies an **abstract design** for solutions to a family of related problems (Johnson and Foote 1988)
- A set of cooperating classes that make up a **reusable design** for a specific class of software (Gamma et al. 1995)
- The **skeleton** of an application that can be **customized** by an application developer (Fayad et al. 1999)
- A definition of a **high-level language** with which applications within a domain are created through **specialization** (Pree 1999)
- An **architectural pattern** that provides an **extensible template** for applications within a domain (Booch et al. 1999)

Characteristics of a Framework

Skeleton / design / high-level language / template

→Application at a high level of abstraction; a basic conceptual structure

Characteristics of a Framework

Skeleton / design / high-level language / template

→Application at a high level of abstraction; a basic conceptual structure

Application / class of software / within a domain

→Behavior in a well-defined domain

Characteristics of a Framework

Skeleton / design / high-level language / template

→Application at a high level of abstraction; a basic conceptual structure

Application / class of software / within a domain

→Behavior in a well-defined domain

Cooperating / collaborating classes

→Defined protocol for a set of well-defined components/objects; user must understand the protocol(s) and program accordingly

Characteristics of a Framework

Skeleton / design / high-level language / template

→Application at a high level of abstraction; a basic conceptual structure

Application / class of software / within a domain

→Behavior in a well-defined domain

Cooperating / collaborating classes

→Defined protocol for a set of well-defined components/objects; user must understand the protocol(s) and program accordingly

Customize / abstract classes / reusable / specialize

→Flexibility; can be tailored to a concrete context (within the domain)

Characteristics of a Framework

Skeleton / design / high-level language / template

→Application at a high level of abstraction; a basic conceptual structure

Application / class of software / within a domain

→Behavior in a well-defined domain

Cooperating / collaborating classes

→Defined protocol for a set of well-defined components/objects; user must understand the protocol(s) and program accordingly

Customize / abstract classes / reusable / specialize

→Flexibility; can be tailored to a concrete context (within the domain)

Classes / implementation

→ Reuse of working code as well as reuse of design

Frameworks: Examples

Pay Station framework (evolved from AlphaTown Pay Station application)

- A skeleton within a particular domain
- Collaborating classes
- Classes customized for a particular product variant
- Implementation / code reuse as well as design

Frameworks: Examples

Pay Station framework (evolved from AlphaTown Pay Station application)

- A skeleton within a particular domain
- Collaborating classes
- Classes customized for a particular product variant
- Implementation / code reuse as well as design

HotCIv

- Framework supports different game variants

Frameworks: Examples

Pay Station framework (evolved from AlphaTown Pay Station application)

- A skeleton within a particular domain
- Collaborating classes
- Classes customized for a particular product variant
- Implementation / code reuse as well as design

HotCIv

- Framework supports different game variants

MiniDraw

- A framework that supports user interaction with 2D image-based graphics via mouse events
- Will use for HotCiv GUI

Frameworks: Use

Who are the users of a framework?

- Application developers – customize the framework for a particular need

Frameworks: Use

Who are the users of a framework?

- Application developers – customize the framework for a particular need

How do developers choose a framework?

- Framework domain is relevant to the problem being addressed
- Flexible/customizable in ways necessary for the application
- Reliable implementation, well-tested

Frameworks: Use

Who are the users of a framework?

- Application developers – customize the framework for a particular need

How do developers choose a framework?

- Framework domain is relevant to the problem being addressed
- Flexible/customizable in ways necessary for the application
- Reliable implementation, well-tested
- Framework provides design/functionality that is otherwise expensive to acquire/develop (lower development and maintenance cost)
    - May have initial training overhead

Frameworks: Customization

Key point: Frameworks are not customized by code modification!

Frameworks: Customization

Key point: Frameworks are not customized by code modification!

- Source code must not be altered, even if accessible
- Customize through mechanisms provided by framework developers

→ **Change by addition**

Frameworks: Customization

Key point: Frameworks are not customized by code modification!

- Source code must not be altered, even if accessible
- Customize through mechanisms provided by framework developers

→ **Change by addition**

Frozen spots: Parts of the framework code that cannot be altered; define the basic design and protocols

Hot spots / hook methods / variability points: Clearly defined parts of the framework in which specialization code can alter or add behavior to the final application

Frameworks: Customization

Key point: Frameworks are not customized by code modification!

→Source code must not be altered, even if accessible

→Customize through mechanisms provided by framework developers

→**Change by addition**

Recall: Template Method

Frozen spots: Parts of the framework code that cannot be altered; define the basic design and protocols

Hot spots / hook methods / variability points: Clearly defined parts of the framework in which specialization code can alter or add behavior to the final application

Frameworks: Customization

Key point: Frameworks are not customized by code modification!

→Source code must not be altered, even if accessible

→Customize through mechanisms provided by framework developers

→**Change by addition**

Recall: Template Method

Frozen spots: Parts of the framework code that cannot be altered; define the basic design and protocols

Hot spots / hook methods / variability points: Clearly defined parts of the framework in which specialization code can alter or add behavior to the final application

→ Subclassing or delegation

Frameworks: Customization

Key point: Frameworks must use dependency injection!

Frameworks: Customization

Key point: Frameworks must use dependency injection!

- Enables customization by the application developer
- Objects that define variability points (hot spots) are instantiated by the **application** code and **injected** into the framework

Frameworks: Customization

Key point: Frameworks must use dependency injection!

- Enables customization by the application developer
- Objects that define variability points (hot spots) are instantiated by the **application** code and **injected** into the framework

How can a developer define “hot spot” objects?

Existing concrete classes: predefined classes in the framework	**Subclass abstract class:** the framework contains abstract classes	**Implement an interface**: the framework contains an interface

Frameworks: Customization

Key point: Frameworks must use dependency injection!

- Enables customization by the application developer
- Objects that define variability points (hot spots) are instantiated by the **application** code and **injected** into the framework

How can a developer define “hot spot” objects?

Existing concrete classes: predefined classes in the framework	**Subclass abstract class:** the framework contains abstract classes	**Implement an interface**: the framework contains an interface

Ease of use / speed — Control

Frameworks: Customization

Key point: Frameworks must use dependency injection!

- Enables customization by the application developer
- Objects that define variability points (hot spots) are instantiated by the **application** code and **injected** into the framework

How can a developer define “hot spot” objects?

Existing concrete classes: predefined classes in the framework	**Subclass abstract class:** the framework contains abstract classes	**Implement an interface**: the framework contains an interface

Frameworks: Customization

Key point: Frameworks should provide a spectrum of no implementation (interface) to partial implementation (abstract) to full implementation for variability points

Existing concrete classes: predefined classes in the framework	**Subclass abstract class:** the framework contains abstract classes	**Implement an interface**: the framework contains an interface

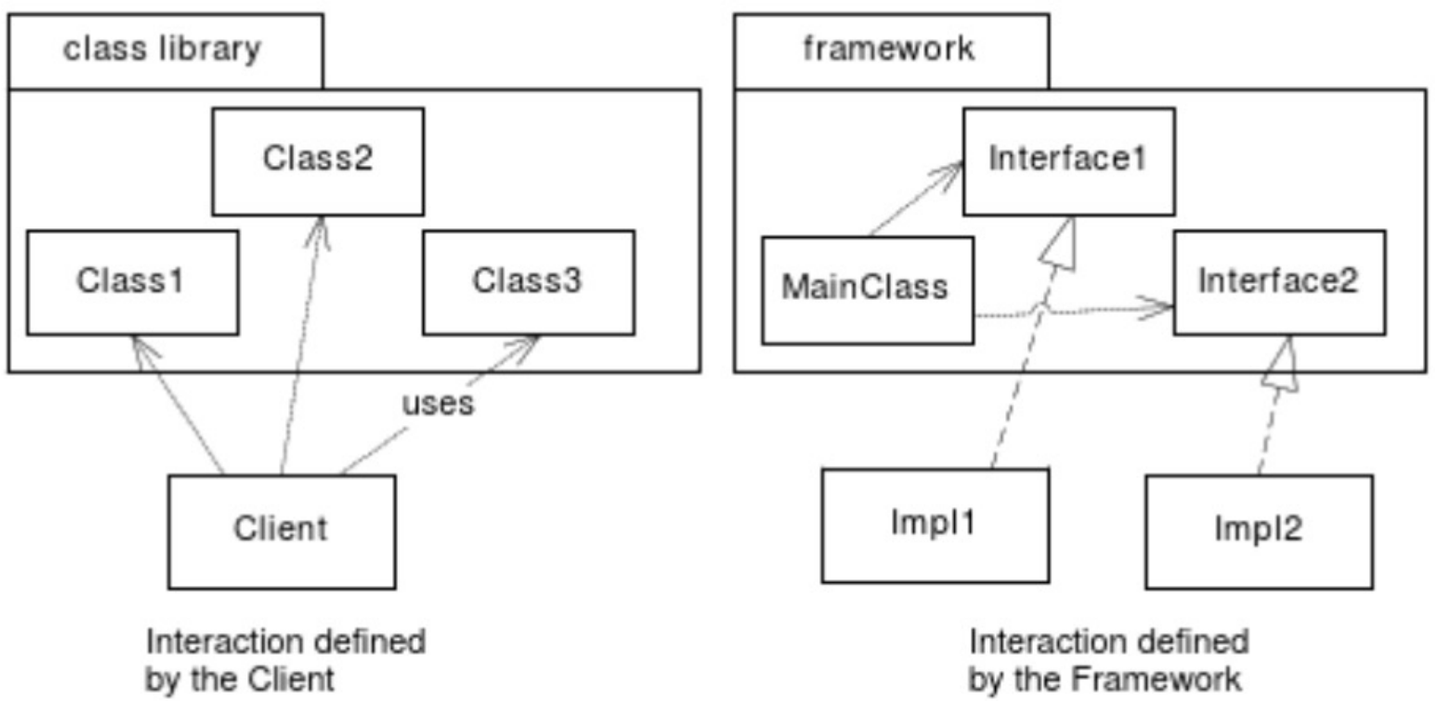
Frameworks: Inversion of Control

The **framework** (not the application developer) defines the **flow of control** in an application.

Frameworks: Inversion of Control

The **framework** (not the application developer) defines the **flow of control** in an application.

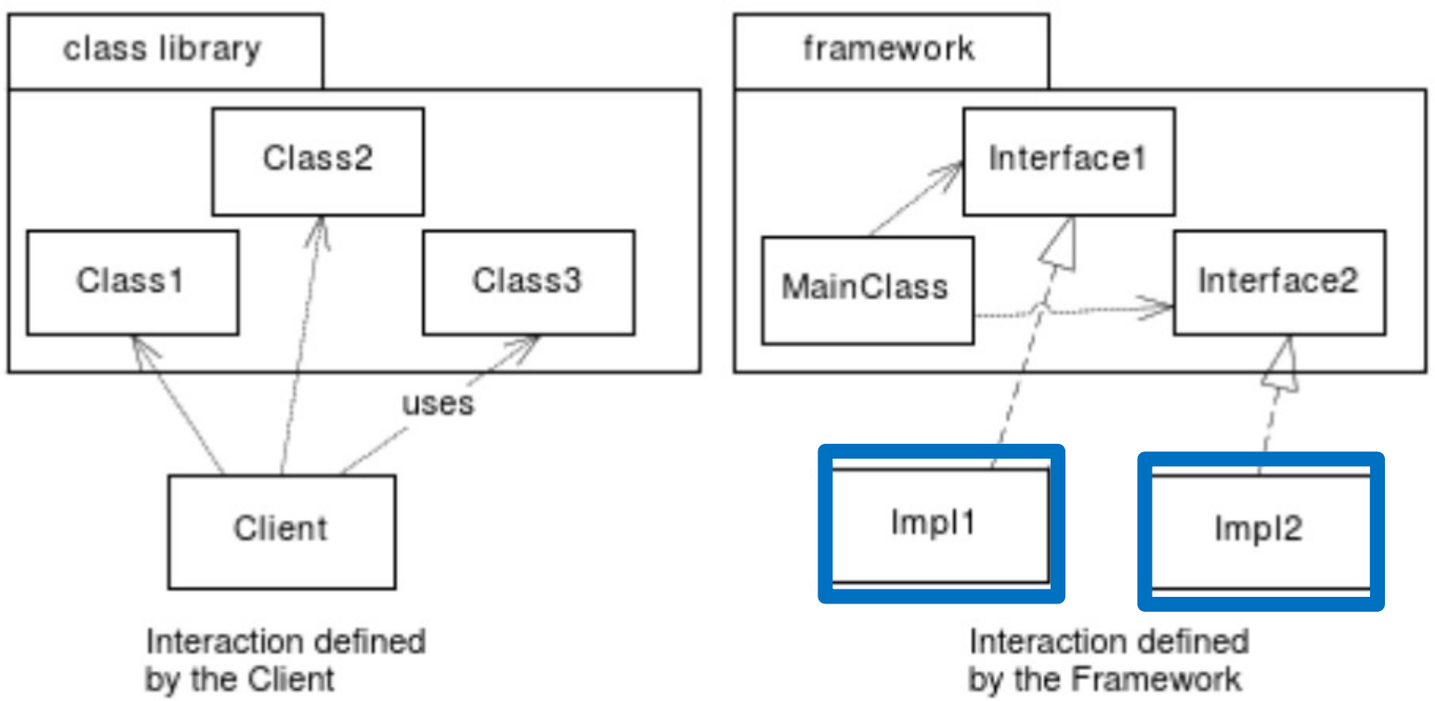
→ Contrast with libraries, where the application maintains control



Frameworks: Inversion of Control

The **framework** (not the application developer) defines the **flow of control** in an application.

→ Contrast with libraries, where the application maintains control



Hot spot objects

Framework Composition

We may want to use multiple frameworks to cover multiple domains in a single system

- Graphical framework, distributed framework, etc.

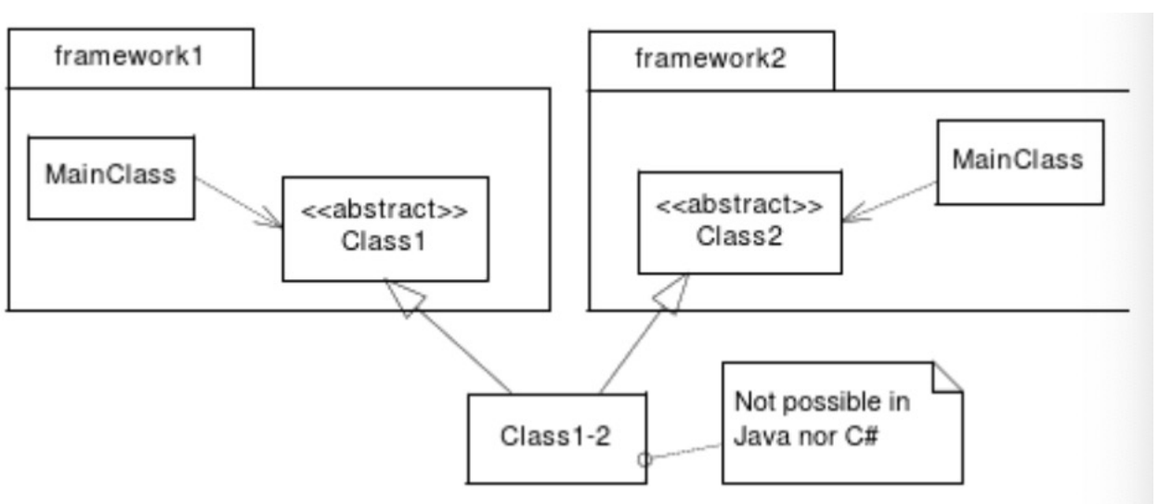
What if the frameworks are based on subclassing?

Framework Composition

We may want to use multiple frameworks to cover multiple domains in a single system

- Graphical framework, distributed framework, etc.

What if the frameworks are based on subclassing?

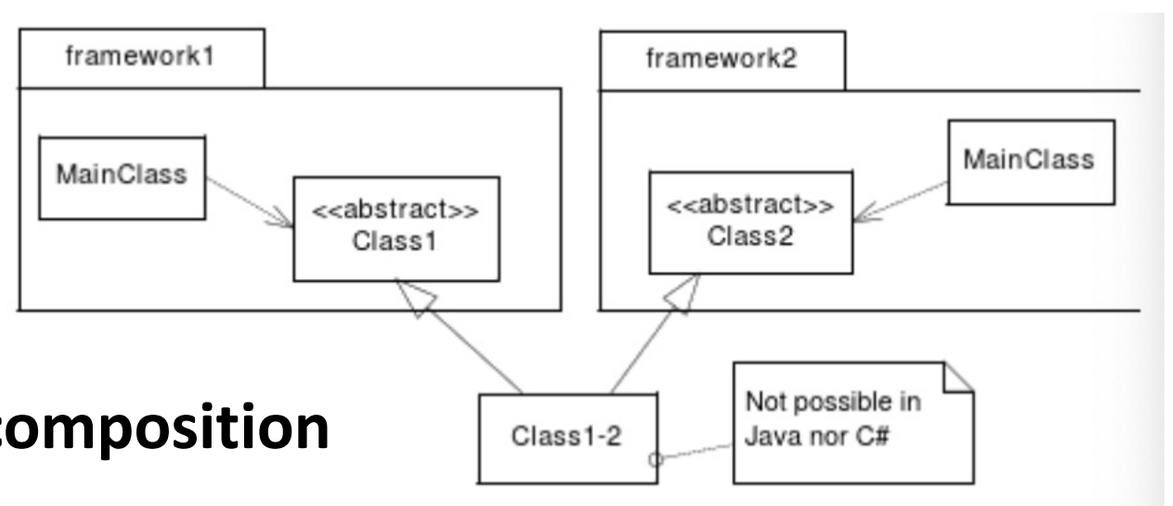


Framework Composition

We may want to use multiple frameworks to cover multiple domains in a single system

- Graphical framework, distributed framework, etc.

What if the frameworks are based on subclassing?



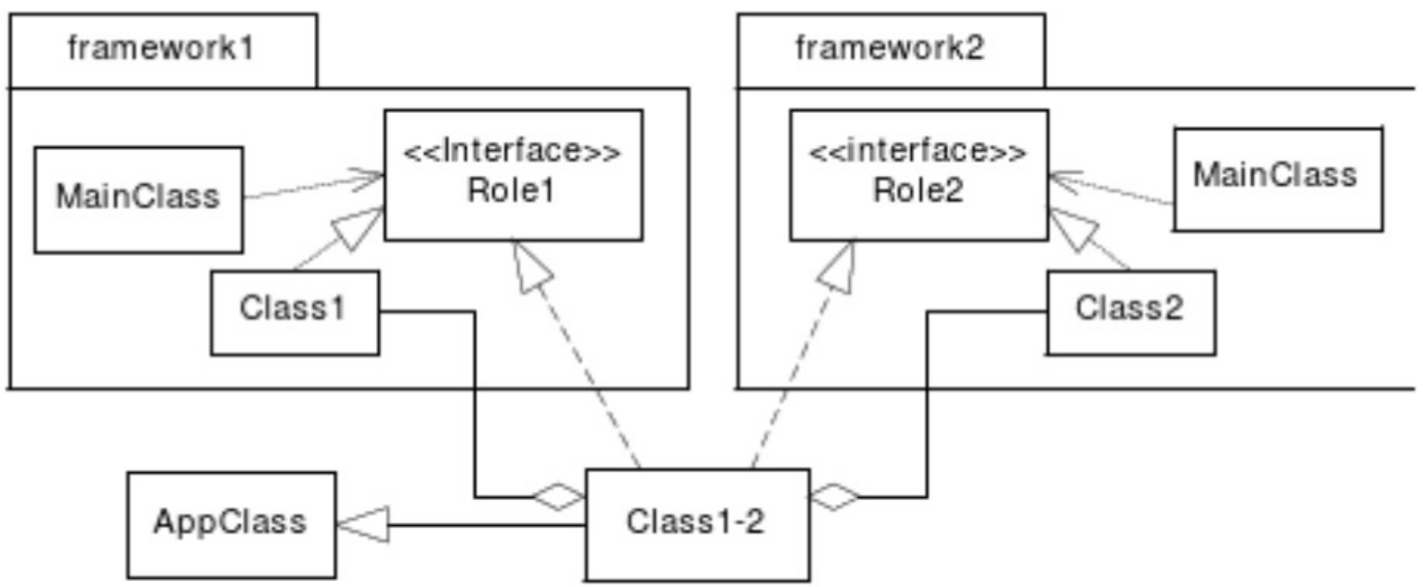
→**Framework composition problem**

Framework Composition

We may want to use multiple frameworks to cover multiple domains in a single system

- Graphical framework, distributed framework, etc.

If frameworks follow “program to an interface” :

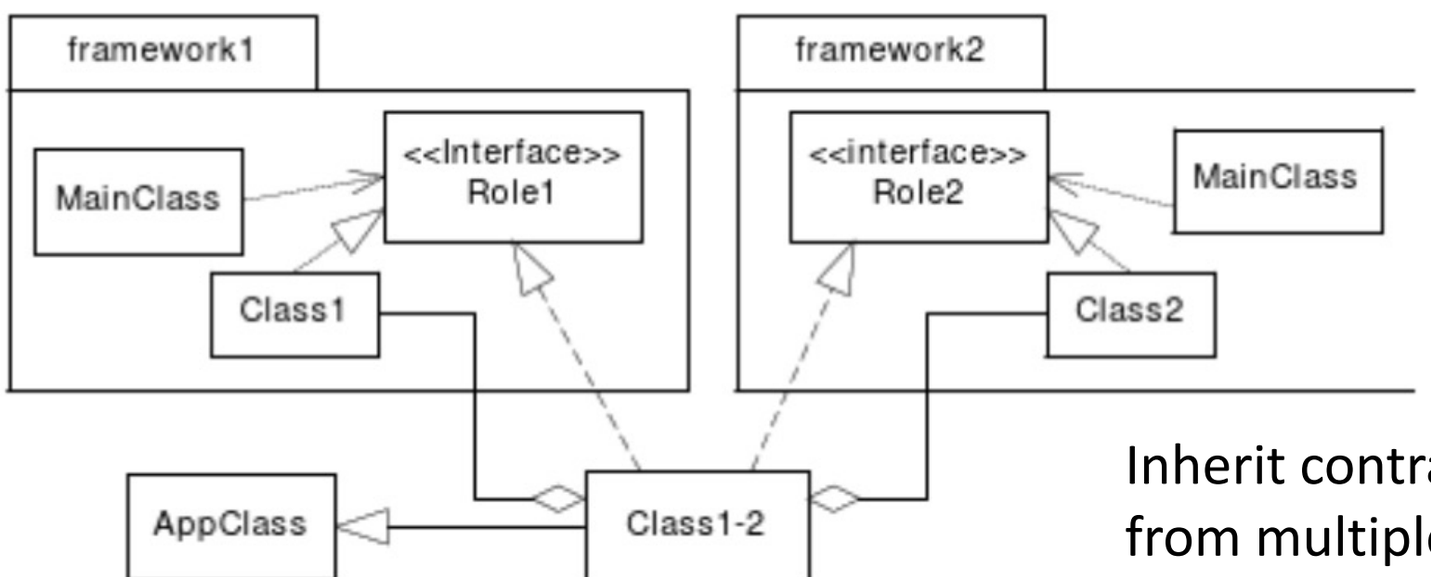


Framework Composition

We may want to use multiple frameworks to cover multiple domains in a single system

- Graphical framework, distributed framework, etc.

If frameworks follow “program to an interface” :



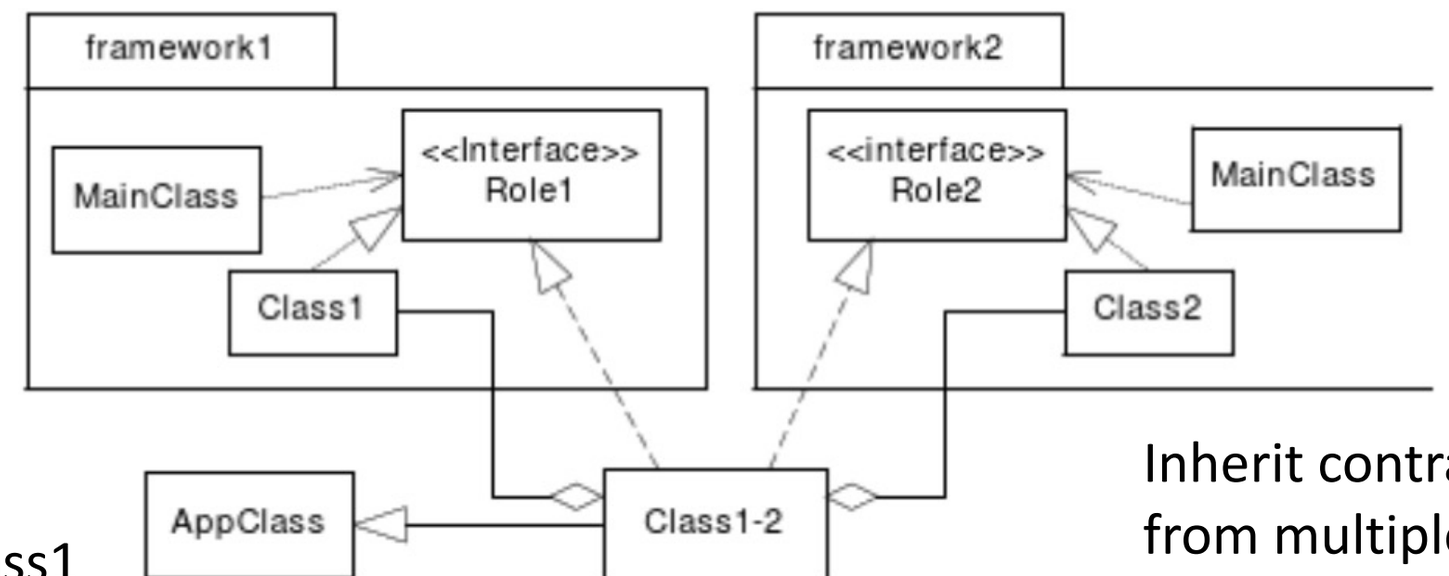
Inherit contracts from multiple interfaces

Framework Composition

We may want to use multiple frameworks to cover multiple domains in a single system

- Graphical framework, distributed framework, etc.

If frameworks follow “program to an interface” :



Class 1-2 can inherit from AppClass, delegate to Class1

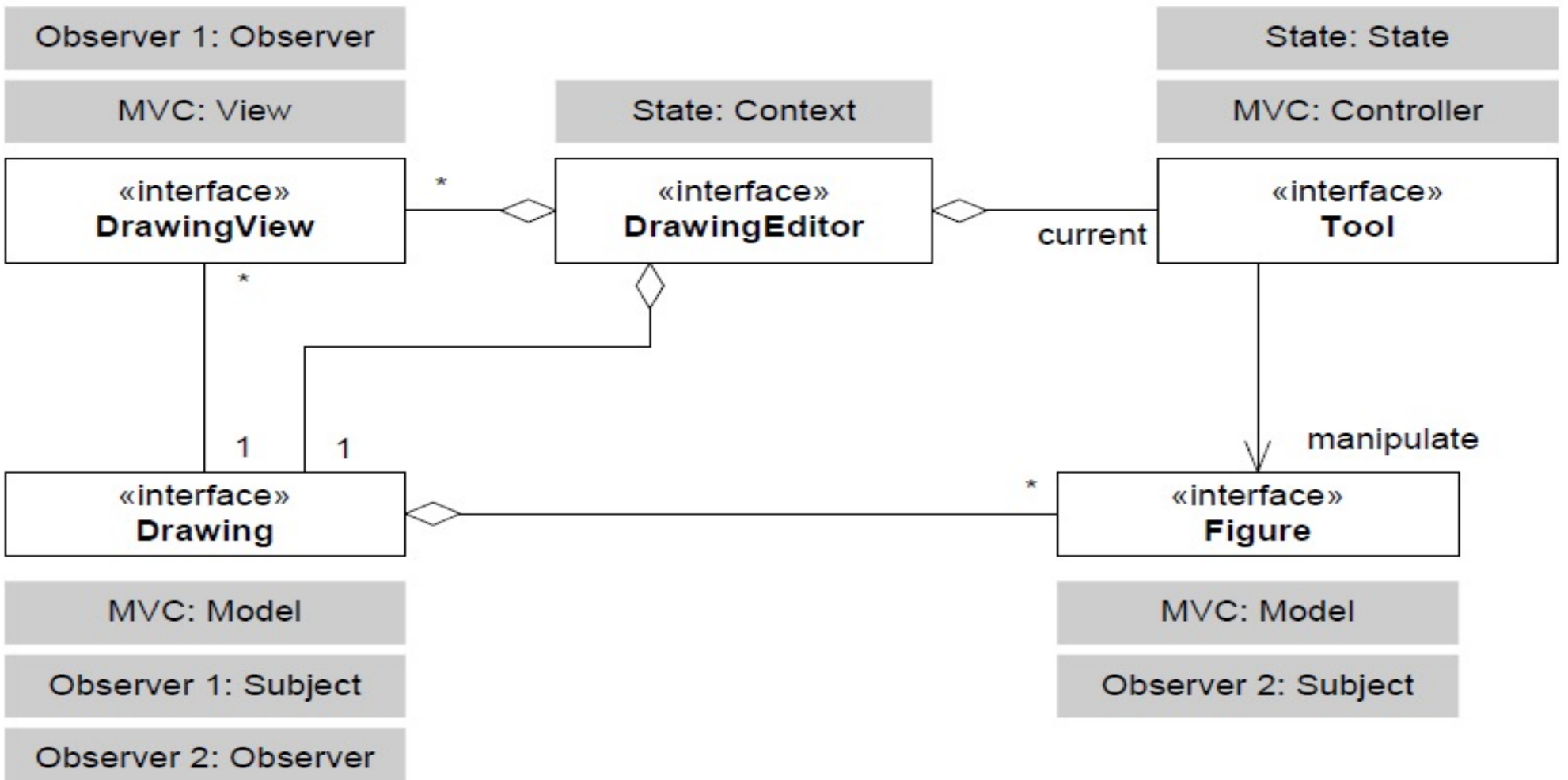
Inherit contracts from multiple interfaces

Next Time: MiniDraw

MiniDraw is a framework that supports user interaction with 2D image-based graphics via mouse events

- Hot spots / frozen spots:
  - Customization is done by defining new tools, adding image files, or configuring the factory with proper implementations of MiniDraw's roles
- Inversion of control
  - When calling open(), it does all the processing of mouse events, calls your tool, draws your images at the correct times
- Framework composition
  - MiniDraw + Java Swing
  - MiniDraw uses compositional design and design patterns
- Reuse of working code as well as reuse of design

Next Time: MiniDraw



Next Time: MiniDraw