Comp 401
Command Objects and Undo

Instructor: Prasun Dewan

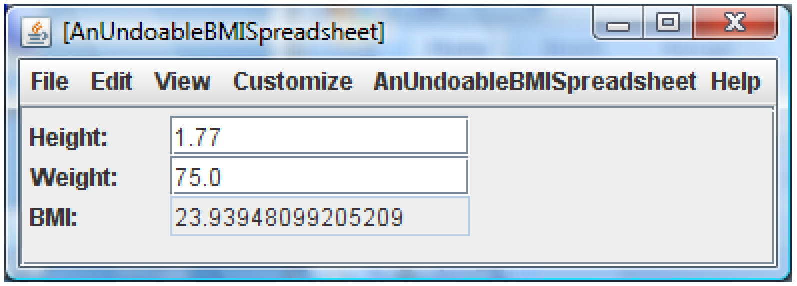
Prerequisites

- Animation Threads Commands

Topics

- Command Object
  - Object representing an action invocation such as "Do your homework".
- Threads
  - Support non blocking action invocation.
- Undo/Redo
  - Supports undoable/re-doable commands (action invocations)

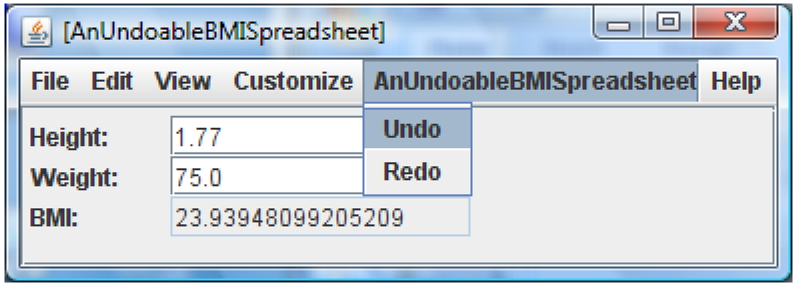
Undo: Initial State


[AnUndoableBMISpreadsheet]
File Edit View Customize AnUndoableBMISpreadsheet Help

Height: 1.77
Weight: 75.0
BMI: 23.93948099205209

Initial State

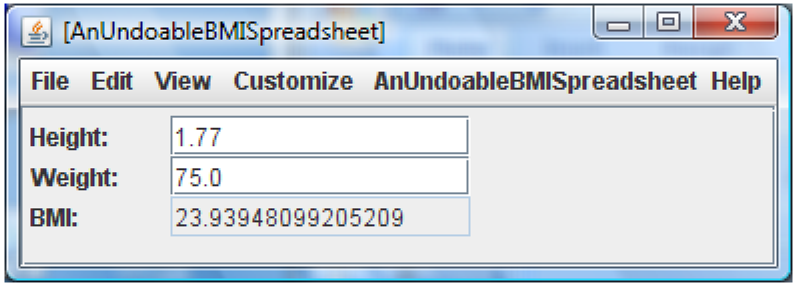
UNDO INITIAL STATE



Height: 1.77
Weight: 75.0
BMI: 23.93948099205209

Undo Initial State

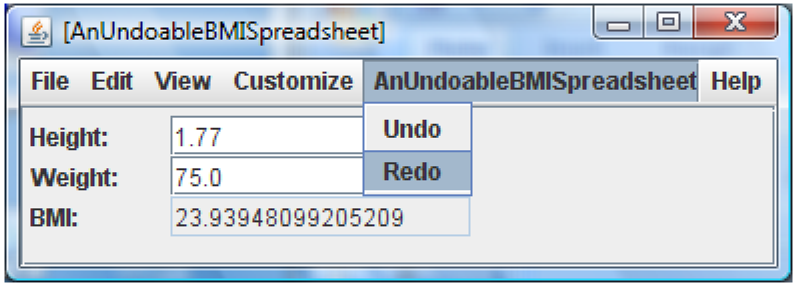
Cannot Always Undo


[AnUndoableBMISpreadsheet]
File Edit View Customize AnUndoableBMISpreadsheet Help

Height: 1.77
Weight: 75.0
BMI: 23.93948099205209

If no command executed, undo does nothing

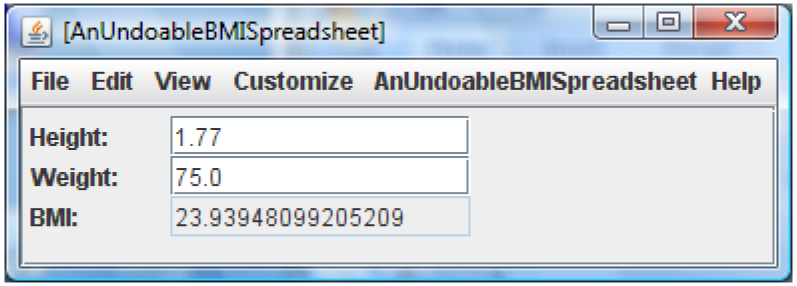
Redo Initial State



Height: 1.77
Weight: 75.0
BMI: 23.93948099205209

If no command executed, undo does nothing

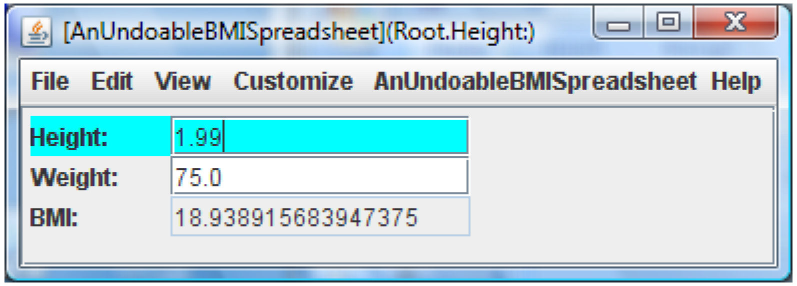
Cannot Always Redo


[AnUndoableBMISpreadsheet]
File Edit View Customize AnUndoableBMISpreadsheet Help

Height: 1.77
Weight: 75.0
BMI: 23.93948099205209

If no command executed, redo does nothing

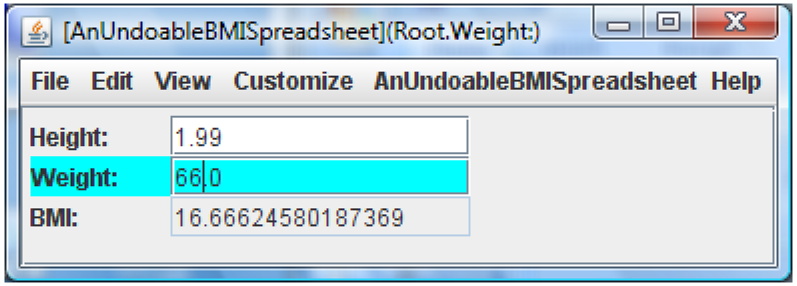
Change Height


[AnUndoableBMISpreadsheet](Root.Height:)
File Edit View Customize AnUndoableBMISpreadsheet Help

Height: 1.99
Weight: 75.0
BMI: 18.938915683947375

Execute change height command.

Change Weight


[AnUndoableBMISpreadsheet](Root.Weight:)

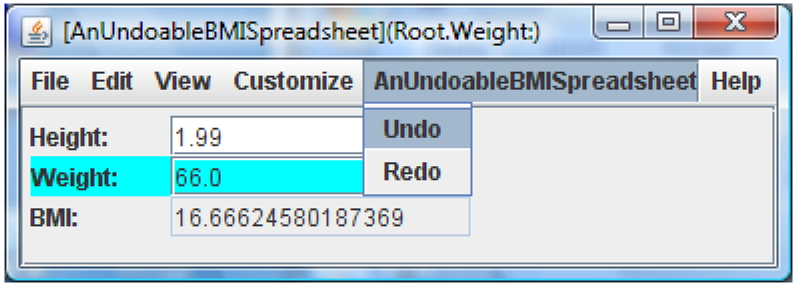
File Edit View Customize AnUndoableBMISpreadsheet Help

Height: 1.99
Weight: 66.0
BMI: 16.66624580187369

Execute change weight command.

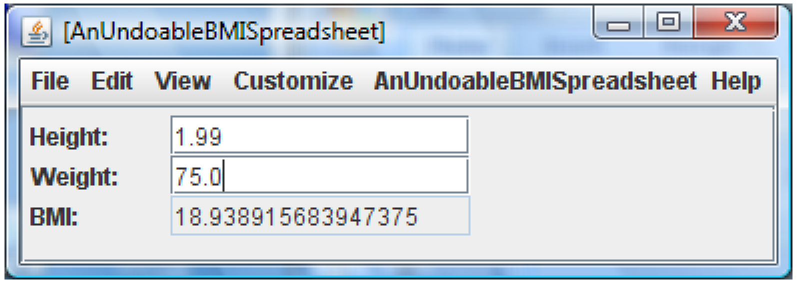
Height and bmi change.

Undo



Height: 1.99
Weight: 66.0
BMI: 16.66624580187369

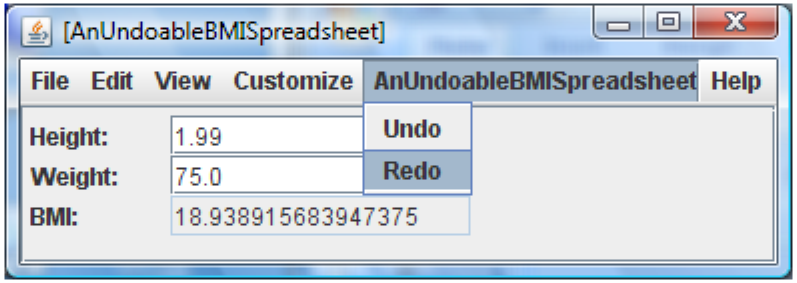
UNDO


[AnUndoableBMISpreadsheet]
File Edit View Customize AnUndoableBMISpreadsheet Help

Height: 1.99
Weight: 75.0
BMI: 18.938915683947375

Weightand BMI both undone to restore state before setWeight() call

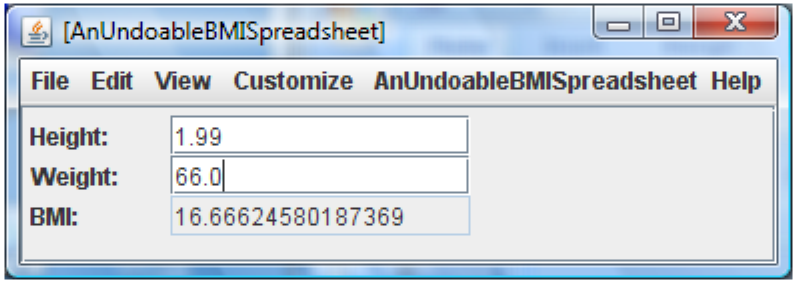
Redo



Height: 1.99
Weight: 75.0
BMI: 18.938915683947375

Redo last undone command

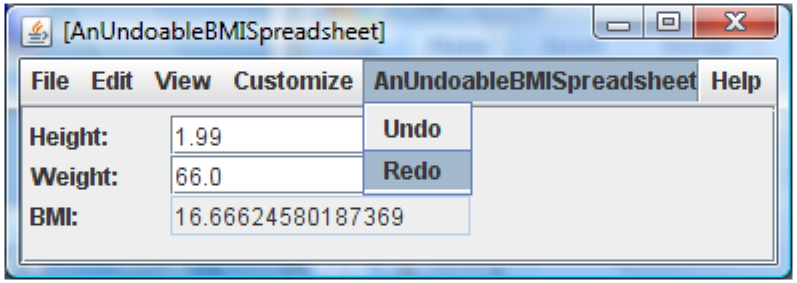
Redo


[AnUndoableBMISpreadsheet]
File Edit View Customize AnUndoableBMISpreadsheet Help

Height: 1.99
Weight: 66.0
BMI: 16.66624580187369

Last undone command reexecuted

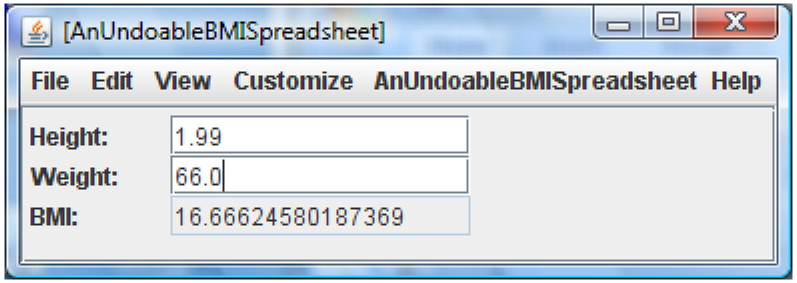
Last Command Undo



Height: 1.99
Weight: 66.0
BMI: 16.66624580187369

Redo after a redo.

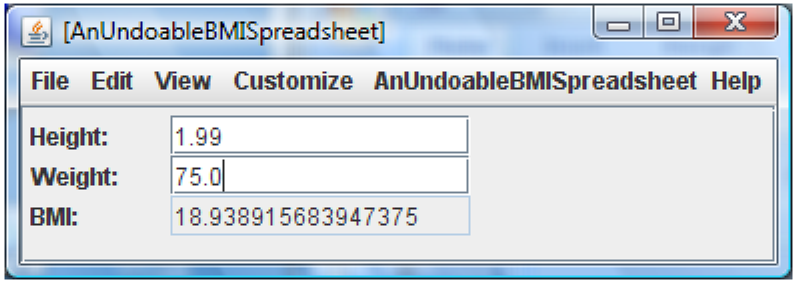
Cannot Always Redo


[AnUndoableBMISpreadsheet]
File Edit View Customize AnUndoableBMISpreadsheet Help

Height: 1.99
Weight: 66.0
BMI: 16.66624580187369

No undo commands to redo

Undo Implementation


[AnUndoableBMISpreadsheet]

File Edit View Customize AnUndoableBMISpreadsheet Help

Height: 1.99
Weight: 75.0
BMI: 18.938915683947375

Weightand BMI both undone to restore state before setWeight() call

setWeight() called with old weight.

Undoable command object remembers method and its parameters.

In the application could have multiple spreadsheets, points, … all sharing one undo history.

Special global undoer keeps track of command history

setWeight in undoable creates command and gives it to undoer

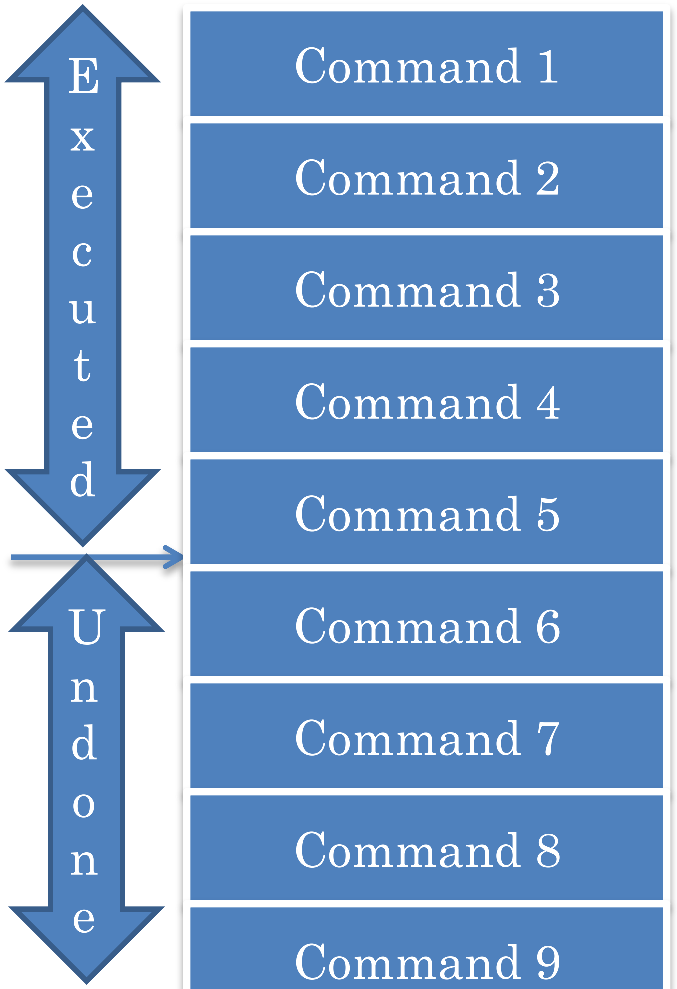
Undo in BMI

$$\begin{tikzcd}
\text{ABMISpreadsheet} & \text{ASetWeight(Height)Command} \arrow[l] \\
\text{AnUndoableBMISpreadsheet} \arrow[u] \arrow[r] & \text{AHistoryUndoer} \arrow[u]
\end{tikzcd}$$

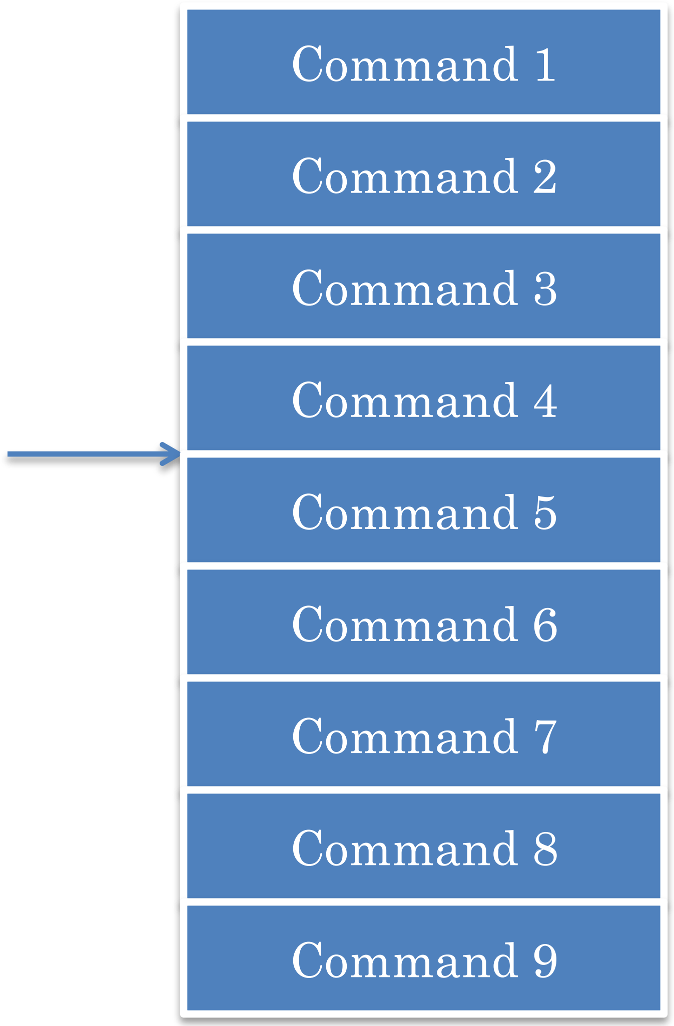
General Undoer Interface

```
public interface Undoer {
    public void undo();
    public void execute(Command command);
    public void redo();
}
```

History Undoer

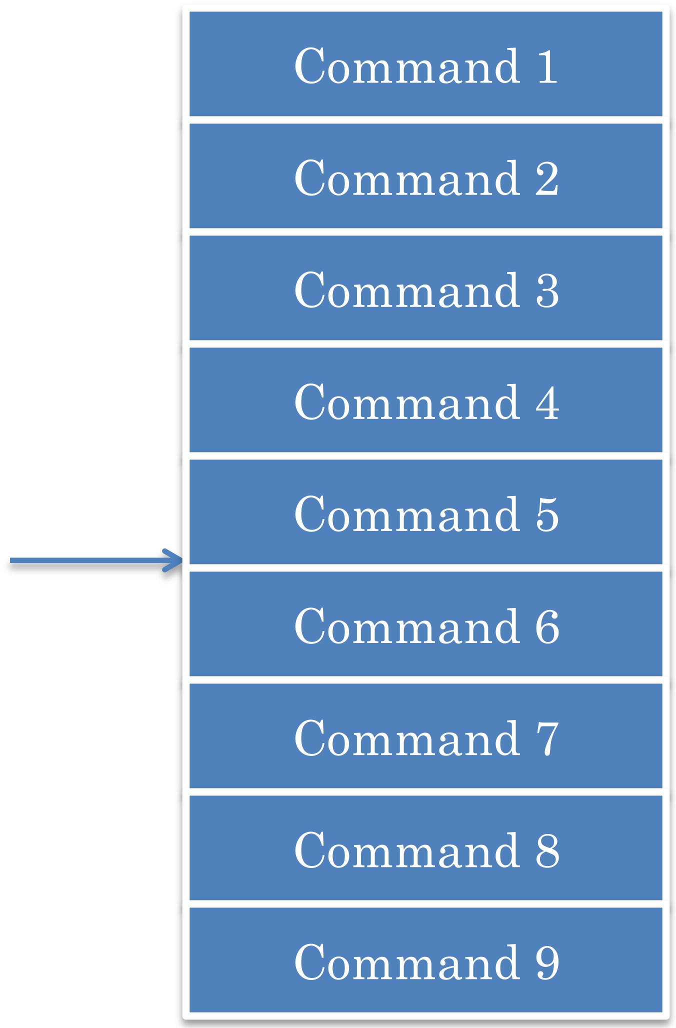


Undo

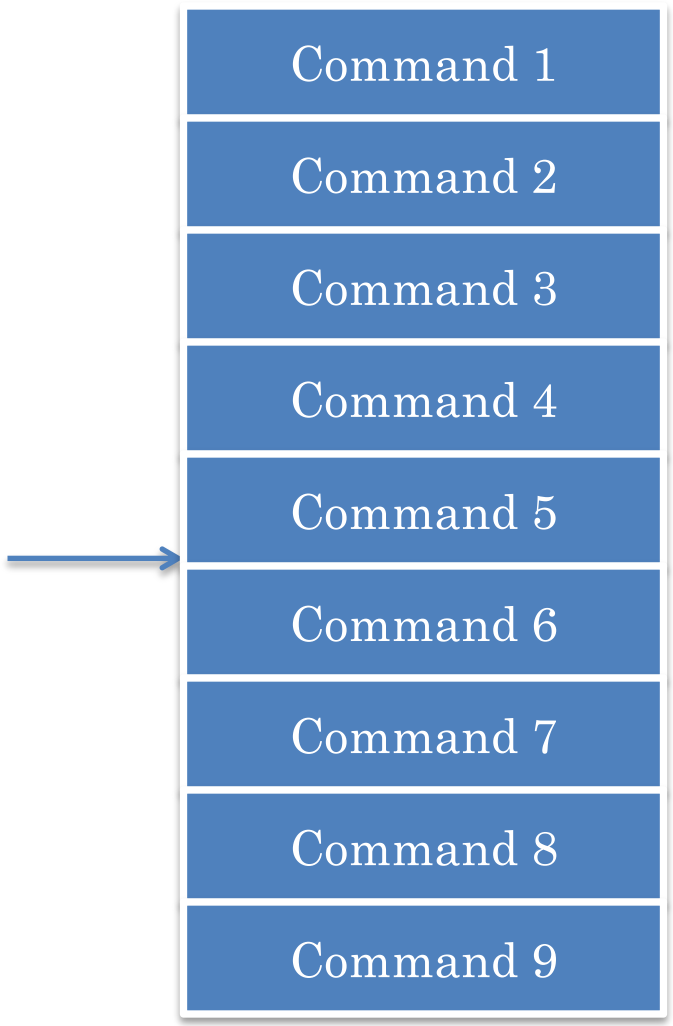


Command 1
Command 2
Command 3
Command 4
Command 5
Command 6
Command 7
Command 8
Command 9

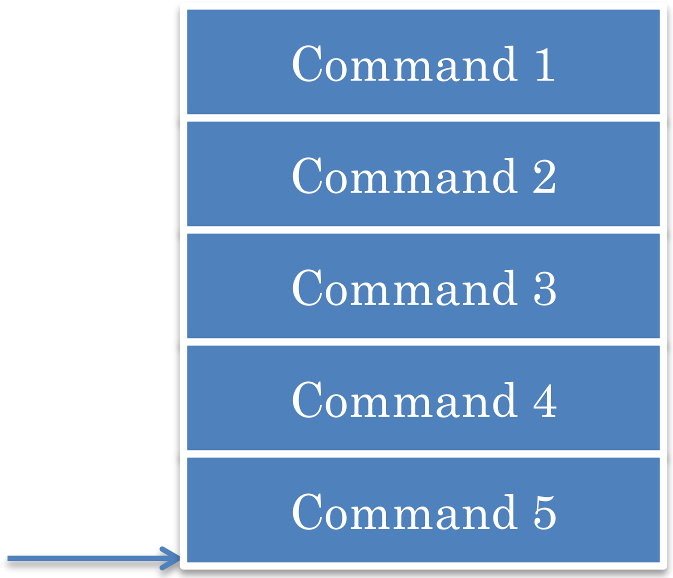
Redo


Command 1
Command 2
Command 3
Command 4
Command 5
Command 6
Command 7
Command 8
Command 9

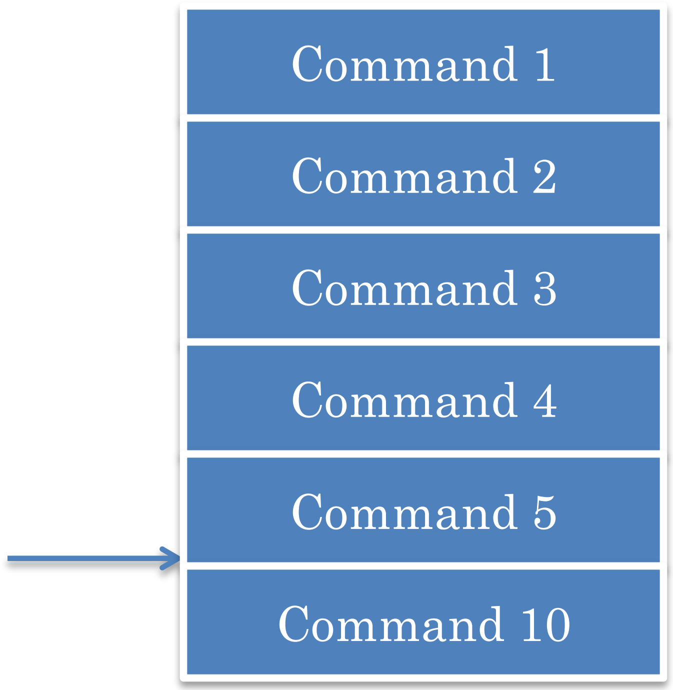
EXECUTE


Command 1
Command 2
Command 3
Command 4
Command 5
Command 6
Command 7
Command 8
Command 9

Execute



Execute



General History Undoer

```
public class HistoryUndoer implements Undoer {
  List<Command> historyList = new ArrayList();
  int nextCommandIndex = 0;
  public void execute (Command c) {
    while(nextCommandIndex < historyList.size()) {
      historyList.remove(nextCommandIndex); // clear redo chain
    }
    c.execute();
    historyList.add(c);
    nextCommandIndex++;
  }
  public void undo() {
    if (nextCommandIndex == 0)return;
    nextCommandIndex--;
    Command c = historyList.get(nextCommandIndex);
    c.undo();
  }
  public void redo() {
     if (nextCommandIndex == historyList.size()) return;
     Command c = historyList.get(nextCommandIndex);
     c.execute();
     nextCommandIndex++;
  }
}
```

Undoable Command

```
public interface Command {
    public  void execute();
    public  void  undo();
}
```

ASetWeightCommand

```
public class ASetWeightCommand implements Command {
BMISpreadsheet bmiSpreadsheet;
 double oldWeight;
 double weight;
 public ASetWeightCommand (BMISpreadsheet
theBMISpreadsheet, double theWeight) {
       bmiSpreadsheet = theBMISpreadsheet;
       weight = theWeight;
       oldWeight = bmiSpreadsheet.getWeight();
}
 public void execute() {bmiSpreadsheet.setWeight(weight);}
 public void undo() {bmiSpreadsheet.setWeight(oldWeight);}
}
```

ASetHeightCommand

```
public class ASetHeightCommand implements Command {
BMISpreadsheet bmiSpreadsheet;
 double oldHeight;
 double height;
 public ASetHeightCommand (BMISpreadsheet
theBMISpreadsheet, double theHeight) {
      bmiSpreadsheet = theBMISpreadsheet;
      height = theHeight;
      oldHeight = bmiSpreadsheet.getHeight();
}
 public void execute() {bmiSpreadsheet.setHeight(height);}
 public void undo() {bmiSpreadsheet.setHeight(oldHeight);}
}
```

Reflection could allow these two command objects to be combined.

Undoable BMISpreadsheet

```
public interface UndoableBMISpreadsheet extends
BMISpreadsheet{
    public  void redo();
    public  void  undo();
}
```

Usually user-invokable undo/redo methods would be provided by a global application object for all objects in the application.

The interface of AnUndoableSpreadsheet and BMiSpreadsheet would be same if global object

Here there is only one application object so undo/redo in UndoableBMISpreadsheet

Undoable BMISpreadsheet

```
public class AnUndoableBMISpreadsheet implements
UndoableBMISpreadsheet {
	BMISpreadsheet bmiSpreadsheet;
	Undoer undoer;
	public AnUndoableBMISpreadsheet (BMISpreadsheet
theBMISpreadsheet, Undoer theUndoer) {
		bmiSpreadsheet = theBMISpreadhseet;
		undoer = theUndoer;
	}
	public double getBMI() {
		return bmiSpreadsheet.getBMI();
	}
	public double getHeight() {
		return bmiSpreadsheet.getHeight();
	}
	public double getWeight() {
		return bmiSpreadsheet.getWeight();
	}
```

UNDOABLE BMISPREADSHEET

```
        public void setHeight(double theHeight) {
                undoer.execute(new
ASetHeightCommand(bmiSpreadsheet, theHeight));
        }
        public void setWeight(double theWeight) {
                undoer.execute(new
ASetWeightCommand(bmiSpreadsheet, theWeight));
        }
        public void undo() {undoer.undo();}
        public void redo() {undoer.redo();}
}
```

Undoable BMISpreadsheet

```
      public void setHeight(double theHeight) {
            undoer.execute(new ASetHeightCommand(this,
theHeight));
      }
      public void setWeight(double theWeight) {
            undoer.execute(new ASetWeightCommand(this,
theWeight));
      }
      public void undo() {undoer.undo();}
      public void redo() {undoer.redo();}
}
```

this would cause infinite recursion

Undoable BMISpreadsheet (review)

```
public interface UndoableBMISpreadsheet extends
BMISpreadsheet{
    public  void redo();
    public  void  undo();
}
```

Usually user-invokable undo/redo methods would be provided by a global application object for all objects in the application.

The interface of AnUndoableSpreadsheet and BMiSpreadsheet would be same if global object

Here there is only one application object so undo/redo in UndoableBMISpreadsheet

Undoable BMISpreadsheet (Review)

```
public class AnUndoableBMISpreadsheet implements
UndoableBMISpreadsheet {
	BMISpreadsheet bmiSpreadsheet;
	Undoer undoer;
	public AnUndoableBMISpreadsheet (BMISpreadsheet
theBMISpreadsheet, Undoer theUndoer) {
		bmiSpreadsheet = theBMISpreadhseet;
		undoer = theUndoer;
	}
	public double getBMI() {
		return bmiSpreadsheet.getBMI();
	}
	public double getHeight() {
		return bmiSpreadsheet.getHeight();
	}
	public double getWeight() {
		return bmiSpreadsheet.getWeight();
	}
```

Undoable BMISpreadsheet (review)

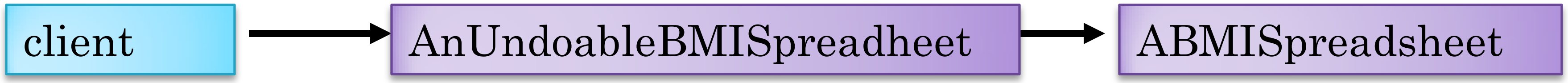
```
      public void setHeight(double theHeight) {
            undoer.execute(new
ASetHeightCommand(bmiSpreadsheet, theHeight));
      }
      public void setWeight(double theWeight) {
            undoer.execute(new
ASetWeightCommand(bmiSpreadsheet, theWeight));
      }
      public void undo() {undoer.undo();}
      public void redo() {undoer.redo();}
}
```

Undoable BMISpreadsheet (review)

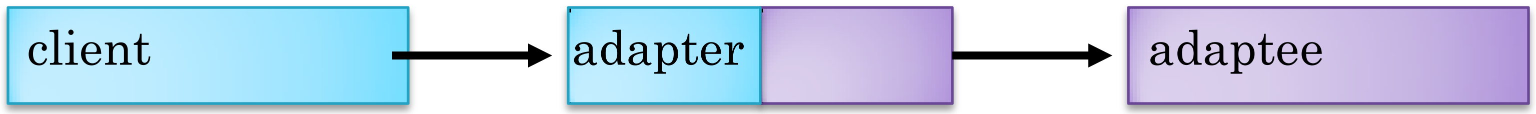
```
      public void setHeight(double theHeight) {
            undoer.execute(new ASetHeightCommand(this,
theHeight));
      }
      public void setWeight(double theWeight) {
            undoer.execute(new ASetWeightCommand(this,
theWeight));
      }
      public void undo() {undoer.undo();}
      public void redo() {undoer.redo();}
}
```

this would cause infinite recursion

Intermediary Delegating Class

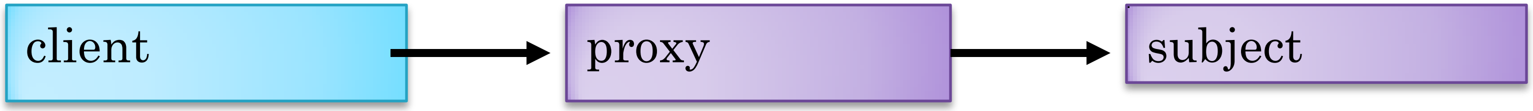


ADAPTER?



- Adapter is a class that sits between a client and adaptee class much like an adapter sits between two objects that need to interact with each other.
- Methods called in adaptee through adapter.
- Degree of adaptation undefined.
  - Assumed no extra functionality offered but some may be removed.
- Methods offered to client
  - Adapted name
  - Adapted parameters.

PROXY

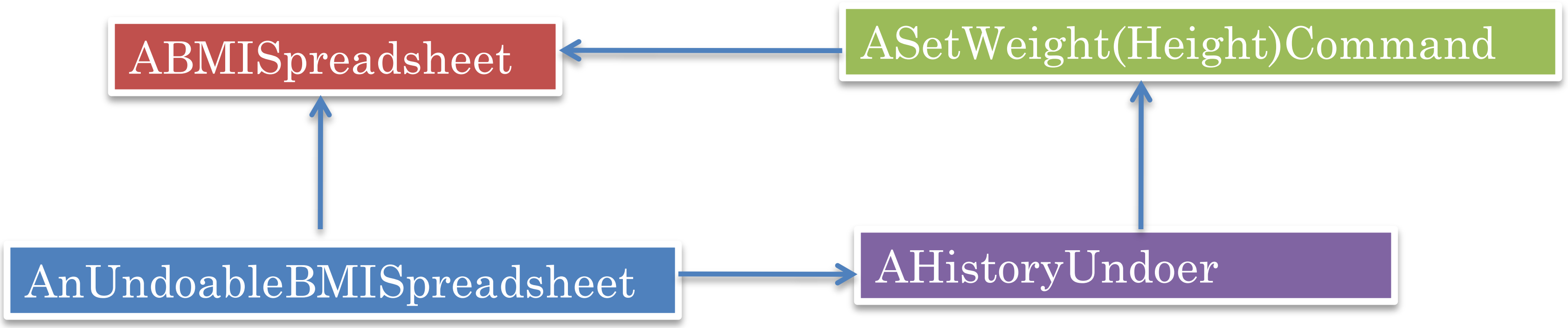


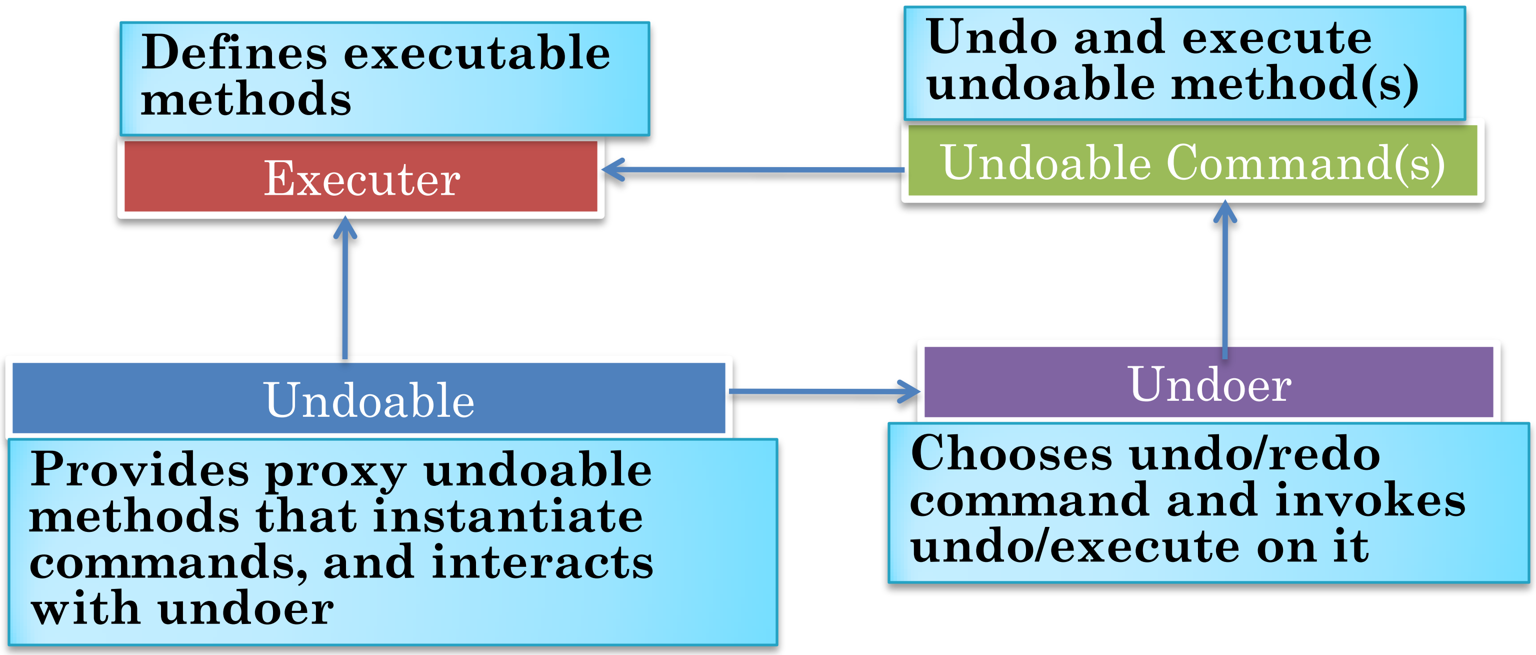
- Proxy is a class that sits between a client and subject class, offering the same interface
- Proxy is a stand-in for real subject.
- Methods called in subject through proxy methods.
- A proxy method does not change the behavior of subject method
- A proxy can add functionality
- Like a regulated power supply, or one with a special fuse

Proxies in Everyday Apps

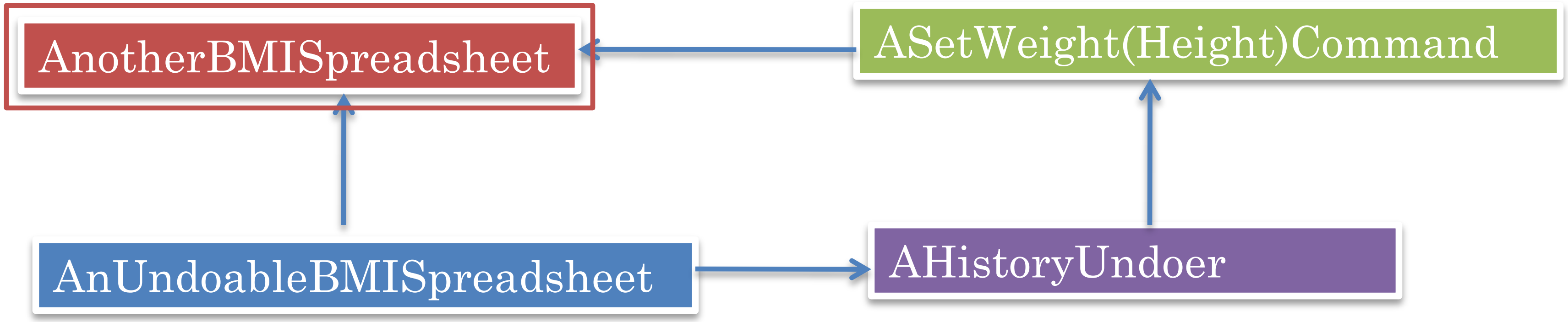
- Proxies adding support for:
  - Logging
  - Collaboration
  - Cache data
  - Redirect to nearest server
  - Access control
  - Assertions
  - Undo/redo

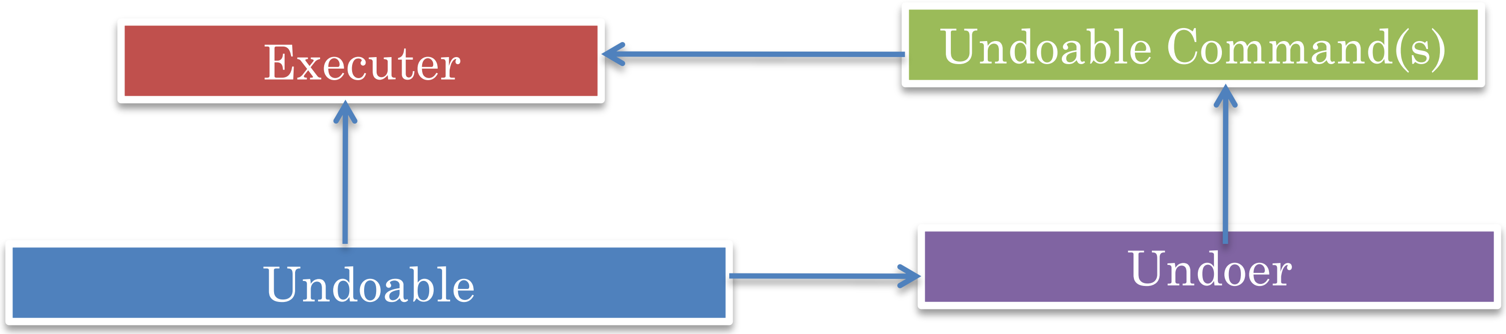
Undo Pattern



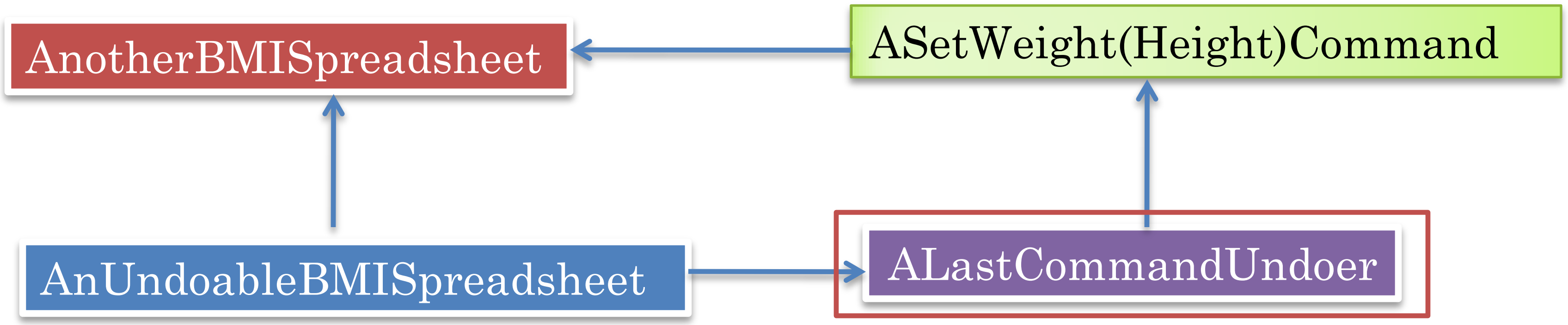

Defines executable methods
Executer
Undo and execute undoable method(s)
Undoable Command(s)
Undoable
Provides proxy undoable methods that instantiate commands, and interacts with undoer
Undoer
Chooses undo/redo command and invokes undo/execute on it

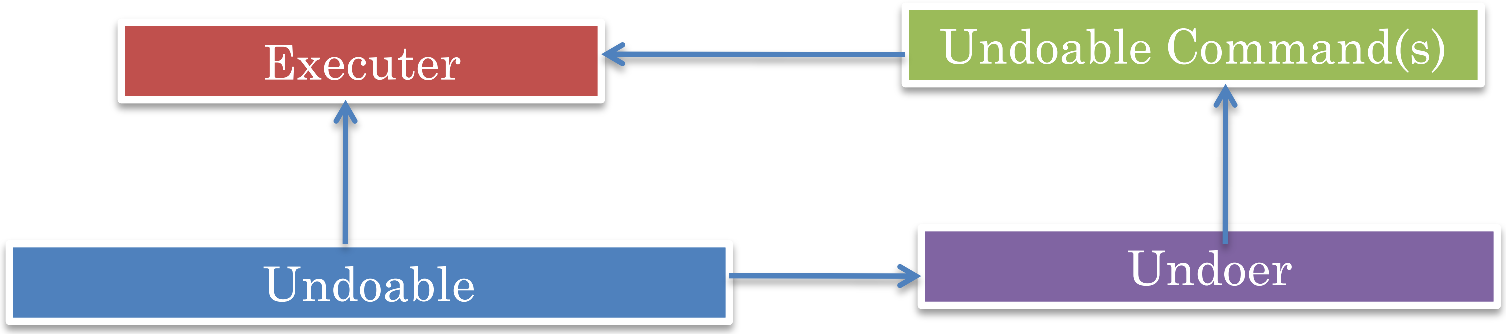
CHANGING UNDOABLE



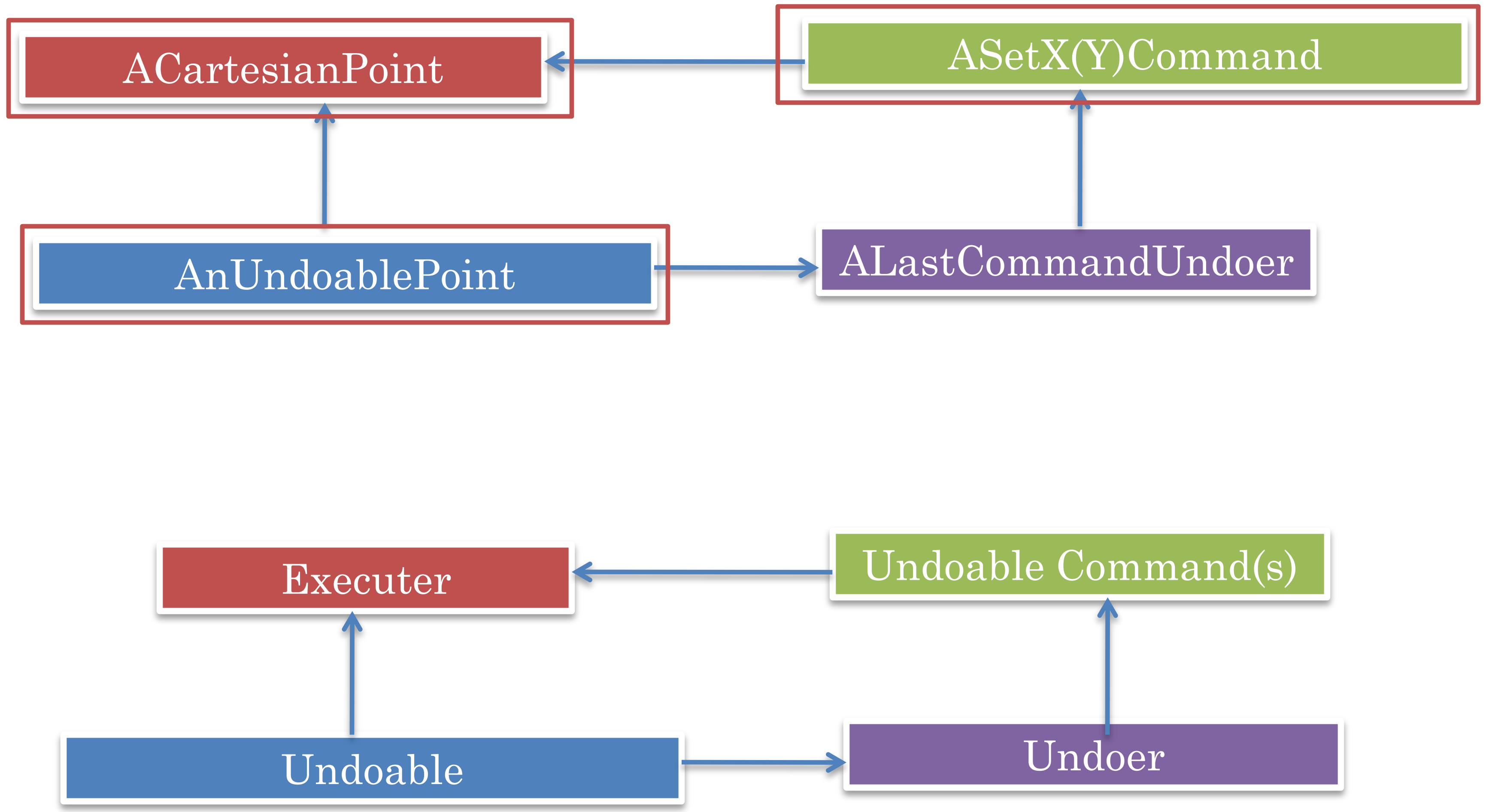


Changing Undoer

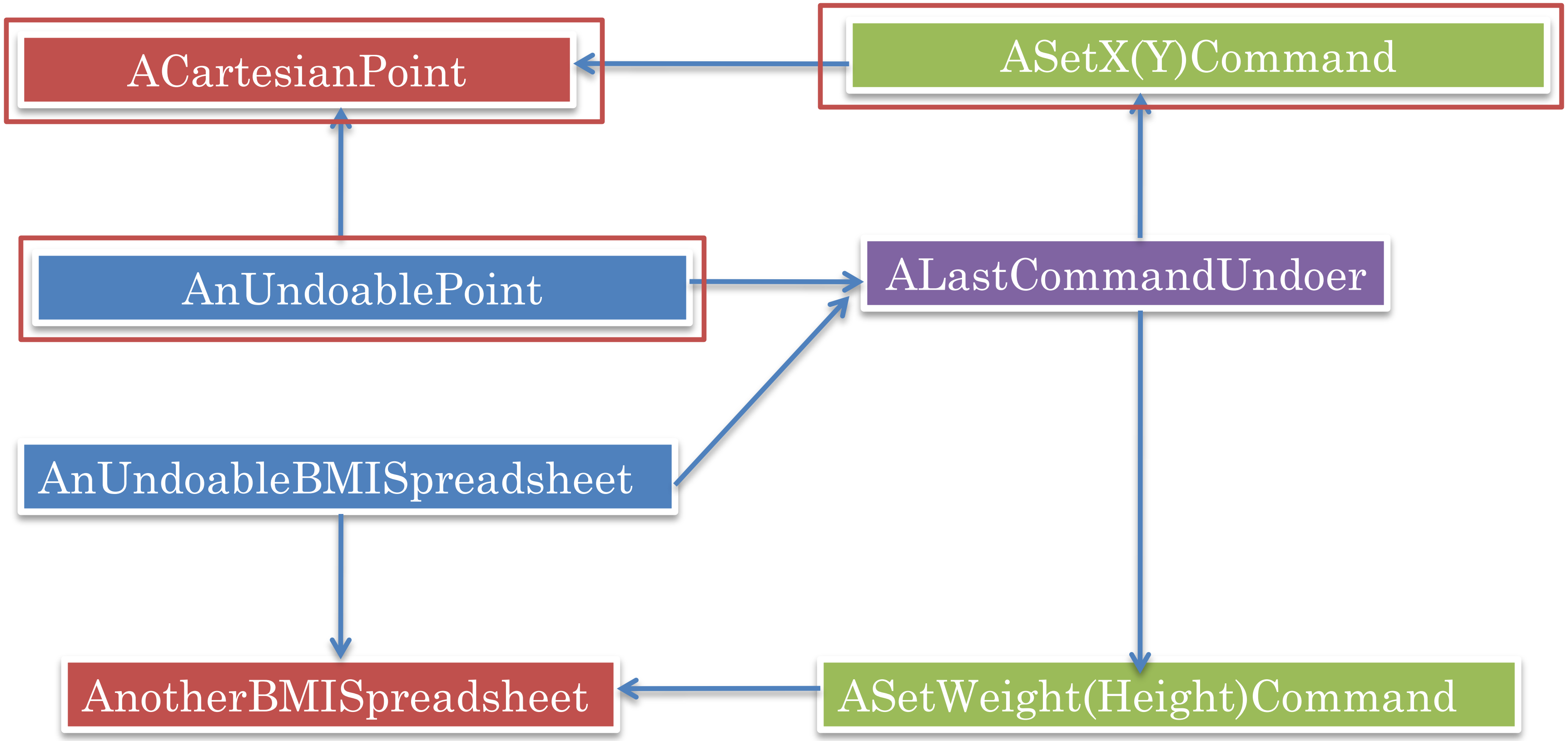




Changing Everything But Undoer

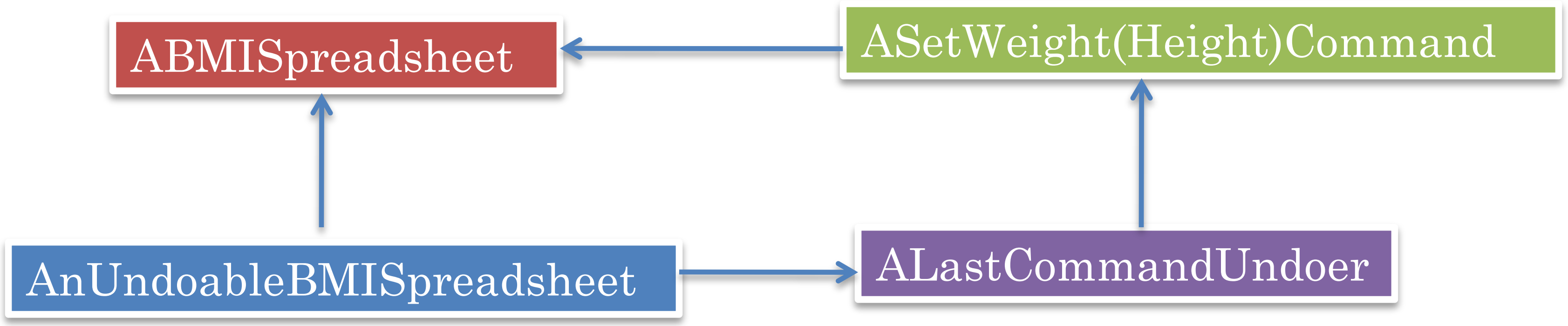


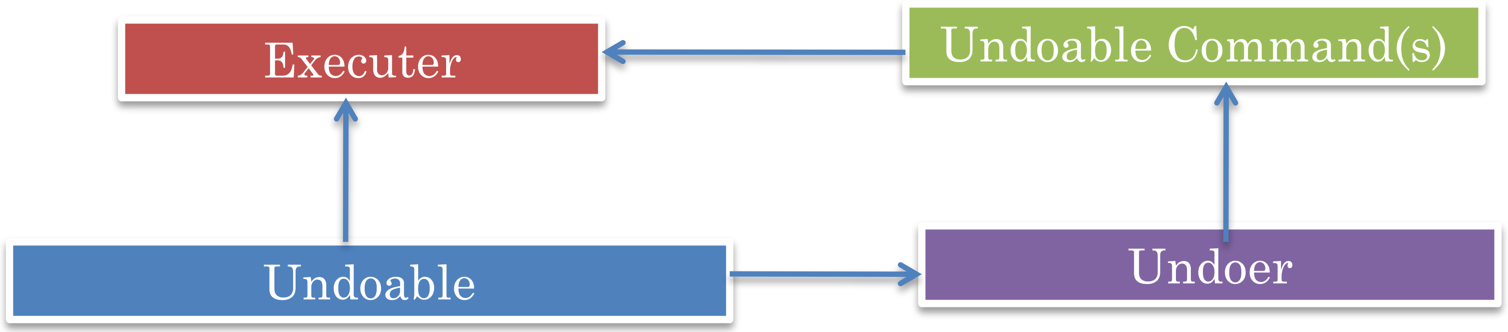
Global User Interface Undoer



All Objects in a UI share a single Undoer

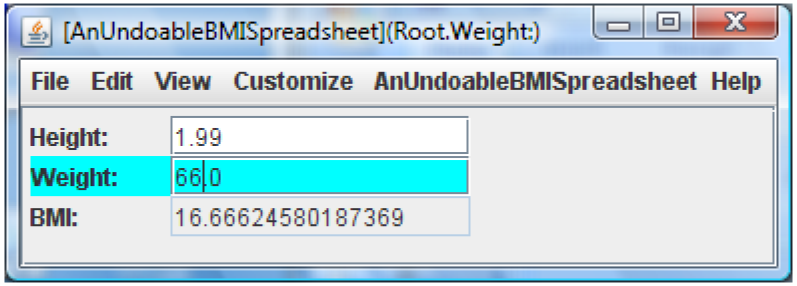
Undo Pattern





Executing an Undoable Method

```
        public void setWeight(double theWeight) {
                undoer.execute(new
ASetWeightCommand(bmiSpreadsheet, theWeight));
        }
```

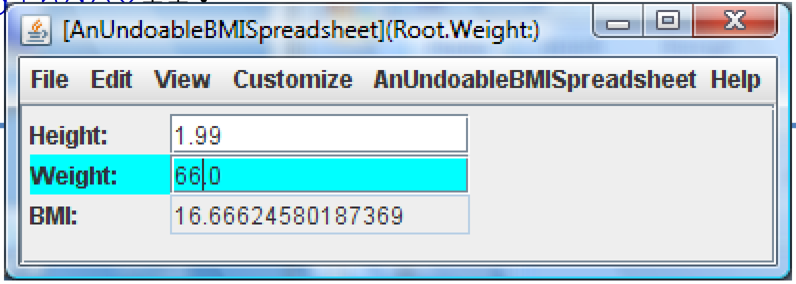

[AnUndoableBMISpreadsheet](Root.Weight:)
File Edit View Customize AnUndoableBMISpreadsheet Help

Height: 1.99
Weight: 66.0
BMI: 16.66624580187369

Delegating setWeight() creates command and asks undoer to execute it.

Executing an Undoable Method

```
public void execute (Command c) {
        if (nextCommandIndex != historyList.size()) {
           historyList.clear(); //ignore remaining undone commands
           nextCommandIndex = 0;
        }
        c.execute();
        historyList.add(c);
        nextCommandIndex++;
    }
```



[AnUndoableBMISpreadsheet](Root.Weight:)
File Edit View Customize AnUndoableBMISpreadsheet Help

Height: 1.99
Weight: 66.0
BMI: 16.66624580187369

Delegating setWeight() creates command and asks undoer to execute it.

Undoer calls execute() in command.

EXECUTING AN UNDOABLE METHOD

```
public class ASetWeightCommand implements Command {
BMISpreadsheet bmiSpreadsheet;
 double oldWeight;
 double weight;
 public ASetWeightCommand (BMISpreadsheet
theBMISpreadsheet, double theWeight) {
      bmiSpreadsheet = theBMISpreadsheet;
      weight = theWeight;
      oldWeight = bmiSpreadsheet.getWeight();
}
 public void execute() {bmiSpreadsheet.setWeight(weight);}
 public void undo() {bmiSpreadsheet.setWeight(oldWeight);}
}
```

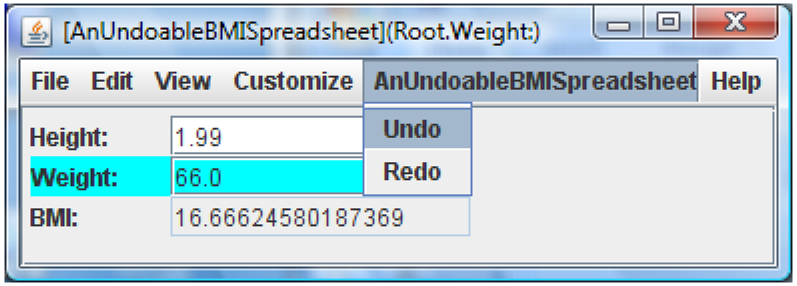
Delegating setWeight() creates command and asks undoer to execute it.

Undoer calls execute() in command.

Command asks delegate to invoke() setWeight() with constructor parameter value 66.0.

Undoable

```
public void undo() {undoer.undo();}
```

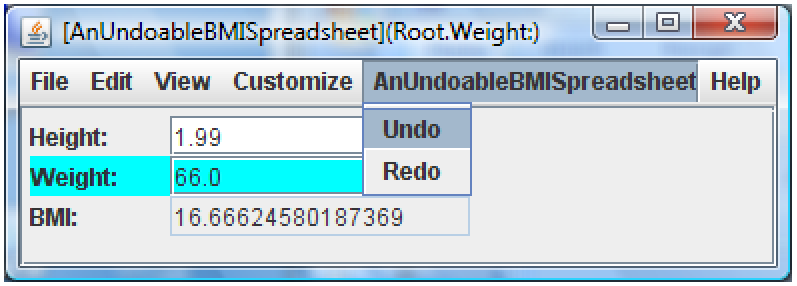


Height: 1.99
Weight: 66.0
BMI: 16.66624580187369

Delegating undo asks undoer to execute undo.

Undoer

```
public void undo() {
       if (nextCommandIndex == 0)return;
       nextCommandIndex--;
       Command c = historyList.get(nextCommandIndex);
       c.undo();
   }
```



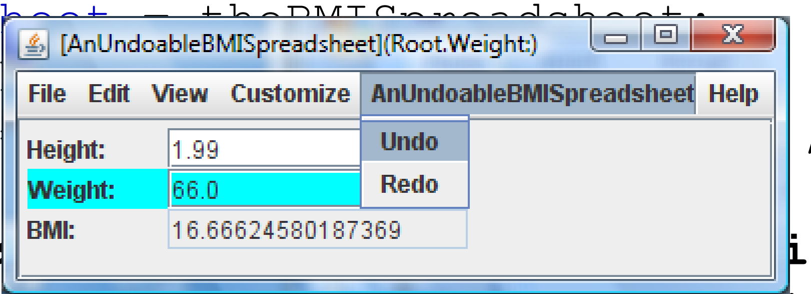
Height: 1.99
Weight: 66.0
BMI: 16.66624580187369

Delegating undo asks undoer to execute undo.

Undoer finds command object of last command and calls undo method of command object.

Undo Command

```
public class ASetWeightCommand implements Command {
BMISpreadsheet bmiSpreadsheet;
 double oldWeight;
 double weight;
 public ASetWeightCommand (BMISpreadsheet
theBMISpreadsheet, double theWeight) {
      bmiSpreads
      weight = t
      oldWeight                                        ;
}
 public void exec                                    ight(weight);}
 public void undo() {bmiSpreadsheet.setWeight(oldWeight);}
}
```


[AnUndoableBMISpreadsheet](Root.Weight:)
File Edit View Customize AnUndoableBMISpreadsheet Help
Undo
Redo

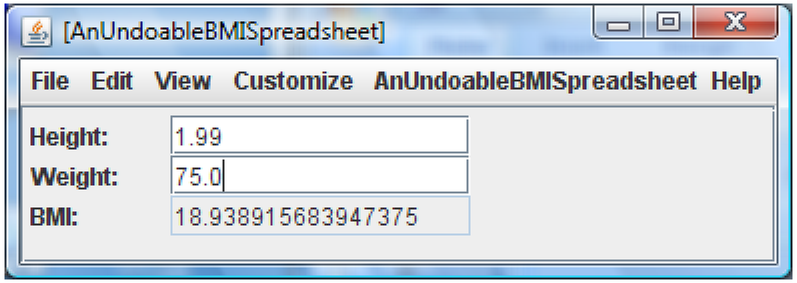
Height: 1.99
Weight: 66.0
BMI: 16.66624580187369

Delegating undo asks undoer to execute undo.

Undoer finds command object of last command and calls undo method of command object.

Undo method of command object calls setWeight() method of delegate with old value of weight: 75.0

Undo Effect

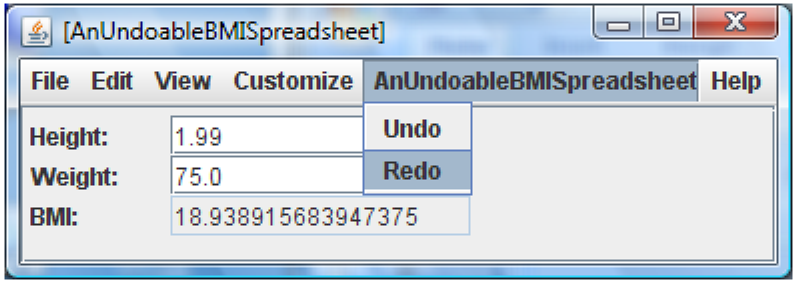

[AnUndoableBMISpreadsheet]
File Edit View Customize AnUndoableBMISpreadsheet Help

Height: 1.99
Weight: 75.0
BMI: 18.938915683947375

command undone.

Undoable

```
      public void redo() {undoer.redo();}
}
```

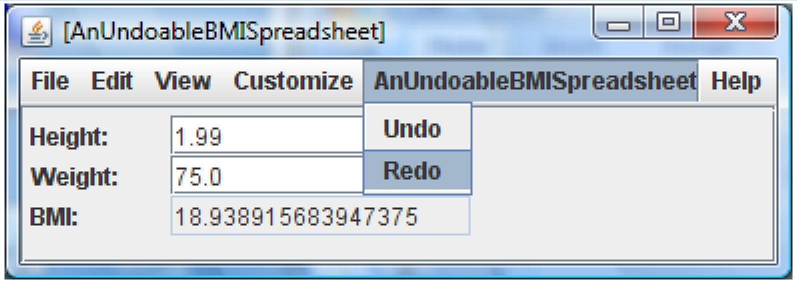


Height: 1.99
Weight: 75.0
BMI: 18.938915683947375

Delegating redo asks undoer to execute redo.

Undoer

```
public void redo() {
        if (nextCommandIndex == historyList.size()) return;
        Command c = historyList.get(nextCommandIndex);
        c.execute();
        nextCommandIndex++;
    }
```



Height: 1.99
Weight: 75.0
BMI: 18.938915683947375

Delegating redo asks undoer to execute redo.

Undoer finds last undone command object and calls execute method of command object.

Command

```
public class ASetWeightCommand implements Command {
BMISpreadsheet bmiSpreadsheet;
 double oldWeight;
 double weight;
 public ASetWeightCommand (BMISpreadsheet
theBMISpreadsheet, double theWeight) {
      bmiSpreadsheet = theBMISpreadsheet;
      weight = theWeight;
      oldWeight = bmiSpreadsheet.getWeight();
}
 public void execute() {bmiSpreadsheet.setWeight(weight);}
 public void undo() {bmiSpreadsheet.setWeight(oldWeight);}
}
```

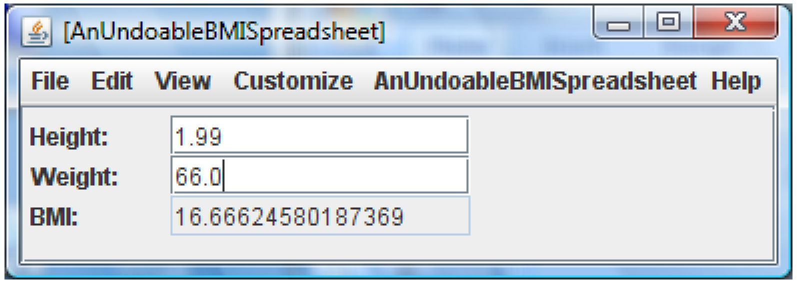
execute redo.

Undoer finds last undone command object and calls redo method of command object.

Execute method of same command object executed multiple times!

Redo method of command object calls setWeight() method of delegate with its constrictor parameter value : 66.0

Redo Effect


[AnUndoableBMISpreadsheet]

Height: 1.99

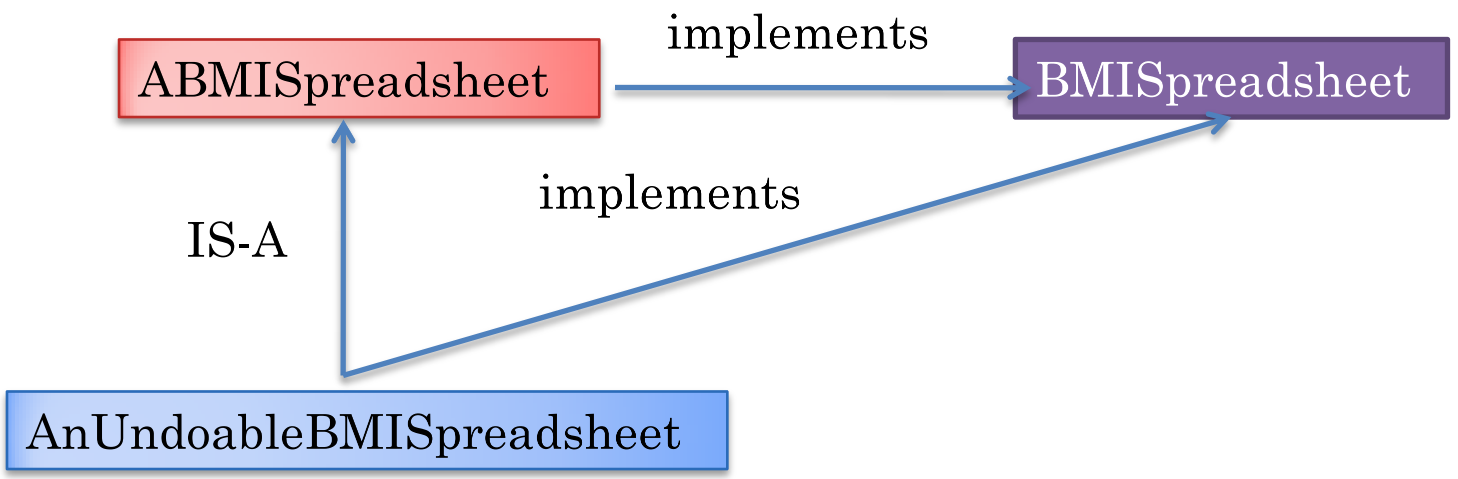
Weight: 66.0

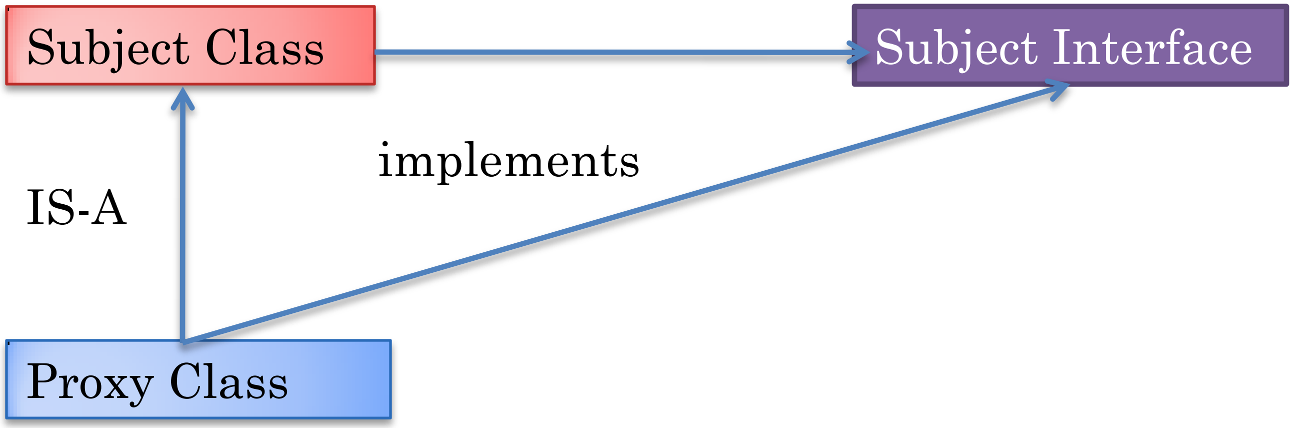
BMI: 16.66624580187369

Last undone command reexecuted

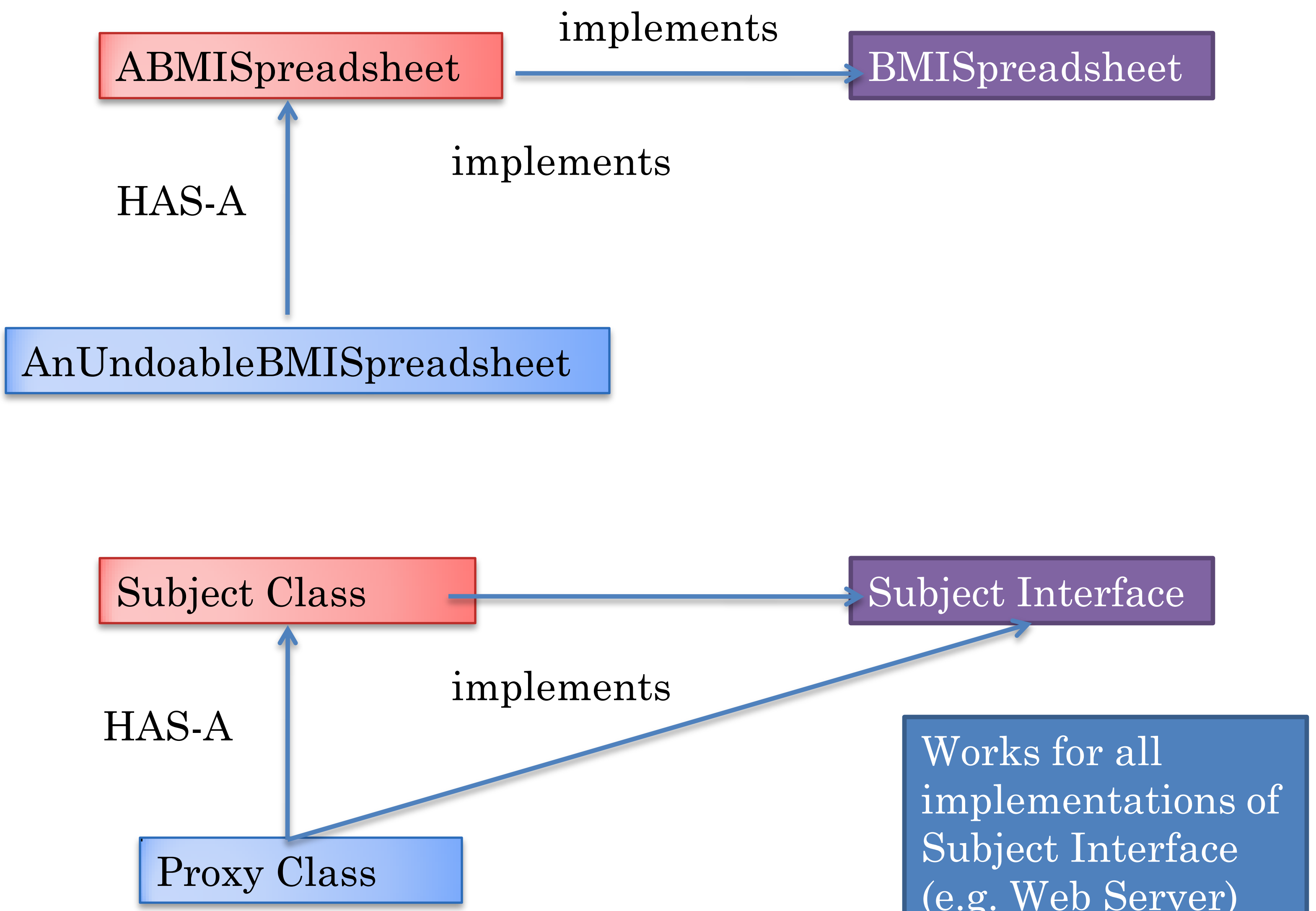
Extra Slides

Inheritance-based Proxy Pattern



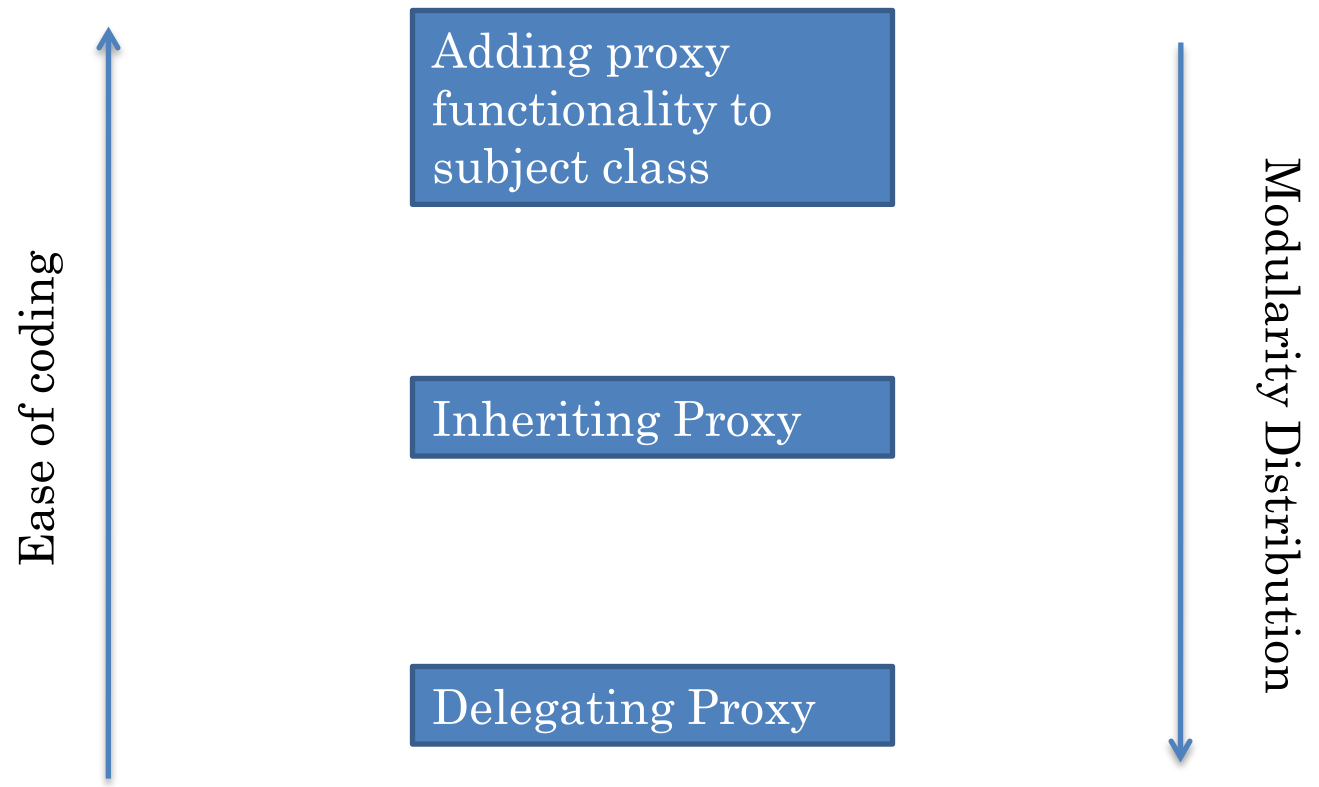


Delegation-based Proxy Pattern

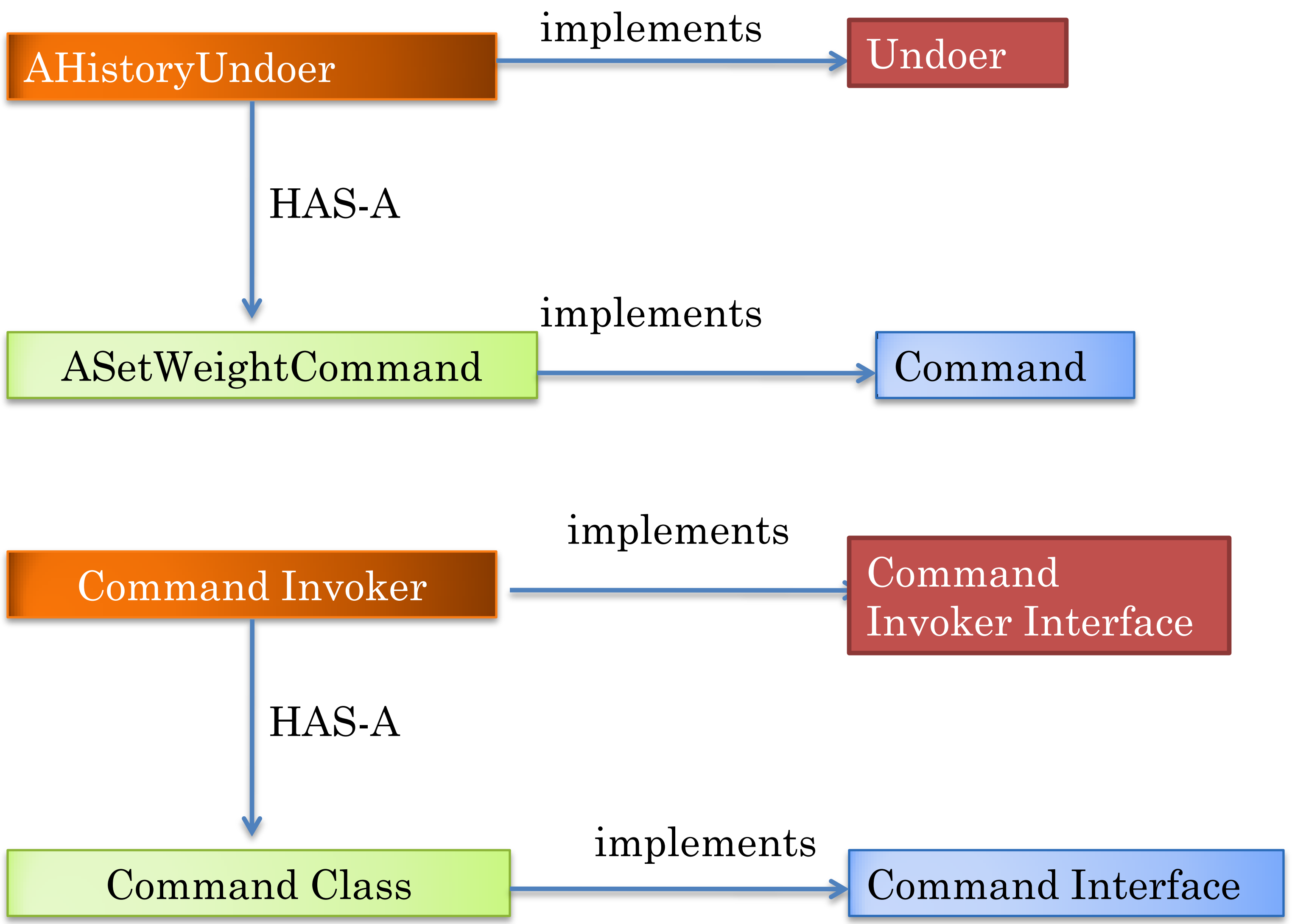


Works for all implementations of Subject Interface (e.g. Web Server)

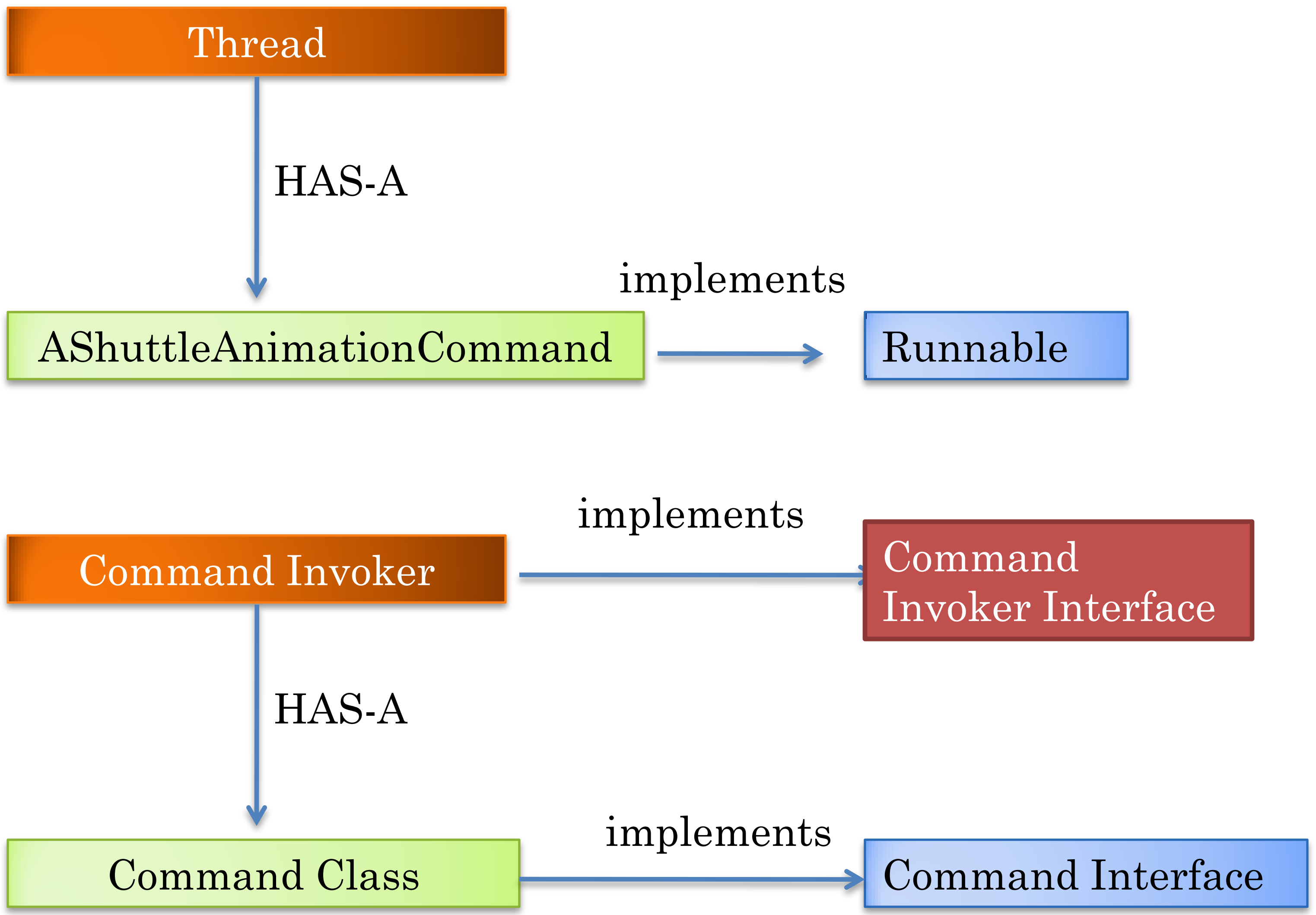
Inheritance-based Proxy Pattern



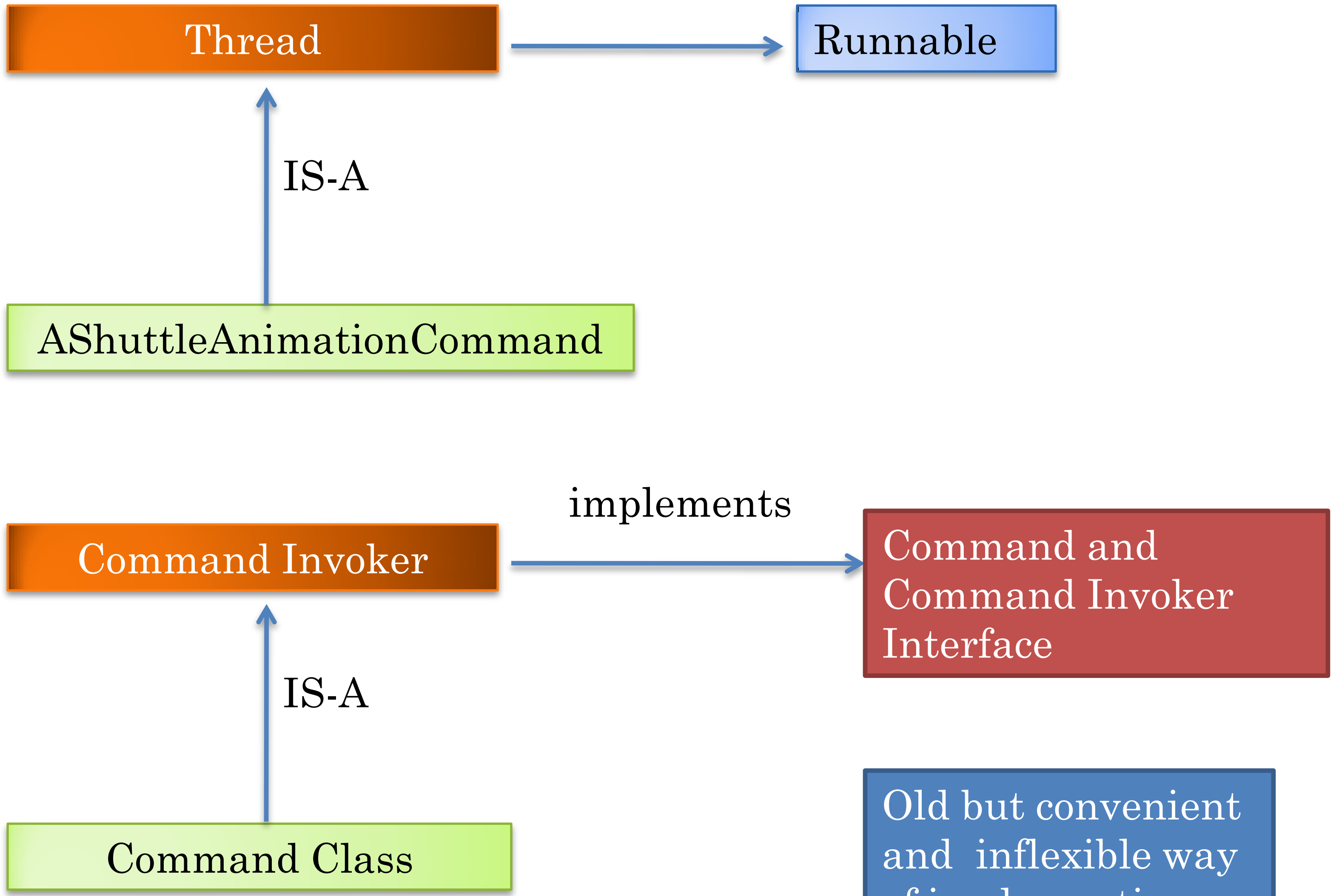
Delegation Based Commands



Delegation Based Commands



Inheritance Based Commands



Old but convenient and inflexible way of implementing threads

Extra Slides

Command Objects in Everyday Apps

- Thread: Runnables
- Undo/Redo: Undoable Command Object

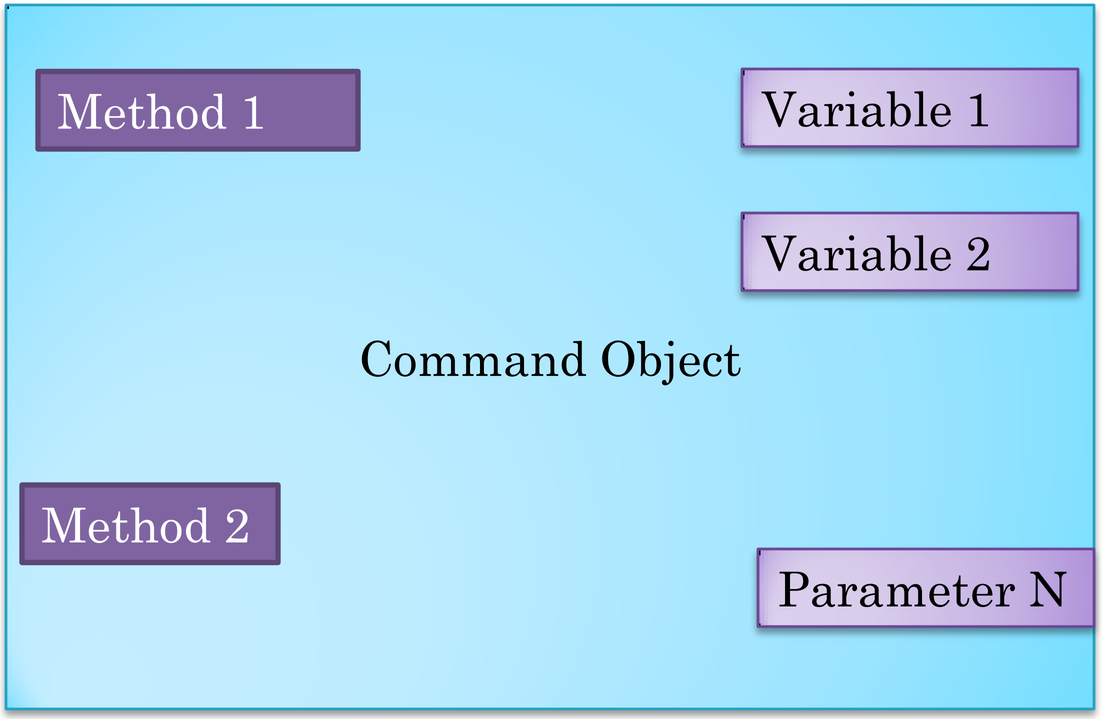
Animating vs. Updating Classes

- In general, a method that performs the animation steps and a method that changes the value of some animating property may be in different classes:
  - AnAnimatingShuttleLocation
  - ALabel

Method Parameters in Java

Java does not allow method parameters

Objects include methods and data



Command Object

When a method m1 wants to pass method m2 to method m3, it passedsa command object for the method m2

When method m3 wants to call m2 on object o, it passes

Parameterless method that calls method to be called.

Parameter 1

Parameter 2

Command Object

Parameter N

Constructor takes parameters of method to be called and target.

Object on which method is to be called

Method Parameters in Java

Java does not allow method parameters

Objects include methods and data

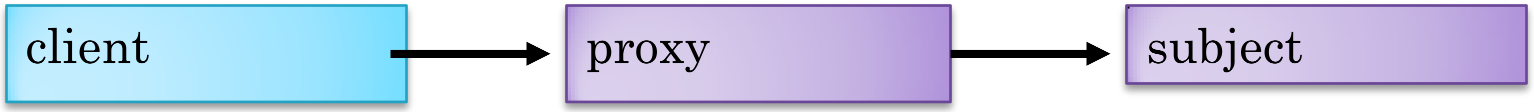
Instead of passing a method, pass a command object

Command object = method + parameters

Undoer BMISpreadsheet Separation

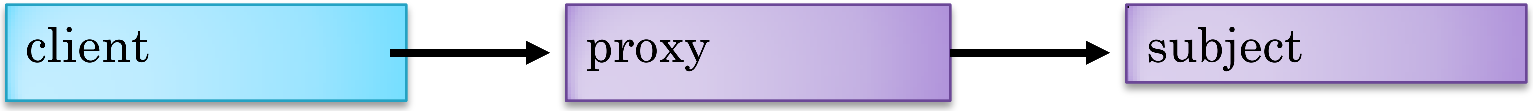
- Can use BMISpreadsheet with different undoer.
- Can use undoer with different object.
- Can use undoer with multiple objects in a single user interface.

PROXY



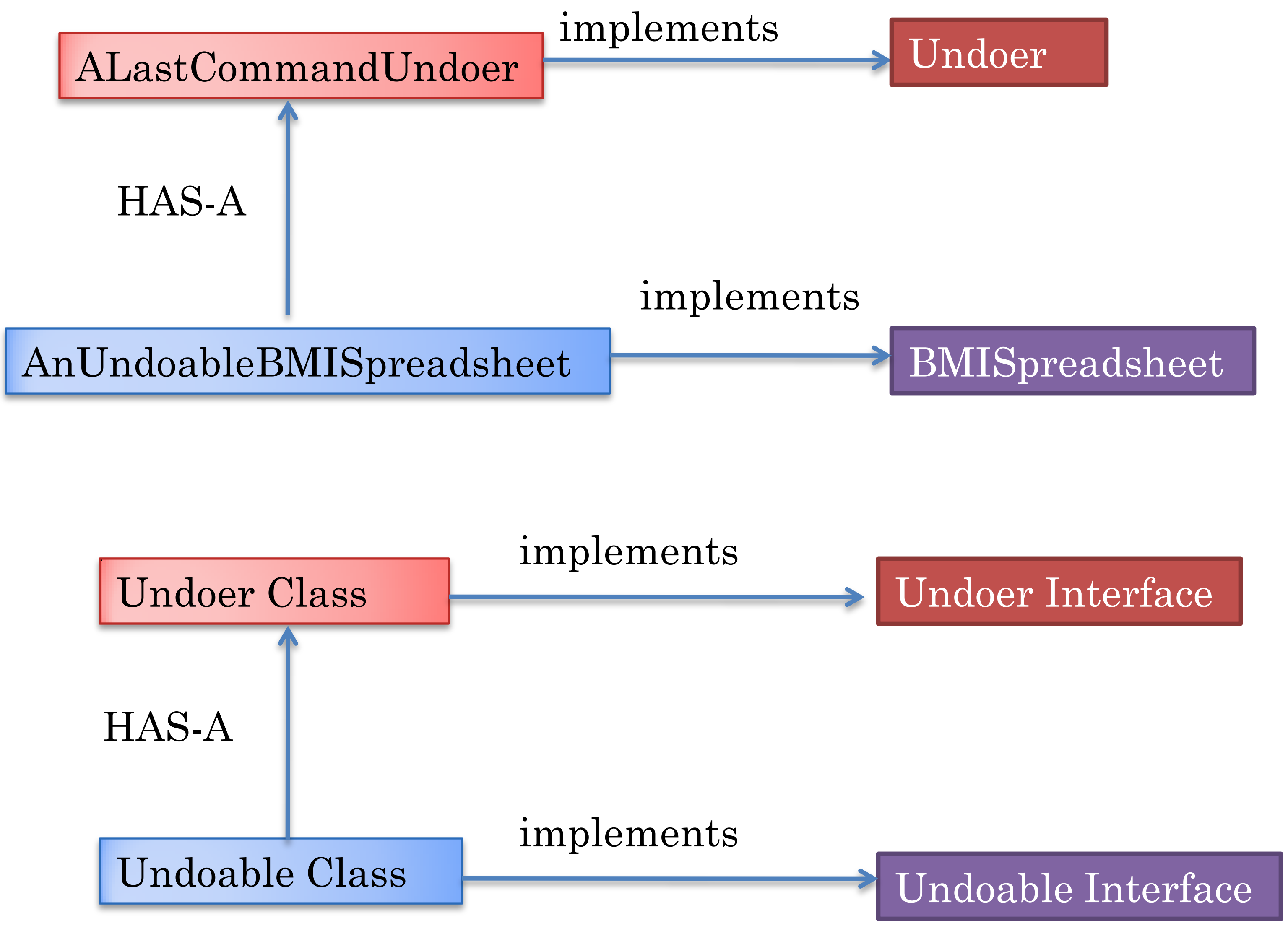
- Proxy is a class that sits between a client and subject class.
- Proxy is a stand-in for real subject.
- Methods called in subject through proxy methods,
- A proxy method does not change the behavior of subject method
- A proxy adds subject-independent functionality – which is independent of specific subject.
  - The interface of functionality is independent of subject interface (undo/redo)
  - The implementation may not be (required subject-specific commands)

PROXY

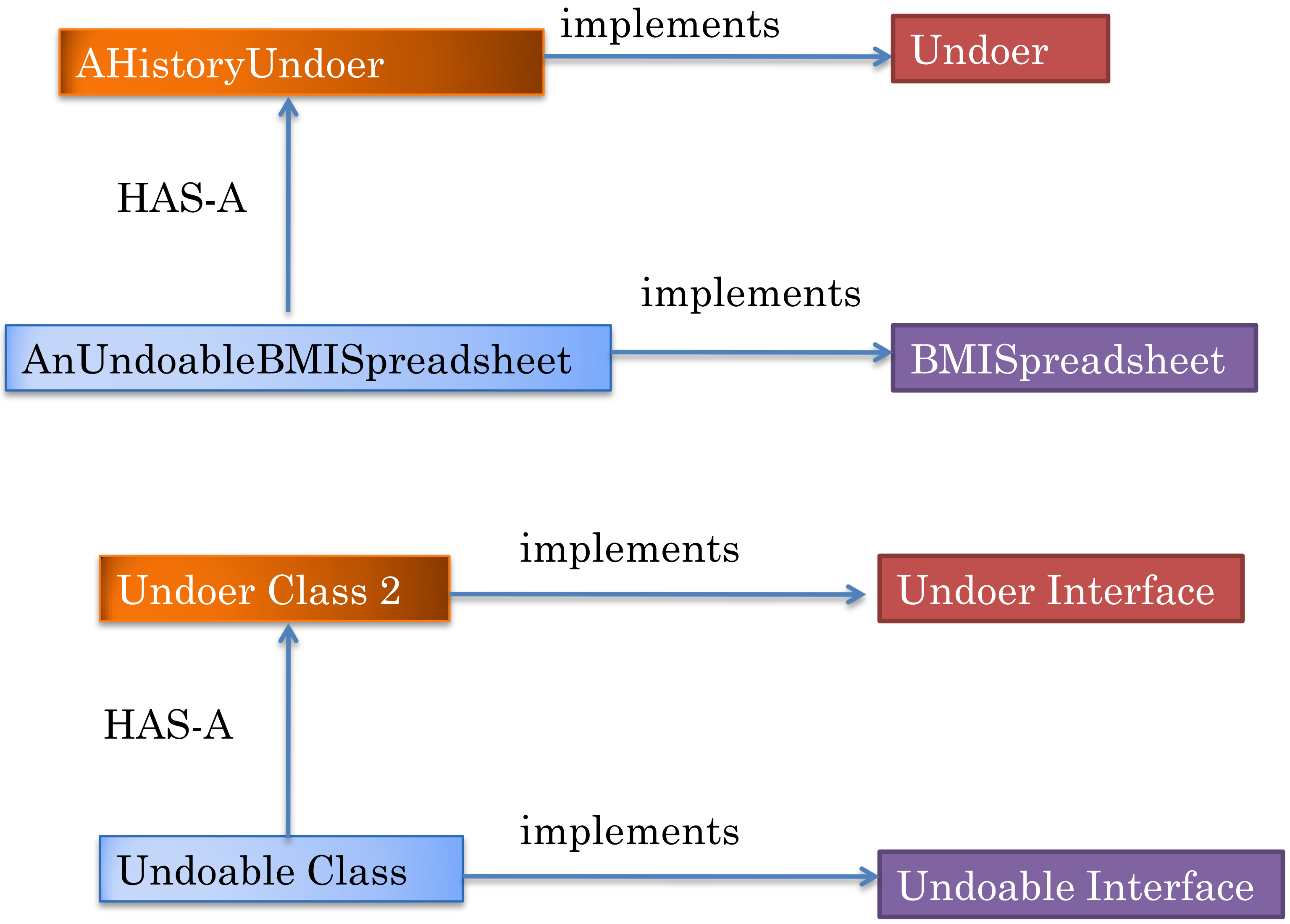


- Proxy is a class that sits between a client and subject class.
- Proxy is a stand-in for real subject.
- Methods called in subject through proxy methods,
- A proxy method does not change the behavior of subject method
- A proxy adds functionality – which is independent of specific subject.
  - The interface of functionality is independent of subject interface (undo/redo)
  - The implementation may not be (required subject-specific commands)

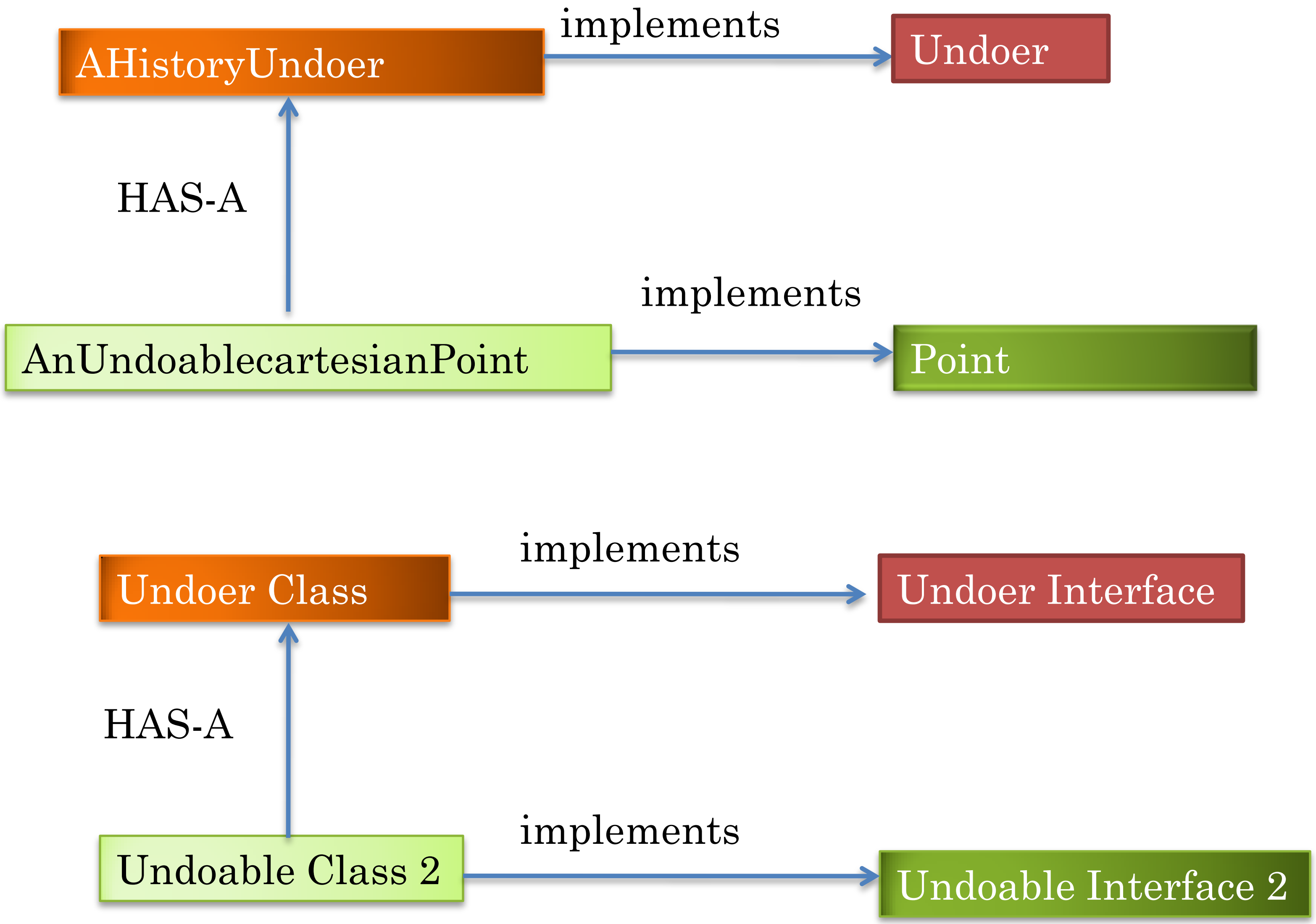
Undoer/Undoable Separation



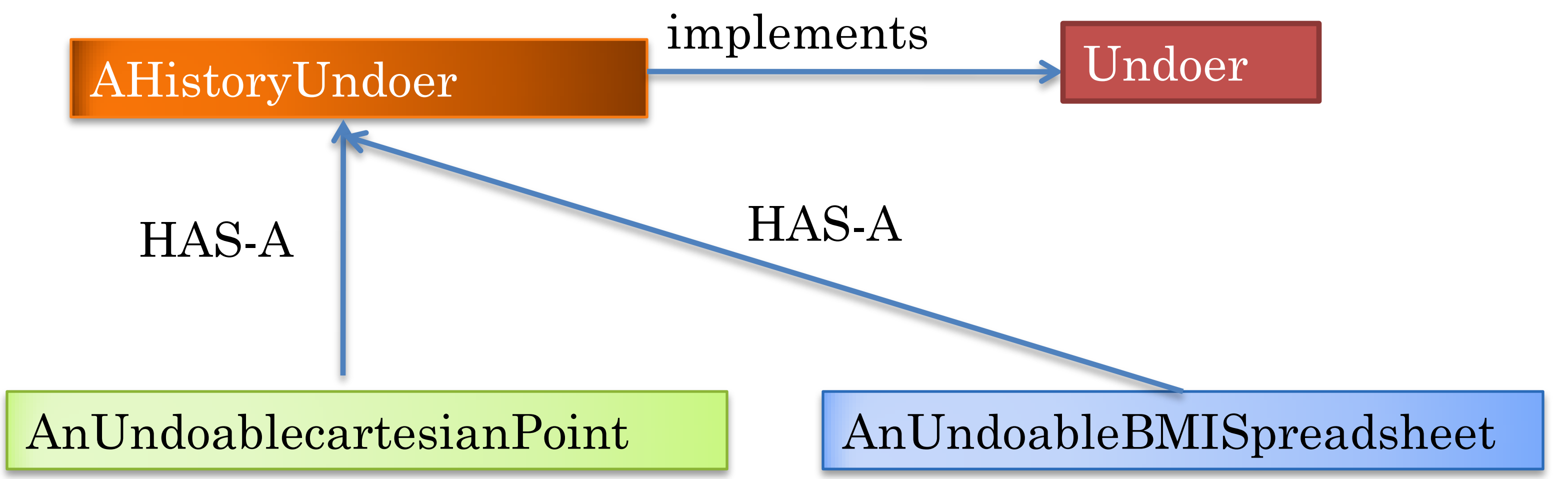
Changing the Undoer

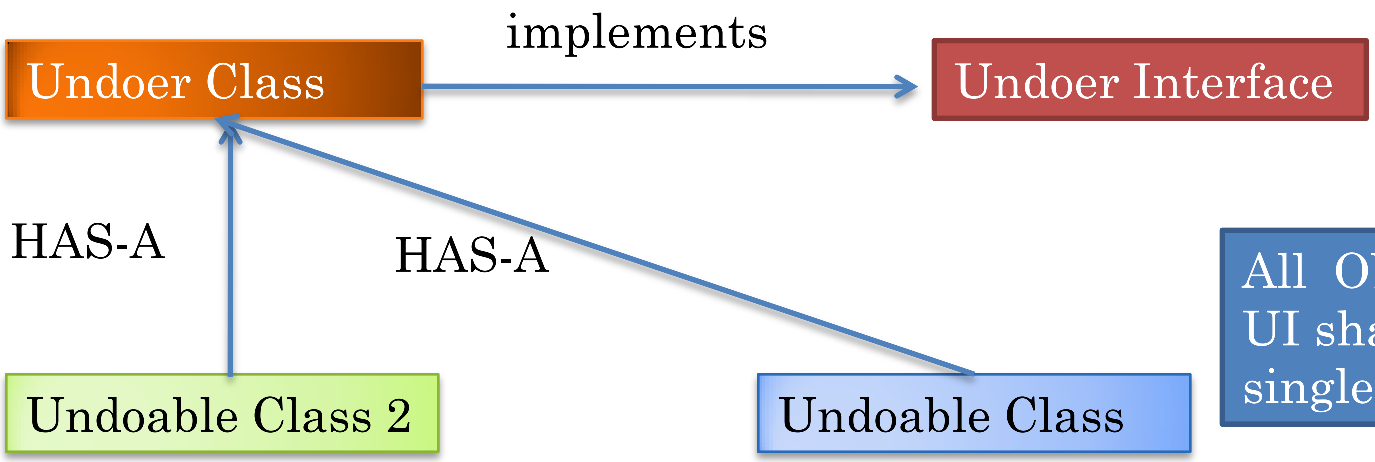


Changing the Undoablle



Multiple Undoables Per Undoer





All Objects in a UI share a single Undoer