AARHUS UNIVERSITET

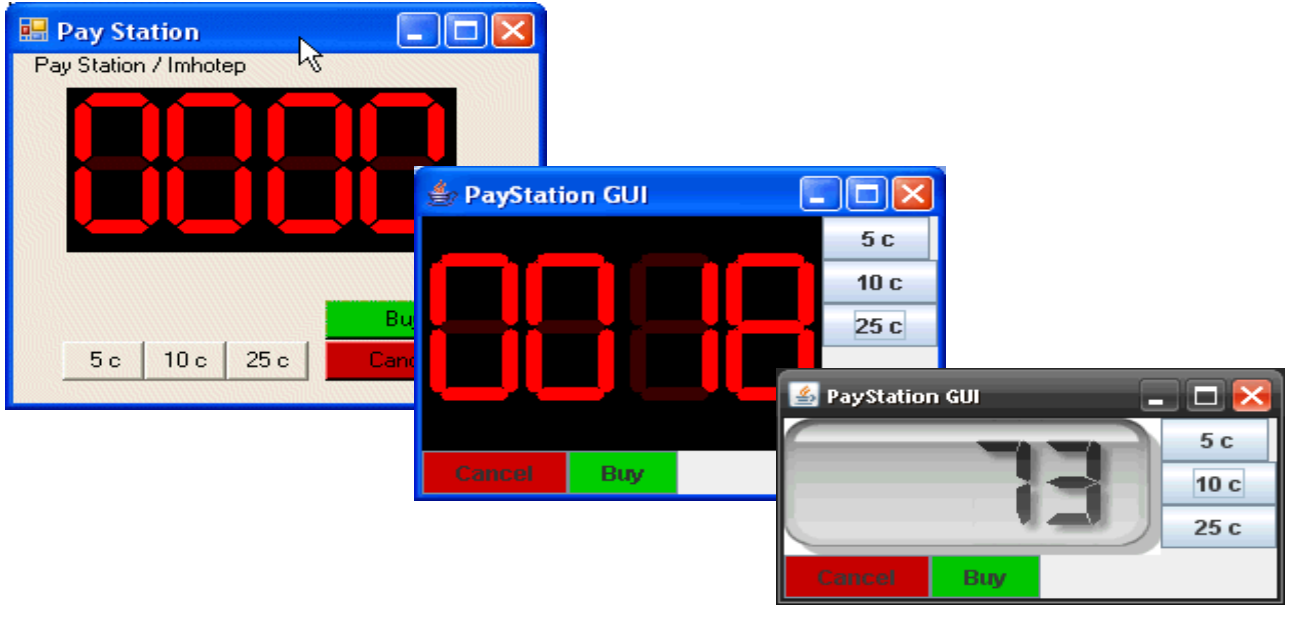
Software Engineering and Architecture

Pattern Catalog: Facade

AARHUS UNIVERSITET

New Requirement

- It would be nice with a simple GUI “to see something” instead of just xUnit tests...



“The GUI History of FRS” ☺

AARHUS UNIVERSITET

[Demo]

- Run it

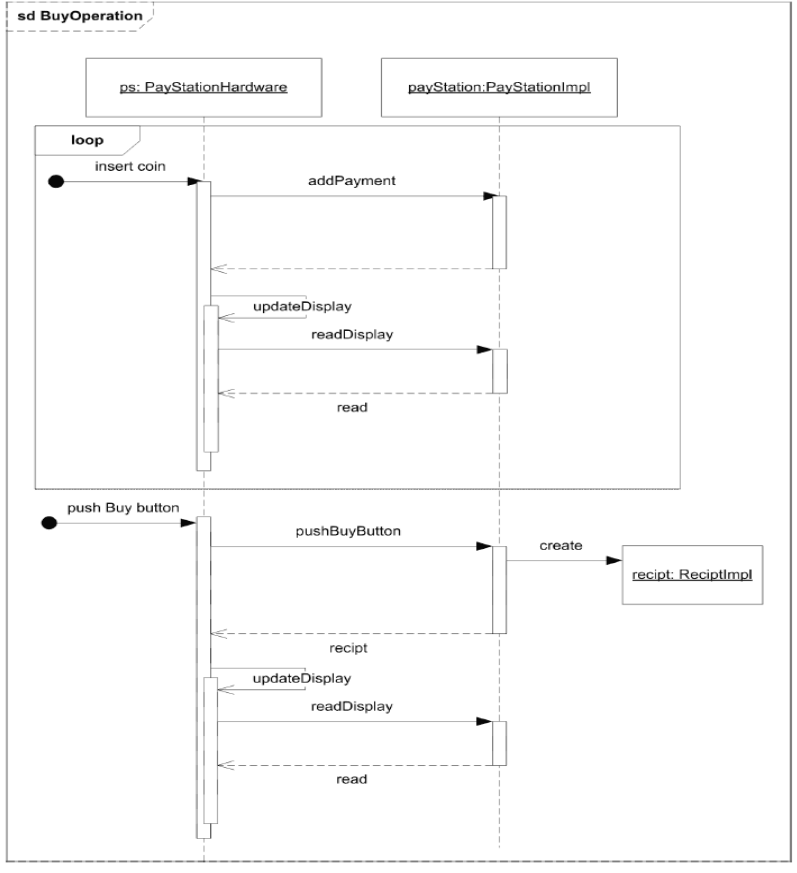
```
/** Create the panel of buttons */
private JComponent createButtonPanel() {
  Box p = new Box( BoxLayout.X_AXIS );
  JButton b;
  b = new JButton("Cancel");
  b.setAlignmentX(Component.CENTER_ALIGNMENT);
  p.add( Box.createHorizontalGlue() );
  p.add( b );
  b.addActionListener( new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        payStation.cancel();
        updateDisplay();
      } } );

  b = new JButton("Buy");
  b.setAlignmentX(Component.CENTER_ALIGNMENT);
  p.add( Box.createHorizontalGlue() );
  p.add( b );
  p.add( Box.createHorizontalGlue() );
  b.addActionListener( new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        Receipt r = payStation.buy();
        updateDisplay();
        // print the receipt
        showReceiptInWindow(r);
      } } );

  return p;
}
```

```
/** Update the digital display with whatever the
    pay station domain shows */
private void updateDisplay() {
  String prefixedZeros =
    String.format("%4d", payStation.readDisplay() );
  display.set( prefixedZeros );
}

/** Create the coin input panel */
private JComponent createCoinInputPanel() {
  Box p = new Box( BoxLayout.Y_AXIS );
  p.add( defineButton( " 5 c", "5" ));
  p.add( defineButton( "10 c", "10" ));
  p.add( defineButton( "25 c", "25" ));
  return p;
}

/** The button action listener that reacts on clicking the
    coin buttons */
private ActionListener buttonActionListener = new ActionListener() {
    public void actionPerformed(ActionEvent e) {
      String s = e.getActionCommand();
      int coin = Integer.parseInt(s);
      try {
        payStation.addPayment( coin );
      } catch (IllegalCoinException exc) {
        // illegal coins just do nothing.
      }
      updateDisplay();
    }
  };
```

AARHUS UNIVERSITET

Analysis

- Seq Diagram
  - No difference in behaviour of a GUI versus real hardware!



AARHUS UNIVERSITET

Conclusion

- *Any* kind of user interface can operate the PayStation!
- Wow – *Change by addition...*

- How come we are so lucky?

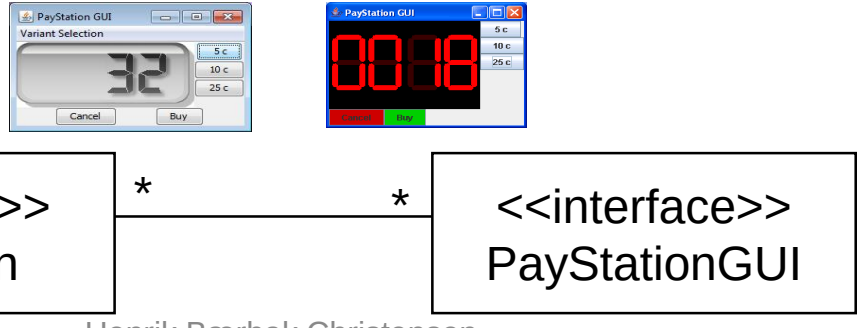
AARHUS UNIVERSITET

Design considerations

- (3) Behaviour that may vary
  - the *same* hardware must operate *varying* pay station implementations: AlphaTown, BetaTown, EpsilonTown...
- (1) Variable behaviour behind *interface*
  - **PayStation** interface...
- (2) Compose behaviour by delegation
  - Gui/Hardware does not itself calculate rates, issue receipts, etc., but *lets an instance of PayStation do the dirty job...*

Result

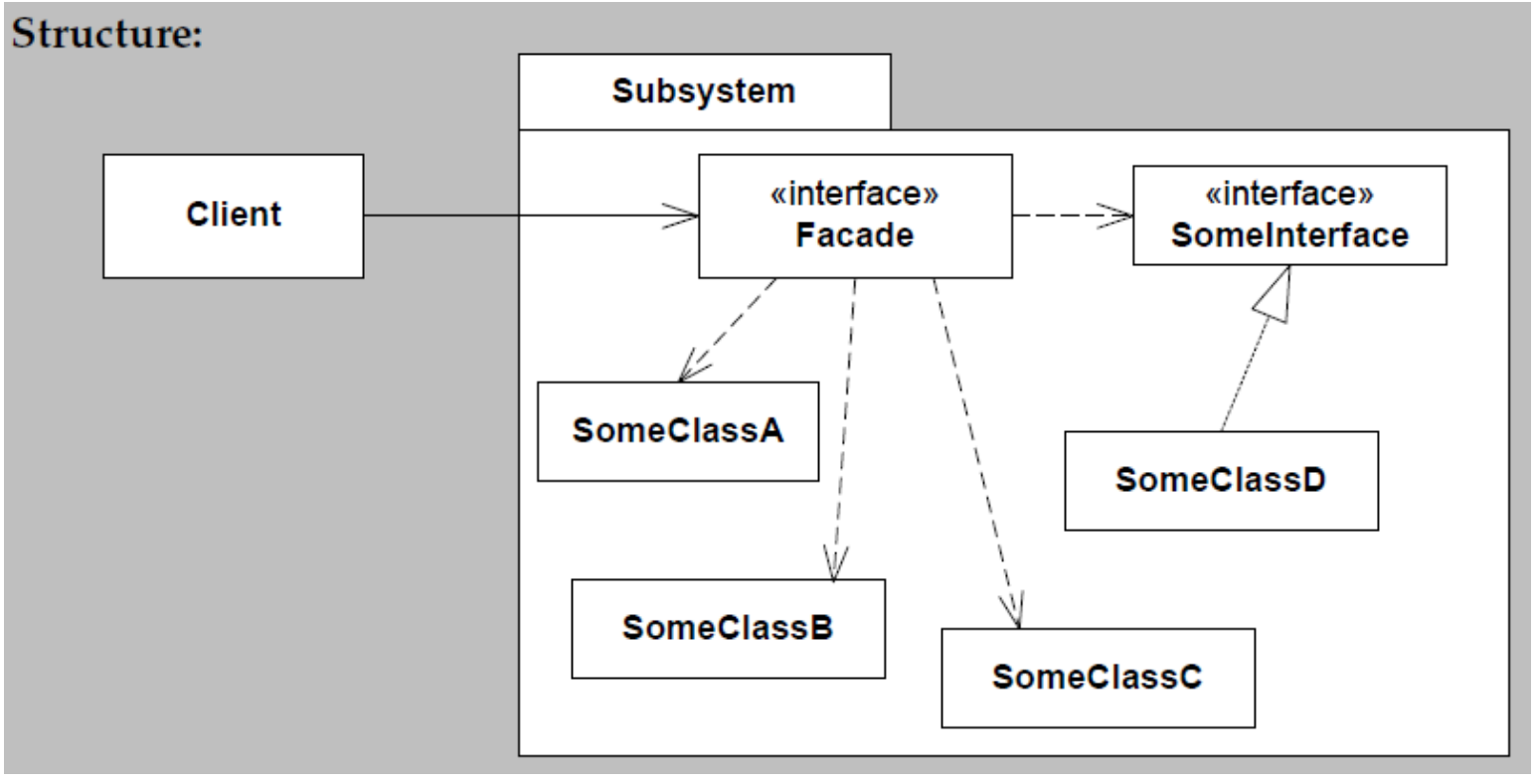
- The side effect of this decision is that *interface decouples both ways!!!*
  - *Hardware may operate different kinds of PayStation implementations*
    - *Alpha, Beta, Gamma, …*
  - *Different kinds of user interfaces may operate the same PayStation implementation*



AARHUS UNIVERSITET

Automagical pattern?

- PayStation is an example of the **Facade** pattern



AARHUS UNIVERSITET

Consequences

- Benefits
  - Shields clients from subsystem objects
    - (depends… Consider HotCiv)
  - Weak coupling
    - **Many to many** relation between client and façade
- Liabilities
  - Bloated interface with *lots of methods*
    - Because façade must have the sum of responsibilities of the subsystem
  - How to avoid access to the inner objects?
    - Read-only interfaces; no access (require dumb data objects to be passed and parsed over the facade).