USE-RELATIONSHIP AND CLASS HIERARCHY DESIGN

FSM for Window Operations: \{Open, close, lock, unlock\}.

Use-relationship:
- The parameter key in lock(key) and unlock(key) must match the value origKey for these operations to succeed.
- Two other operations: ExtOpen(key) means unlock first and then open; similarly for ExtLock(key).

\[ V(v_1) = \{\text{open, closed}\}, \quad V(v_2) = \{\text{locked, unlocked}\} \]

<table>
<thead>
<tr>
<th>Variable-use relationship</th>
<th>( f_1 ): Open</th>
<th>( f_2 ): Close</th>
<th>( f_3 ): Lock</th>
<th>( f_4 ): Unlock</th>
<th>( f_5 ): Ext-Open</th>
<th>( f_6 ): Ext-Lock</th>
</tr>
</thead>
<tbody>
<tr>
<td>( v_1 ): open-ClosedStatus</td>
<td>( x )</td>
<td>( x )</td>
<td>( x )</td>
<td>( x )</td>
<td>( x )</td>
<td>( x )</td>
</tr>
<tr>
<td>( v_2 ): locked-Status</td>
<td>( x )</td>
<td>( x )</td>
<td>( x )</td>
<td>( x )</td>
<td>( x )</td>
<td>( x )</td>
</tr>
<tr>
<td>( v_3 ): origKey</td>
<td>( x )</td>
<td>( x )</td>
<td>( x )</td>
<td>( x )</td>
<td>( x )</td>
<td>( x )</td>
</tr>
</tbody>
</table>

Question: How can you make the use-relationship more informative?
**A NOT-SO-GOOD CLASS HIERARCHY FROM USE-RELATIONSHIP**

<table>
<thead>
<tr>
<th>Variable-use relationship</th>
<th>$f_1$: Open</th>
<th>$f_2$: Close</th>
<th>$f_3$: Lock</th>
<th>$f_4$: Unlock</th>
<th>$f_5$: Ext-Open</th>
<th>$f_6$: Ext-Lock</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_1$: open-ClosedStatus</td>
<td>$rw$</td>
<td>$rw$</td>
<td>$r$</td>
<td></td>
<td>$rw$</td>
<td>$rw$</td>
</tr>
<tr>
<td>$v_2$: locked-Status</td>
<td>$r$</td>
<td>$rw$</td>
<td>$rw$</td>
<td></td>
<td>$rw$</td>
<td>$rw$</td>
</tr>
<tr>
<td>$v_3$: origKey</td>
<td>$r$</td>
<td>$r$</td>
<td></td>
<td></td>
<td>$r$</td>
<td>$r$</td>
</tr>
</tbody>
</table>

The dashed line shows a reference to $f_{LockedStatus}$.

```
abstract CLockedStatus
  prot fLockedStatus
  abstract CUnlock
  prot const fOrigKey
  publ Unlock(key)

abstract COpenClosedStatus
  prot fOpenClosedStatus
  publ Close()

abstract COpen
  const ref CLockedStatus::fLockedStatus
  publ Open()

CDoorWithLock
  publ Lock(key)
```

The dashed line shows a reference to $f_{LockedStatus}$. 
THE MEMBER FUNCTIONS

bool COpenClosedStatus::Close()
{ if (open == fOpenClosedStatus) {
    fOpenClosedStatus = closed;
    return(true);
} else return(false);
}

bool COpen::Open()
{ if ((closed == fOpenClosedStatus) &&
    (unlocked == fLockedStatus)) {
    fOpenClosedStatus = open;
    return(true);
} else return(false);
}

bool CUnlock::Unlock(key)
{ if ((key == fOrigKey) &&
    (locked == fLockedStatus)) {
    fLockedStatus = unlocked;
    return(true);
} else return(false);
}

bool CDoorWithLock::Lock(key)
{ if ((key == fOrigKey) &&
    (unlocked == fLockedStatus) &&
    (closed == fOpenClosedClosedStatus)) {
    fLockedStatus = locked;
    return(true);
} else return(false);
AN IMPROVEMENT BY REFACTORING OF OPEN-FUNCTION

- Refactoring of Open-function eliminates the reference outside the class-subclass hierarchy.

```cpp
bool CSubOpen::SubOpen()
{ if (closed == fOpenClosedStatus)
   { fOpenClosedStatus = open; return(true); } 
   else return(false); }

bool CDoorWithLock::Open()
{ if (unlocked == fLockedStatus)
   return(SubOpen());
   else return(false); }
```
WHICH ONE IS THE BEST

After merging each abstract class with its single which has a single parent.

Final form using a factoring of Lock(key).
How good are these two class-hierarchy designs by piecewise approach:

(i) Doors with a lock as a subclass of doors.

(ii) Result of combining fOpenClosedStatus and fLockedStatus.

Question:

• Why we should not merge \( v_1 = \text{openClosedStatus} \) and \( v_2 = \text{lockedStatus} \) into a single variable?