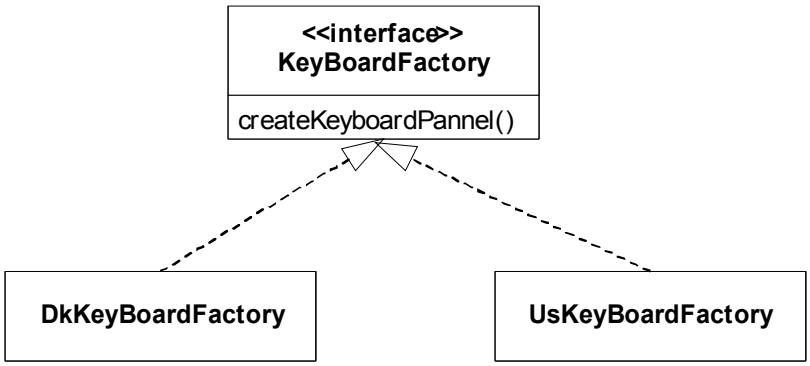
Exercise 9.3

Problems encountered

The GUI setup (makeButton and keyCodeListener) are hardcoded in the dooralarm.gui

3-1-2 process

3. Identify behavior likely to change
   - Button-layout on GUI.
1. State responsibility that covers behavior likely to change
   <<interface>>
   KeyPanelFactory
   - createKeyboardPanel (return instance of JPanel with implementation specific layout)
2. Refactor - implement KeyBoardFactory



Iteration outline

Iteration 1: Introduce KeyBoardFactory

5. Introduce KeyBoardFactory
6. (ant dk) Test case succeeded (manual inspection of GUI)

Iteration 2: Implement DkKeyBoardFactory

3. Obvious implementation - Create createKeyboardPanel with same implementation as existing.
4. (ant dk) Test case succeeded (manual inspection of GUI)

Iteration 3: Use DkKeyBoardFactory

5. Refactor - Replace hard-coded createKeyboardPanel with KeyBoardFactory
6. Test case failed (Java crash)
7. Create KeyBoardFactory constructor argument to DoorAlarm and use DkKeyBoardFactory
8. (ant dk) Test case succeeded (manual inspection of GUI)

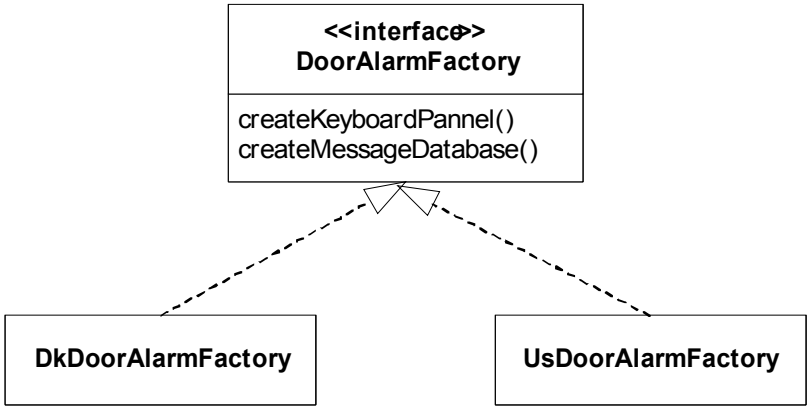
Iteration 4: Implement UsKeyBoardFactory

1. Copy DkKeyBoardFactory to UsKeyBoardFactory and adjust.
2. (ant us) Test case failed (manual inspection of GUI - no change = error)
3. Triangulation - use command line argument to choose Factory implmentation
4. (ant us) Test case succeeded (manual inspection of GUI)
5. (ant dk) Test case succeeded (manual inspection of GUI)
6. Refactor - Create base class with common functionality.

Interface-level triangulation

Second 3 - 1 - 2 process

3. Identify behavior likely to change
   - Already defined in MessageDatabase (language specific text)
1. State responsibility that covers behaviour likely to change
   - Already defined in MessageDatabase
   <<interface>>
   DoorAlarmFactory
   - createMessageDatabase (return instance of MessageDatabase with implementation of specific languages)
2. Refactor - implement DoorAlarmFactory from KeyBoardFactory



Iteration outline

Iteration 5: Refactor Factory responsibility

1. Refactory - Rename factory to DoorAlarmFactory
4. (ant us) Test case succeeded (manual inspection of GUI)
5. (ant dk) Test case succeeded (manual inspection of GUI)

Iteration 6: Update Factory to handle MessageDatabase

2. Create method createMessageDatabase() on all factories (fake-it - return NULL)
4. (ant us) Test case succeeded (manual inspection of GUI)
5. (ant dk) Test case succeeded (manual inspection of GUI)

Iteration 7: Use createMessageDatabase

1. Update DoorAlarm to create MessageDatabase from factory (requires using the setTitle method of JFrame rather than super(<title>)
2. Test case failed (Java crash) - no implementation of factory
3. Implement DK factory (obvious implementation)
5. (ant dk) Test case succeeded (manual inspection of GUI)

Iteration 8: Create UsMessageDatabase

1. Create UsMessageDatabase
2. (ant us) Test case failed (java crash)
3. Update factory to return UsMessageDatabase
4. (ant us) Test case succeeded (manual inspection of GUI)

Other possibility (Solution 2)

Alternative 3 - 1 - 2 process

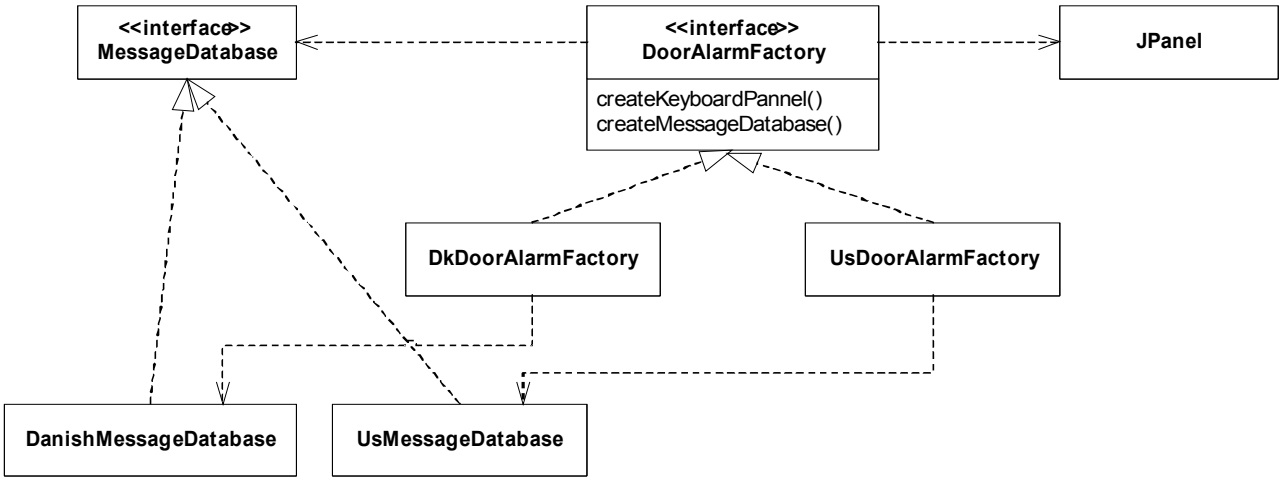
3. Identify behaviour likely to change
   - The text on the JPanel
1. State responsibility that covers behaviour likely to change
   - Already defined in MessageDatabase
   <<interface>>
   DoorAlarmFactory
   - createKeyboardPanel (return array of strings to apply to panel)
2. Refactor - Update DoorAlarmFactory

Iteration outline

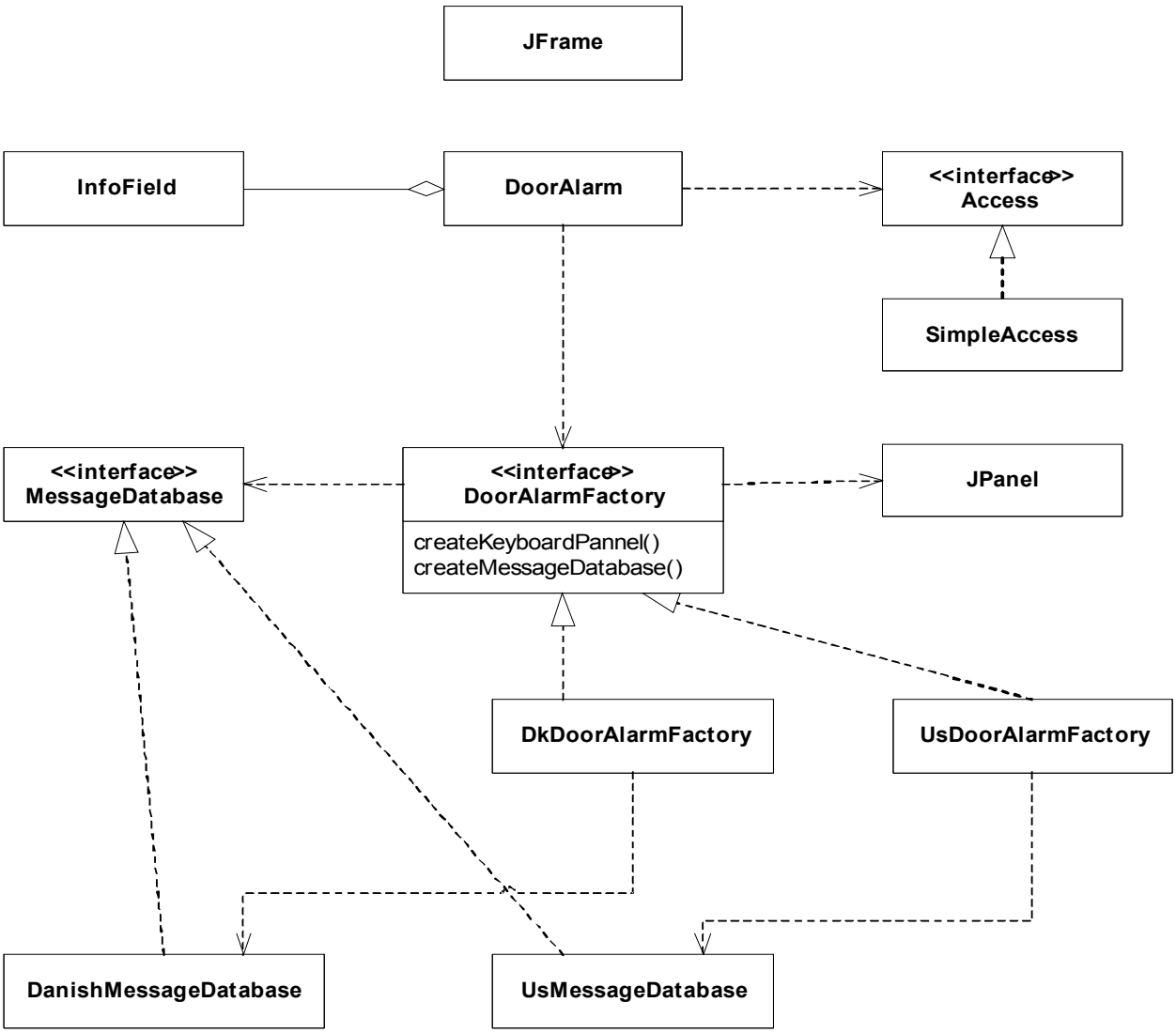
Iteration 9: Change factory return value to String[]

1. Change factory to return String[] array instead of JPanel
2. Refactor code by reintroducing old code and use string array
3. (ant dk) Test case succeeded (manual inspection of GUI)

Final design of Factory



Final design



Conclusion

The two solutions differ mainly in defining what may vary.
In the first solution, where the factory create a JPanel, it is possible to alter the entire Panel, meaning that the US may have different buttons, colours, size, etc.
In solution two this is not possible, as it is known that only the text on the buttons may change.
If it is known that the size, font, etc. that is used will never change,
e.g. due to physical restrictions and architectural decisions,
it is risky to allow the factory to change the layout of the panel, as if it is possible,
some programmer at some point in time, will attempt to do it.
Which of the two solutions is better is difficult to say as it depends on what may change in the future.