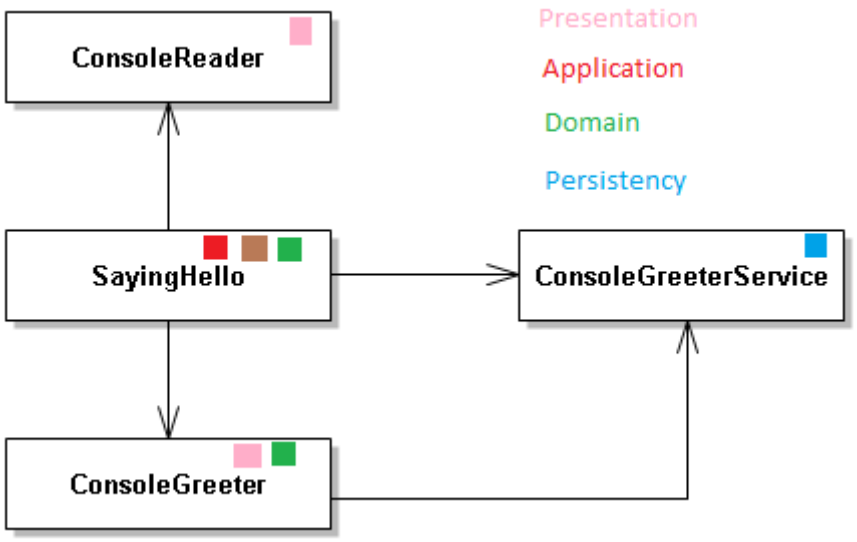
SayingHello - Mark1



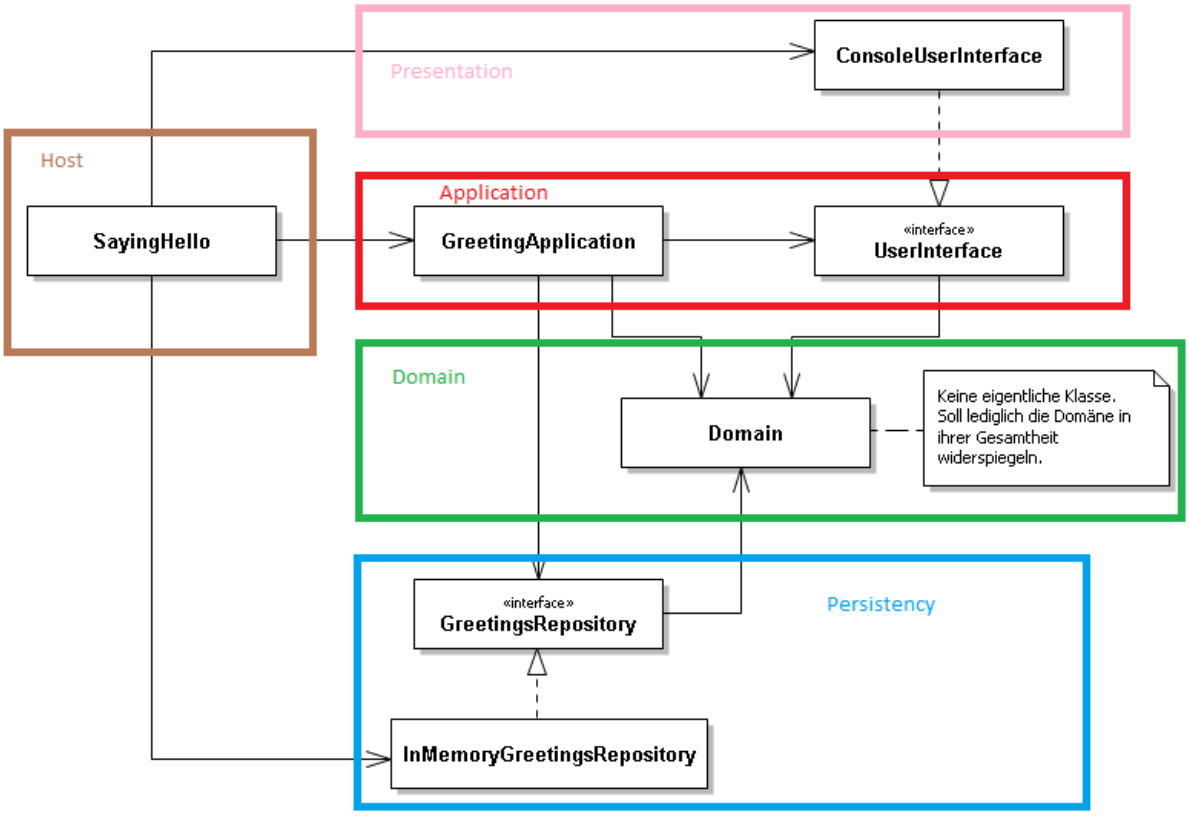
Vorteile:

Die Implementierung ist übersichtlich, schlicht und für die Aufgabenstellung angemessen. Allerdings nur unter der Voraussetzung, dass die Software nicht mehr gewartet und erweitert wird!!

Nachteile:

Durch die statischen Abhängigkeiten entsteht ein rigides Gesamtsystem, dass sich nicht frei beliebig ändern lässt. Außerdem lassen sich einzelne Bestandteile durch die statischen Abhängigkeiten nur schlecht isoliert testen. Jede Änderung hat unter Umständen zur Folge, dass das gesamte System eingefasst werden muss. Mehr Anpassungen => höhere Bugwahrscheinlichkeit!

SayingHello - Mark2



Vorteile:

- Das gesamte System ist lose gekoppelt, damit kann jeder Bereich gezielt angepasst, ausgetauscht oder getestet werden.
- Klare Auftrennung der Logik in ein Schichten-System
- Extrem hohe Codewiederverwendbarkeit durch die lose Kopplung!!
- SOLID, DRY
- Unterstütze Veränderungsachsen:
    - Andere Persistierungstechnology (z.B. Dateibasiert und Datenbank)
    - Anderer Anwendungskontext (z.B. WebService, der serverseitig HTML-Ansichten generiert)
    - Austauschen/Erweiter der Applikationslogik
    - Neue GUI-Technologie (z.B. JavaFX)

Nachteile:

- Höherer initialer Umsetzungsaufwand
- Erhöhte Komplexität, durch die unterstützen Veränderungsachsen.

Fazit

Software ist stehts in einem konstanten Flux und kein System ist wirklich fertig, solange es supported wird. Daher gilt für mich immer die Faustregel, je entkoppelter ein System ist, desto anpassungsfähiger ist es. Allerdings darf man dabei nicht außer Acht lassen, dass mit jeder eingeführten Flexibilität die Komplexität steigt! Die Kunst liegt letztendlich darin eine Balance zwischen Übersichtlichkeit und Flexibilität zu finden.