Kontrollfragen A

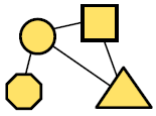
1. Welche Disziplin bei der Software-Erstellung beinhaltet das Klassen-Design?
   Realisierung

2. Was kann schlechtes Klassen-Design zur Folge haben?
   - *langsame Entwicklung*
   - *schwierige Entwicklung*
   - *fehlerhafte Entwicklung*
   - *abgebrochene Entwicklung*
   - *nachträgliche Änderungen schwierig umsetzbar!*

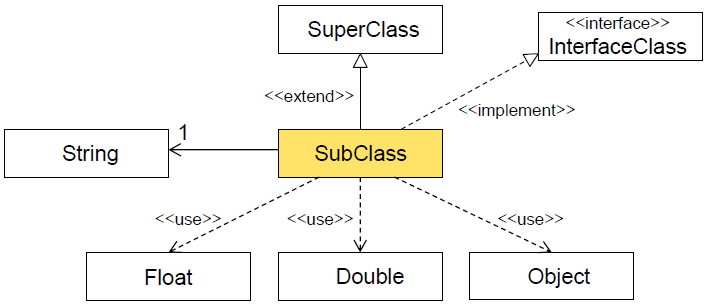
3. Inwiefern begünstigt gutes Klassen-Design die Software-Qualität?
   - *bessere Verständlichkeit / Lesbarkeit*
   - *bessere Testbarkeit*
   - *bessere Wiederverwendung (auch in anderem Kontext)*
   - *Auswirkungen von Fehlern halten sich in Grenzen*
   - *Änderungen sind einfacher möglich (lokal durchführbar, betreffen nur wenige Klassen)*

4. Was versteht man unter starker Kohäsion?
   starker innerer Zusammenhalt einer Klasse

5. Was versteht man unter loser Kopplung?
   loser Zusammenhang zwischen den Klassen



6. Visualisieren Sie mögliche Klassen-Beziehungen mit Hilfe von UML.

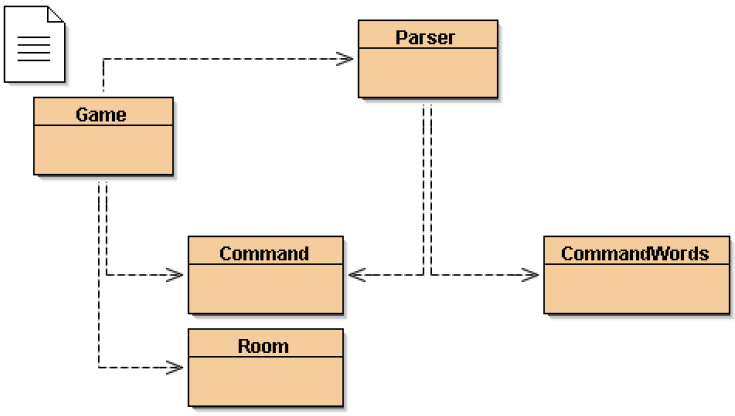


7. "Information Hiding" wurde mit einer "Burg" verglichen. Erklären Sie.
   - *Information Hiding (siehe OOP7) ist ein allgemeines Prinzip und begünstigt **starke Kohäsion** und **lose Kopplung**.*
   - *Details über die Implementation sollen dem Benutzer einer Klasse verborgen bleiben.*
   - *Der Benutzer soll sich nicht mit der Implementation beschäftigen müssen.*
   - *Dem Benutzer soll ein möglichst einfaches Interface angeboten werden.*
   - *Nur ein möglichst **einfaches Interface ist sichtbar** – das WAS.*
   - *Die Implementation ist verborgen – das WIE.*
   - *Der Zugriff erfolgt kontrolliert mit möglichst wenigen **Methoden** (Accessors und Mutators).*
   - ***Instanzvariablen** werden **private** deklariert.*
   - *Methoden für internen Gebrauch werden private deklariert.*

Kontrollfragen B

1. Wieso verschlechtert Code-Duplikation die Kohäsion?
   Durch die Duplikation wird der innere Zusammenhalt / Kohäsion verschlechtert, zerstört.

2. Was bedeutet "implizite Kopplung"?



   Implizite Kopplung: Game ist „indirekt" (implizit) mit CommandWords via Parser gekoppelt.
   Bei der Impliziten Kopplung besteht keine direkte Abhängigkeit, jedoch via weiteren Klassen die dazwischen sind bestehen trotzdem Abhängigkeiten, Kopplungen.

3. Wieso ist anstrebenswert, GUI-spezifische Aspekte zu kapseln?
   Weil ggf. später die Ausgaben nicht über das GUI laufen sondern z.B. via Netzwerk oder CLI (Command Line Interface).

4. Refactoring ist vor allem bei gutem Klassen-Design von Bedeutung. Stimmt diese Aussage?
   Nein, Refactoring sollte bei einem schlechtem Klassen Design angewendet werden.
   Refactoring ist eine Technik um das Design zu verbessern.
5. Refactoring ist eine wichtige Debugging-Methode. Wie steht's mit dieser Aussage?
   Falsch, Refactoring ist keine Debugging-Methode.
   Refactoring kann aber das Debuggen erleichtern.