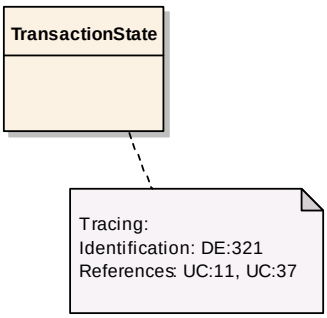
Übung 2: *Annotation* und *Reflection*

Ausgangslage:

Um die Nachvollziehbarkeit resp. Verfolgung von *Anforderungen* über das *Design* bis zur *Implementation* sicherzustellen, soll eine entsprechende *Traceability* sichergestellt werden (vgl. dazu z.B. auch *Capability Maturity Model Integration CMMI*).

Use-Cases, Packages, Klassen, Methoden etc. sollen alle eine eindeutige *Identifikation* enthalten. Im Weiteren soll jedes obiger Elemente eine Liste enthalten von den *Identifikationsnummern* derjenigen anderer Elemente, dessen Anforderungen dieses Element mitrealisiert resp. von jenen Anforderungen abhängig ist.

Beispiel:
Die Design-Klasse *TransactionState* mit der ***Identifikation*** **DE:321**, hängt von den Anforderungen der *Use-Case's* mit den ***Identifikationen*** **UC:11** und **UC:37** ab:

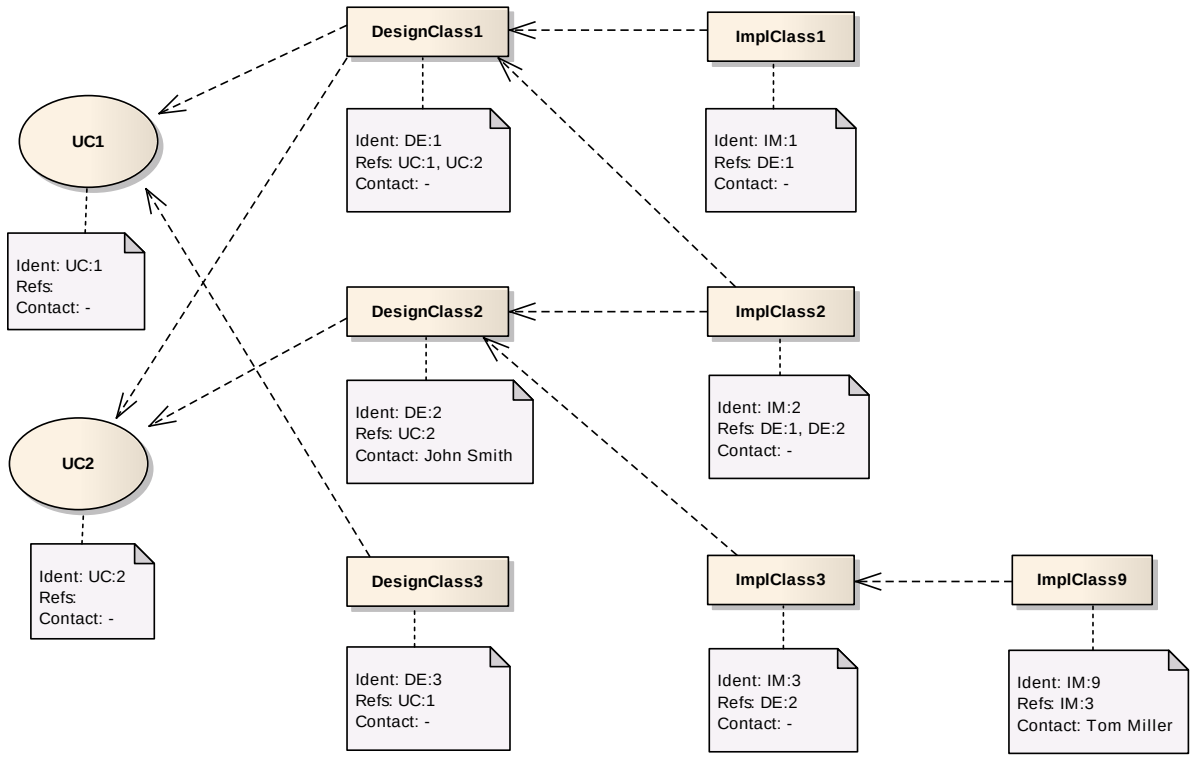


Ein solcher Mechanismus soll in unserem Projekt mit ***Java-Annotationen*** realisiert werden.
In einer ersten Phase werden nur drei Arten von Elementen identifiziert:

- Use-Case's: *UC*
- Design-Klassen: *DE*
- Implementations-Klassen: *IM*

Im Tracing soll optional auch eine *Kontakt-Person* angegeben werden können (wenn nicht gleich dem Autor des Elementes).

Gegeben ist folgendes *Test-Szenario*:



Hinweis: Die Abhängigkeiten gehen somit nur *unidirektional* von *hinten* (meint *Implementation*) nach *vorne* (meint *Requirements*)!

Aufgabe 1: Annotationen

Aufgabe a):

Es sollen die entsprechenden Annotationen definiert werden.

Diese sollen möglist sicher gestaltet sein.
Es soll z.B. nicht möglich sein, dass man nicht existierende Elemente angeben kann (z.B. *CU* anstelle von *UC*) oder anstelle einer Nummer irgend einen String (z.B. *UC:l1* anstelle von *UC:11*).

Aufgabe b):

Es sollen die Klassen gemäss vorangegangenem *Test-Szenario* erstellt werden.

Hinweise:

- nur die 7 *Klassen* mit den jeweils entsprechenden Annotationen abbilden (die *Use-Cases* sind nicht nötig).
- diese Klassen sind sonst *leer*.

- Beispiel:

```
@Tracing( ... )
class DesignClass1 {
}
```

Aufgabe 2: Reflection

Aufgabe a):

Es sollen *obige Annotationen* des Projektes zur *Laufzeit* ausgelesen werden.

Somit müssen zuerst einfach alle Klassen des Projektes über den *ClassLoader* mittels *Class.forName() „von Hand“* geladen werden.
Wir beschränken uns einfachheitshalber nur auf das gegenwärtige *bin*-Directory (wo sich alle unsere *class*-Files befinden. Siehe nächste Seite).

Als Beginn würde man über alle Files iterieren:

```
java.io.File binDir = new java.io.File("bin");   // Binary-Directory
for(String filename: binDir.list()) {
   ...
```

Eine entsprechende Ausgabe auf der Konsole könnte schlussendlich z.B. wie folgt aussehen:

```
Testing all Files in the bin-Directory:
File: DesignClass1.class - loading... found Tracing-Annotation!
File: DesignClass2.class - loading... found Tracing-Annotation!
File: DesignClass3.class - loading... found Tracing-Annotation!
File: Ident.class - loading...
File: IdentType.class - loading...
File: ImplClass1.class - loading... found Tracing-Annotation!
File: ImplClass2.class - loading... found Tracing-Annotation!
File: ImplClass3.class - loading... found Tracing-Annotation!
File: ImplClass9.class - loading... found Tracing-Annotation!
File: Tracing.class
File: TracingTest.class
```

Darauf folgend sollen alle Annotationen gemäss nachfolgendem Beispiel auf die Konsole ausgegeben werden.

```
Printing all Tracing-Annotations:
DesignClass1 :  Ident: DE:1   :  Refs: UC:1,UC:2   :  Contact: -
DesignClass2 :  Ident: DE:2   :  Refs: UC:2        :  Contact: John Smith
DesignClass3 :  Ident: DE:3   :  Refs: UC:1        :  Contact: -
ImplClass1   :  Ident: IM:1   :  Refs: DE:1        :  Contact: -
ImplClass2   :  Ident: IM:2   :  Refs: DE:1,DE:2   :  Contact: -
ImplClass3   :  Ident: IM:3   :  Refs: DE:2        :  Contact: -
ImplClass9   :  Ident: IM:9   :  Refs: IM:3        :  Contact: Tom Miller
```

Hinweise:
Es hat keine definierte *Start*- oder *Root*-Klasse mit welcher man beginnen könnte resp. müsste.

Aufgabe b):

Der Chef möchte die Abhängigkeiten sehen, aber nicht von *Hinten nach Vorne* (*Implementation* Richtung *Requirements*, wie sie in den *Annotationen* definiert sind), sondern umgekehrt.
Er möchte z.B. wissen, welche Klassen den *Use-Case UC:2* realisieren.
Es soll auch die *Abhängigkeits-Tiefe* dargestellt werden (vgl. im nachfolgenden Beispiel *ImplClass9*).

Auftrag:
Mit einem Aufruf-Parameter "*UC:2*" soll folgender Output auf der Konsole generiert werden:

```
Searching for: "UC:2"
DesignClass1 :  Ident: DE:1   :  Refs: UC:1,UC:2   :  Contact: -
   ImplClass1   :  Ident: IM:1   :  Refs: DE:1        :  Contact: -
   ImplClass2   :  Ident: IM:2   :  Refs: DE:1,DE:2   :  Contact: -
DesignClass2 :  Ident: DE:2   :  Refs: UC:2        :  Contact: John Smith
   ImplClass2   :  Ident: IM:2   :  Refs: DE:1,DE:2   :  Contact: -
   ImplClass3   :  Ident: IM:3   :  Refs: DE:2        :  Contact: -
      ImplClass9   :  Ident: IM:9   :  Refs: IM:3        :  Contact: Tom Miller
```

Hinweise:
Es muss noch keine optimale Lösung betreffend Ressourcen/Performance sein.
Es reicht, wenn es einfach funktioniert ;-)

