2. OWL Lite, OWL DL, OWL Full und OWL 2

- **OWL Lite:**
  Die "leichteste" Variante von OWL. OWL Lite ist so konzipiert, dass es nur die wichtigsten Features enthält, vorallem für solche Anwender, die nur Klassenhierarchien und einfache Constraints (nur Kardinalitäten von 0 oder 1) verwenden. Durch seine Einfachheit sollte ermöglicht werden, dass Tools zur Unterstützung bei der Entwicklung von OWL Lite Ontologien entwickelt werden.

  **Beispiel:**
  Punkt 2 und 6 in Aufgabe 2 können nicht mit OWL Lite ausgedrückt werden, statt dessen können wir nur definieren, dass eine Pizza mindestens ein „PizzaTopping" haben muss:

```
<owl:Class rdf:ID="Pizza">
  <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#hasTopping"/>
        <owl:minCardinality>1</owl:minCardinality>
      </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>
```

- **OWL DL:**
  Das DL steht hier für Description Logic und soll darauf hinweisen, dass OWL DL genuaso mächtig ist wie die Descritpion Logic. OWL DL sollte möglichst ausdrucksstark sein, aber gleichzeitig logisch entscheidbar bleiben. Daher verfügt OWL DL über alle Sprachkonstrukte von OWL, jedoch sind diese teilweise nur eingeschränkt verwendbar. So sind z.B. anonyme Klassen nur an bestimmten Stellen erlaubt. Außerdem kann die Kardinalität von transitiven Eigenschaften nicht eingeschränkt werden und Objekteigenschaften und Datentypeigenschaften sind disjunkt.
  **Beispiel:**
  Wir können zwar nun ausdrücken, dass eine Pizza mindestens zwei „PizzaTopping"s hat, jedoch können wir nicht ausdrücken, dass eine Pizza mindestens 4 „Ingredient"s hat, weil die Eigenschaft „hasIngredient" transitiv ist.

```
<owl:Class rdf:ID="Pizza">
  <owl:disjointWith rdf:resource="#PizzaTopping"/>
  <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#hasTopping"/>
        <owl:minCardinality>2</owl:minCardinality>
      </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>
```

- **OWL Full:**
  OWL Full wurde dafür entwickelt vollständig kompatibel mit RDF zu sein. Es unterstützt die gleichen Konstrukte wie OWL DL, jedoch gibt es kein Programm, das OWL Full logisch auswerten kann.
  **Beispiel:**

```
<owl:Class rdf:ID="Pizza">
  <owl:disjointWith rdf:resource="#PizzaTopping"/>
  <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#hasTopping"/>
        <owl:minCardinality>2</owl:minCardinality>
      </owl:Restriction>
   </rdfs:subClassOf>
```

```
    <rdfs:subClassOf>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#hasIngredient"/>
          <owl:minCardinality>4</owl:minCardinality>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>

<owl:ObjectProperty rdf:ID="hasIngredient">
    <rdfs:domain rdf:resource="#Pizza"/>
    <rdfs:range rdf:resource="#Ingredient"/>
        <rdf:type rdf:resource="&owl;TransitiveProperty"/>
</owl:ObjectProperty>
```

- **OWL 2:**
  OWL 2 ist eine vollkommen mit OWL 1.x kompatible Weiterentwicklung von OWL 1. Unter anderem bietet OWL 2 folgende Features: Ketten von Eigenschaften, qualifizierte Kardinalitäten, selbstdefinierbare Datentypen und asymmetrische, reflexive und disjunkte Eigenschaften. Außerdem führt OWL 2 Profile ein.
  **Beispiel:**
  Mit Hilfe der verkettung von Eigenschaften können wir nun die Eigenschaft „hasUncle“ definieren, indem wir sagen, dass diese Eigenschaft eine Verkettung von „hasParent“ und „hasBrother“ ist.

```
<owl:ObjectProperty rdf:ID="hasUncle">
    <owl:propertyChainAxiom rdf:parseType="Collection">
        <rdf:Description rdf:about="hasParent"/>
        <rdf:Description rdf:about="hasBrother"/>
    </owl:propertyChainAxiom>
</owl:ObjectProperty>
```