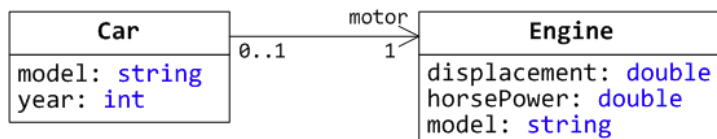


## Platformy Technologiczne Laboratorium 9

W ramach niniejszego ćwiczenia jako zadanie wprowadzające (niepunktowane, ale niezbędne do realizacji dalszych elementów) należy:

- utworzyć klasy implementujące model danych z niniejszego diagramu:



- utworzyć listę obiektów `Car`:

```

List<Car> myCars = new List<Car>(){
    new Car("E250", new Engine(1.8, 204, "CGI"), 2009),
    new Car("E350", new Engine(3.5, 292, "CGI"), 2009),
    new Car("A6", new Engine(2.5, 187, "FSI"), 2012),
    new Car("A6", new Engine(2.8, 220, "FSI"), 2012),
    new Car("A6", new Engine(3.0, 295, "TFSI"), 2012),
    new Car("A6", new Engine(2.0, 175, "TDI"), 2011),
    new Car("A6", new Engine(3.0, 309, "TDI"), 2011),
    new Car("S6", new Engine(4.0, 414, "TFSI"), 2012),
    new Car("S8", new Engine(4.0, 513, "TFSI"), 2012)
};
  
```

Następnie należy:

1. napisać 2 zapytania LINQ (0,5 pkt):
  - pierwsze dokonuje projekcji elementów kolekcji `myCars`, dla których model to A6, na typ anonimowy o dwóch własnościach:
    - `engineType` i `hppl`;
    - `engineType` ma wartość "diesel" dla silników "TDI" a "petrol" dla pozostałych;
      - $hppl = horsepower / displacement$
  - drugie, na podstawie rezultatu zapytania pierwszego, grupuje wartości `hppl` po typie silnika (`engineType`);
    - używając pętli `foreach` wyświetlić utworzone grupy wraz ze średnią wartością `hppl`:
      - `petrol: 83,9015873015873`
      - `diesel: 95,25`
2. zaimplementować serializację (0.2 pkt) i deserializację (0.2 pkt) kolekcji `myCars` do formatu XML. W trakcie serializacji:
  - zmienić nazwę elementu będącego korzeniem dokumentu na "cars" (0,3 pkt);
  - zmienić nazwę elementu "motor" na "engine" (0,1 pkt);
  - zmienić nazwę elementu "Car" na "car" (0,1 pkt);

- element "model" opisujący silnik ma stać się atrybutem elementu "engine" (0,1 pkt);
3. napisać i zaprezentować wyrażenie *XPath*, które na wygenerowanym dokumencie XML:

- obliczy przeciętną moc samochodów o silnikach innych niż TDI (0,5 pkt)

```
XElement rootNode = XElement.Load("CarsCollection.xml");  
double avgHP = (double) rootNode.XPathEvaluate("myXPathExpression1");
```

- zwróci modele samochodów bez powtórzeń (0,5 pkt)

```
IEnumerable<XElement> models = rootNode.XPathSelectElements("myXPathExpression2");
```

4. napisać zapytanie LINQ do kolekcji *myCars*, tak aby plik XML wygenerowany przez poniższą metodę miał taką samą strukturę jak wygenerowany plik XML z poprzedniego zadania (przy serializacji) (0,5 pkt)

```
private void createXmlFromLinq(List<Car> myCars) {  
    IEnumerable<XElement> nodes = ... // zapytanie LINQ  
    XElement rootNode = new XElement("cars", nodes); // stwórz węzeł  
    zawierający wyniki zapytania  
    rootNode.Save("CarsFromLinq.xml");  
}
```

5. korzystając z LINQ to XML wygenerować na podstawie kolekcji *myCars* dokument XHTML zawierający tabelę, której wiersze reprezentują kolejne elementy kolekcji (1 pkt).

*Podpowiedź: można ułatwić sobie zadanie poprzez załadowanie pustego dokumentu XHTML (template.html) i dołączenie wygenerowanego elementu table.*

6. załadować dokument XML wygenerowany w punkcie 2, a następnie przeprowadzić na nim modyfikację w taki sposób, aby:
- zmienić nazwę elementu *horsePower* na *hp* (0,5 pkt);
  - zamiast elementu *year* utworzyć atrybut o tej samej nazwie w elemencie *model* (0,5 pkt).