Falls Sie die Übung der letzten Woche noch nicht abgeschlossen haben, so widmen Sie sich bitte erst dieser. Das dabei entstandene Programm soll hier erweitert werden.

Datenbankanfragen in db4o

1. **Beispiel-Anfragen**
   Die Anfrage durch Beispiel–Objekte haben wir schon in der Übung 1 kennengelernt. Ein QBE-Sonderfall ist, dass als Beispiel–Objekt ein Klassenobjekt angegeben werden kann, um damit nach allen Objekten dieser Klasse (dem Extent) in der Datenbank zu fragen.

   Modifizieren Sie bitte Ihr Programm der letzten Woche so, dass zur Anfrage aller Fahrer das entsprechende Klassen-Objekt (Visual Basic: `GetType(Pilot)` bzw. Java: `Pilot.class`) in einer Anfrage verwendet wird.

2. **native Anfragen**
   db4o erlaubt es, Anfragen typsicher in Visual Basic zu formulieren. Dies erfolgt in der Form (analog in Java vgl. Vorlesungsfolien):

   ```
   Dim result As IList(Of MyClass) = db.query(Of MyClass) (
      Function (o As MyClass)  Predicate(o)
   )
   ```

   wobei `MyClass` die Klasse bestimmt, von der Objekte angefragt werden und `Predicate(o)` ein boolescher Ausdruck ist, der Bedingungen an die gewünschten Objekte stellt. Dieser Ausdruck kann kompliziert sein. `query` liefert alle Objekte, für die der Ausdruck `true` ist.

   Erweitern Sie bitte Ihr Programm und stellen Sie für die folgenden Bedingungen native Anfragen:

   (a) Fahrer, deren Punktzahl 100 beträgt.

   (b) Fahrer, deren Punktzahl zwischen 99 und 199 Punkten liegt oder die Rubens Barrichello heißen.

   (c) Fahrer, deren Punktzahl zu denen einer gegebene Liste
   `final int[] points = {1, 90, 100};` (Java)
   bzw.
   `Dim points = {1, 90, 100}` (Visual Basic)
   gehört.

   (d) Experimentieren Sie bitte mit eigenen nativen Anfragen.

3. **einfache SODA–Anfragen — nur für Master-Informatik**
   Bei SODA–Anfragen wird zunächst eine (uneingeschränkte) Anfrage erzeugt. Durch wiederholten Aufruf der Methode `constrain()` wird die Anfrage dann eingeschränkt. Schließlich wird die Anfrage mit der Methode `execute` gestellt. Um Bedingungen an Attribute zu formulieren, muss in die entsprechenden Attribute abgestiegen werden (Methode `descend`):

   ```
   Query query = db.query();
   query.constrain(MyClass.class);                 // Einschraenken auf Klasse
   query.descend("myAttr").constrain(myValue); // Bedingung an Attribut
   ObjectSet<MyClass> result=query.execute();  // Anfrage stellen
   ```

wobei, `MyClass` die Klasse ist, deren Objekte wir anfragen, `myAttr` der Name eines Attributs ist, dessen Wert `myValue` wird fordern. `myValue` muss ein Objekt sein (ggf. die Wrapper-Klassen für `int`, `double`, ..., verwenden).

Die mit `constrain()` erzeugten Einschränkungen (Klasse `Constraint`) können vielfältig durch Methoden erweitert werden, etwa durch `and()`, `or()`, `not()` (um Einschränkungen miteinander zu verknüpfen) oder durch `greater()` (Zahlenvergleich), `like()`, `startswith()` (Stringvergleiche) und weitere.

(a) Formulieren Sie bitte die Bedingungen (a) und (b) aus Aufgabe 2 in Form von SODA–Anfragen.

(b) Für Bedingung 2(c) generieren Sie bitte an Hand der Liste `points` eine passende Anfrage (mit enstprechend vielen `or`–Teilen).

(c) Experimentieren Sie bitte auch mit eigenen SODA–Anfragen, machen Sie sich bitte mit den `Constraint`-Methoden vertraut.