Design Patterns and Frameworks	**Exercise Sheet No. 4**
Dipl.-Inf. Florian Heidenreich	Software Technology Group
INF 2080	Institute for Software and Multimedia Technology
`http://st.inf.tu-dresden.de/teaching/dpf`	Department of Computer Science
	Technische Universität Dresden
	01062 Dresden

Extensibility Patterns

Task 1: Raytracer Objects

Raytracing is a technology for generating photo-realistic images from scene description data. Very advanced types of this technology have recently been used for creating quite a few, and very impressive, animated pictures (such as "Shrek", or "Nemo").

Imagine you have to realise a raytracer application. At the core of such an application is a data structure which maintains and represents the scene graph or the "world". Objects can be very simple objects which have a geometrically defined shape and a typically algorithmically defined material, but they can also be arbitrarily complex compositions of other objects. The scene graph is then repeatedly traversed to check for intersections with various optical rays.

1a) Design a class structure which can represent a scene graph. What design pattern is applicable?

1b) It is often useful to be able to transform existing objects in a scene graph—for example, by moving, scaling or rotating them. This should be possible for simple and complex objects alike. Enhance the class structure from the previous subtask to enable transformation of objects. Which design pattern do you use?

Hint A standard raytracing trick is to transform the ray vectors (a start location and a direction) instead of the objects. This is mathematically equivalent and makes life much easier for most situations.

1c) When transforming an object it is typical to construct the transformation from the basic transformation scaling, rotation, and translocation by putting them in a sequence of transformation operations. Extend the class structure to represent such sequences of transformations. Discuss different solutions.

1d) When tracing complex objects it is often worthwhile to first check whether the ray will ever intersect their bounding box (i.e., the smallest box completely enclosing the object). Enhance the class structure to add this functionality. What design pattern could you use? Are there other ways of doing it?

Task 2: Record Extension

Design an access layer for record-oriented employee data, which is held in several files. Your access layer should support the following operations:

- `getSalaryFor (id)` returns the salary for the employee with the given id.
- `addToSalary (id, value)` increases the salary for the employee with the given id.

Employee data is stored in hierarchically organised files. Each file contains a sequence of records, each of which either stores information about one employee or references another file with further employee records. This structure has been used to reflect the hierarchy of the companies organisation in the structure of the data storage.

2a) Design the core class structure. Your design should hide from the client all information regarding the specific file in which a record is situated, while maintaining this information internally. Which design pattern could you use?

2b) The employee database can become rather big. It is therefore not very intelligent to load all records into memory when the database is first opened. Rather, we would like to load only the records that are accessed. Use Chain of Responsibility to implement this feature.

2c) Employee information is rather sensitive data, so we do not want everybody to be able to access it. Enhance the design to allow for role-based access control (RBAC) to data.

2d) Add the capability to keep the physical records in the files updated whenever `addToSalary()` has been called. Realise it so that files can be opened in read-only or in read-write mode.