Design Patterns and Frameworks	**Exercise Sheet No. 6**
Dr.-Ing. Max Leuthäuser	Software Technology Group
INF 2081	Institute for SMT
`http://st.inf.tu-dresden.de/teaching/dpf`	Department of Computer Science
	Technische Universität Dresden
	01062 Dresden

Architecture Mismatch Patterns

Task 6.1: Medi(t)ative Air

Design an application which enables you to book the cheapest flight to a destination of your choice out of a number of providers.

1a) Assume, every provider is known in advance, and implements an interface `IFlightProvider`, which provides operations for querying for a connection, and for booking a flight. Develop an architecture which enables clients to interface to these providers and book the cheapest flight on offer for the destination and date they are interested in. Flight providers should require (and receive) no knowledge on other flight providers known to the system. Also, clients should not need to know which flight providers are registered with the system.

Which design pattern could you use?

1b) Many airlines offer on-line booking services as web services. How can you incorporate such an airline as a flight provider?

Task 6.2: Photo-realistic Facade

Ray tracing is a rather complex technique. It consists of a number of steps from parsing a scene-graph description (often called a 'script'), building a scene-graph instance in memory, optimising the scene graph, tracing rays through all pixels of the target image, possibly oversampling to provide anti-aliasing, to actually rendering the image; that is, transforming the ray colour values into the value range of image colour values. On the other hand, as a client all you want to do is provide a script and obtain an image.

2a)

Use the Facade pattern to provide clients of a ray-tracing subsystem with easy access to ray-tracing functionality.

Task 6.3: Pattern Relations

In this task you will explore the relations between the various patterns that we have been looking at in the course so far.

3a)

Compare Template Method and Template Class. What do they have in common, what is the major difference? How do they achieve variability? What is their relation to the Template Hook and the Objectifier patterns?

3b)

Compare the extensibility patterns Decorator, Composite, Chain of Responsibility, and Observer. What are the mechanisms through which they achieve extensibility? Why does Proxy not provide extensibility? What is the relation of these patterns to Template Class and Object Recursion?

3c)

Now compare the architecture-glue patterns Adapter, Facade, and Mediator. How do they cope with architectural mismatch? How do they compare to the variability and extensibility patterns?

3d) *

Sketch a chart of the relations between the design patterns Template Method, Template Class, Objectifier, Bridge, Strategy, State, Visitor, Proxy, Adapter, Facade, Mediator, Object Recursion, Decorator, Composite, Chain of Responsibility, and Observer. Use arrows to indicate specialisation (based on class structure, behaviour, or intent) and introduce additional helper concepts if you need them to represent commonalities which have not yet been abstracted into an individual pattern.