Einführung in die Softwaretechnik 2018
Sheet 08

Maximilian Frühauf

24th June 2018

1. Use the Pattern Model (introduced in Lecture 08, slides 47ff) to describe the Bridge Pattern (see Lecture 06, slides 83ff).

   - **Pattern Name:** Bridge Pattern
   - **Problem:** Many design decisions have to me made final at design time ("design window") or at compile time as different abstractions can not be exchanged at runtime.
   - **Context:** Different Clients need to choose different implementations at runtime.
   - **Forces:** Delay the binding between an interface and its subclass to the runtime time of the system.
   - **Solution:**
     - Extract different implementations into separate classes Called *Concrete Implementor*
     - Define an interface to all of the *Concrete Implementor* classes as *Implementation*. Implementation services are exposed via the `OperationImpl()` Method.
     - The Client chooses a Specific *Refined Abstraction* which all are a subclass of the interface *Abstraction*.
     - This *Refined Abstraction* then chooses one of the *Concrete Implementor*s in the Solution Domain
   - **Benefits:**
     - A new Implementation can be added without modifying the Abstraction or the Client.
     - Implementations can be chosen dynamically at runtime.
   - **Consequences:**
     - Client is not aware of the detailed concrete Implementations.
     - The different Abstractions decide which concrete Implementation to use at runtime.
   - **Follow-On Problems(s):**
     - If many classes get added as Abstractions or Implementations the pattern will get confusing as many classes have to be considered.