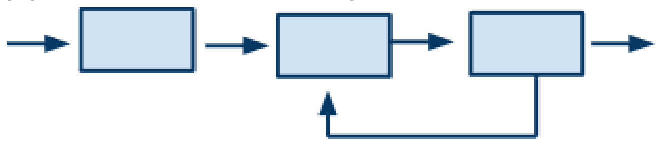
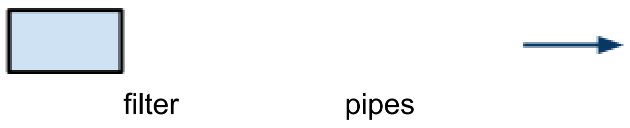
Question 1
(a) Define the pipe & filter style. [2]

- Architectural pattern for stream processing
- It consists of any number of components (filters) that transform or filter data, before passing it on via connectors (pipes) to other components.
- The filters are all working at the same time.

(b) Provide the block diagram for the pipe & filter style. [4]





(c) List advantages & disadvantages of the pipe & filter style. [3]

Advantages:

- Simple composition
- Reuse
- Prototype
- Easy growth & evolution
- Concurrency & parallelism

Disadvantages:

- Poor performance
- Not appropriate for interaction
- Low fault tolerance threshold
- Data transformation
- Increases complexity & computation

(d) Briefly describe the three variation of the pipes & filters style. [3]

- Pipelines: restricted to linear topology
- Bounded pipes: restricted the amount of data on a pipe
- Typed pipes: data on a pipe to be of an acceptable type.

(e) Briefly describe a typical application for the pipe & filter style. [1]

- Unix shell programs: cat file1 | sort | grep keyword
- Compilers: source code -> lex -> syn -> sem -> opt -> code -> machine code

---

Question 2
(a) Define the implicit invocation style.
An event based architectural style where the components do not interact with each other directly. Rather all system changes are published via events and interested/registered parties react to these events.

(b) Provide the block diagram for the implicit invocation style
See the paper

(c) List advantages & disadvantages of the implicit invocation style
Advantages
- Minimal dependency and loose coupling
- High reusability
- High scalability

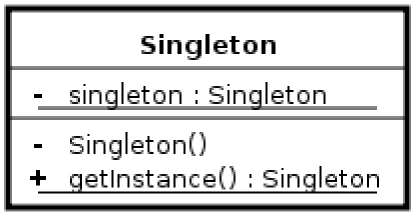
Disadvantages
- Loss of execution control
- Data exchange between components
- Unpredictability of who will react to the events

(d) Describe a variation of the implicit invocation style.
Publish subscribe frameworks where events are generated and published to a central repository. Events can be categorized based on topics. The components register themselves to topics of interests and react to the the events.

(e) Briefly describe a typical application for the implicit invocation style.
MVC based GUI components
IDEs

Question 3
Q3a. What is the intent of the Singleton design pattern?
ensure a class only has one instance, and provide a global point of access to it.

Q3b. Provide a UML class diagram for the Singleton design pattern.
See the Wikipedia/GOF Book or class notes



Q3c. List two advantages of using the Singleton design pattern.
- controlled access
- variable number of instances

Q3d. Describe how Singleton objects result in tighter coupling and what can be done to mitigate it.
Caller needs to know the exact class name and accessor method to obtain the singleton. This results in a tight coupling promoting the use of the exact type of the singleton object.

Q3e. What concerns must be addressed in using stateful Singleton objects?

- access synchronization.
- concurrency

Q3f. Provide two examples of systems where Singleton objects are useful.
- true singleton instances such as the java Math class
- main application GUI Frame
- Loggers

Question 4
(a) What is the intent of the Visitor pattern?
To decouple a structure from the operations that act on it.
(b) Visitor class diagram
- http://en.wikipedia.org/wiki/Visitor_pattern
- Classes in the Visitor hierarchy should have visit(ConcreteVisiteeA), visit(ConcreteVisiteeB), visit(ConcreteVisiteeC), etc, methods
- classes in the Visitee hierarchy should have accept(AbstractVisitor) method

(c) What is the intent of the Interpreter pattern?
“Given a language, define a representation for its grammar along with an interpreter that uses the representation to interpret sentences in the language.”
(d) Interpreter class diagram
- http://en.wikipedia.org/wiki/Interpreter_pattern
- expecting a hierarchy with a superclass AbstractExpression that has two subclasses: Terminal and NonTerminal.
- AbstractExpression declares the interpret(Context) method

(e) Which design pattern would you use to implement an experimental language?
If the language is experimental, then the grammar is likely to change. The Interpreter pattern handles changes in the grammar of the language being compiled in a localized way: just change the classes representing the changed productions in the grammar. The Visitor pattern, by contrast, would require changes to every subclass of AbstractVisitor for such a change.
(f) Which design pattern would you use to implement a well-established language?
If the language is established, then the grammar is likely to be stable and we will be more interested in adding new features to the compiler: type checker, pretty-printer, etc. The Visitor pattern localizes such changes: just add a new Visitor. By contrast, the Interpreter pattern would require changes to almost every class to add such features.

Question 5
Q5a. Briefly describe each of the the following: [3]
- **Concrete architecture:** An architecture that captures the domain knowledge, system requirements (functional & non-functional) and detailed implementation design for a specific instance / version of the software.
- **Conceptual architecture:** An architecture that captures the domain knowledge, high level system requirements (functional & non-functional). Not used for a particular implementation of the system.
  OR

"*direct attention at an appropriate decomposition of the system without delving into the details of interface specification*"
- **Reference architecture**: An architecture that mainly captures the domain knowledge across many different product designs

Q5b. Define architectural drift or architectural erosion.
Difference between conceptual and concrete architecture.
Erosion: conceptual architectural violations
Drift: concrete architecture shifts away from conceptual architecture

---

Q6a. What (4+1) views are defined by Kruchten? (name of view + one line description) [5]
Please see the paper.

Q6b. Which views are required for every design and and which views are optional for some designs. [3]
Scenario (use case) view should be present for all designs. Logical view should also be present.

Process view can be eliminated for single process applications.

Question 7
(a) List some of the shortcomings of the first-generation enterprise web application architecture that were resolved in later generations? [3]
- Script based, therefore difficult evolution path
- Security
- Performance & Scalability: Each script runs as an individual server process. Therefore heavy system load would result in poor performance and complex scalability issues.
- Script based server components generally require a secondary media for inter-component communication.

(b) Describe the request-response cycle of enterprise web applications.
A web client makes an synchronous request to the server for a resource in the form of a URL or a HTML form submission (POST/GET). The server responds to the request by processing the request using the business components and then generating and returning an HTML document.

(c) What is the main advantage of asynchronous communication?
The client side application (GUI) is a lot more responsive since the web client does not have to reload the entire web document for each and every user action.

(d) Where is the user session-state typically maintained in a modern enterprise web applications?
On the presentation layer. The presentation layer utilizes a session token for each user state. The token is tracked via cookies on the browser and can be persisted in the database for future retrieval of the user session.

(e) Name three non-functional requirements that enterprise applications usually get right.
- concurrency
- availability

- security
- performance
- fault-tolerance
- application distribution & deployment
- evolution
- re-usability

(f) Name three non-functional requirements that enterprise applications often do poorly on.

- cost
- ease of use
- interoperability
- portability
- throughput