**Tutorial No. 01**

**Aim:** Implement a calculator based on monolithic and client-server architecture.

**Theory:**

**Monolithic architecture:**

1. A monolithic architecture is the traditional unified model for the design of a software program. Monolithic, in this context, means composed all in one piece.
2. Monolithic software is designed to be self-contained; components of the program are interconnected and interdependent rather than loosely coupled as is the case with modular software programs.
3. In a tightly-coupled architecture, each component and its associated components must be present in order for code to be executed or compiled.
4. There are benefits to monolithic architectures as well. Monolithic programs typically have better throughput than modular approaches, such as the micro service architecture (MSA) and they can be easier to test and debug because, with fewer elements there are fewer variables that come into play.

**Client –Server Architecture:**

1. Client-server (C/S) or "two-tier" architecture (in comparison with a peer-to-peer architecture) has two separate types of nodes on the network: servers, that store information and clients that send requests for information to servers.
2. Usually, but not always, a client computer and a server computer are two separate devices. A server computer contains large amounts of memory and disk space, while client computers features graphic user interface to support the display of data stored on server. There are many different types of client/server software however their basic architecture remains the same.

**Client/Server Benefits**

Client/server approach to networking has proven to be a cost-effective way to share data between tens or hundreds of computers. Considering the client/server computing in terms of a manager-employee relationship, the following list outlines some of the benefits and drawbacks of client/server solutions.

1. **Centralized Information Storage.**

The server stores the data and coordinates the access to information and its modification. This helps to keep the data consistent and up-to-date, even when multiple users/clients are working with it simultaneously.

1. **Delegation**

Managers or team leaders have more information, experience and knowledge about the company and day-to-day operations. Their deep understanding of the business processes, priorities, strategy, goals, and important tasks allows them to easily share information as needed and delegate work to their employees.

1. **Focus**

The employees or team members may have less knowledge and experience or their vision of the strategy and goals is far from clear. By using [client/server software](http://en.wikipedia.org/wiki/Client%E2%80%93server_model) they have more focused tasks and also they might get a clear picture of company strategy if their team leaders provide them with access to such information.

1. **Collaboration**

This is the essential aspect of how client/server computing works. It lets the manager plan, assign and notify the team members of their tasks. Once finished with their work, employees report the results back to the manager for verification.

1. **Security**

Very often the database is securely locked away from unauthorized access and its data is denied 'view' and/or 'edit' permission, that prevents violations from outside and inside the office.

Another type of related software architecture is known as peer-to-peer, because each host or application instance can simultaneously act as both a client and a server (unlike centralized servers of the client-server model) and because each has equivalent responsibilities and status. Peer-to-peer architectures are often abbreviated using the acronym P2P.

The client-server relationship describes the relation between the client and how it makes a service request to the server, and how the server can accept these requests, process them, and return the requested information to the client. The interaction between client and server is often described using sequence diagrams. Sequence diagrams are standardized in the Unified Modeling Language

**Requirement Specification:**

This section describes the external influences imposed on the calculator program.

**Hardware and Software:**

The program shall be written in standard JAVA, as compiled by the JAVA JDK. The program shall use only standard JAVA library functions. The program shall be usable on any system which supports the compiler, and shall not require any particular hardware or software. 2.2

**External Data Bases and File Interfaces:**

There are no existing external data bases or files that will be needed by this program.

**Human Interfaces:**

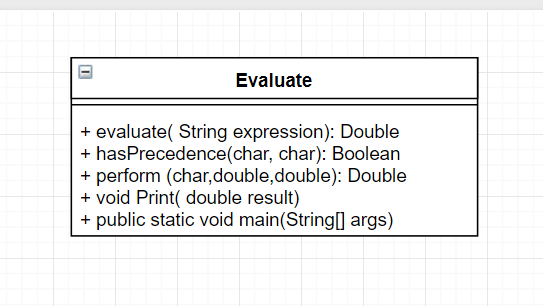
The program shall operate as much as possible in the same way as a regular handheld calculator, such that anyone familiar with the operation of such a device should have little trouble using the program. Once the program is started, the program shall produce a prompt. The user can then type a sequence of numbers and operators, similar to a regular calculator. The program will display the calculated answer to the entered problem on the line following the input line. Some examples that show the proper function of the calculator follow below (the computer-generated output is underlined): 1.47\*3.5+15.92=26.9024

**Functions:**

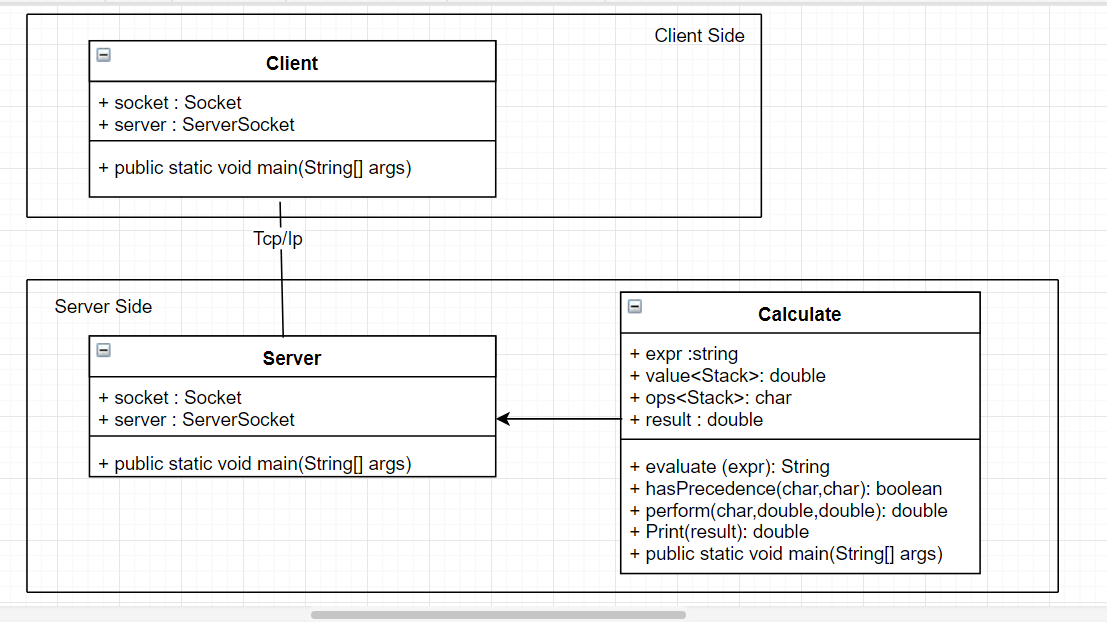
The following section outlines all of the functions required of the calculator program. 3.1 Data Flow Diagram Figure 1 is a data flow diagram which identifies the major functions required of the calculator program and shows their relationship to each other. The numbers included with each function are keyed to the subsections

**Class Diagram:**

**Monolithic:**

****

**Client-Server:**

****

**Conclusion:**

Thus we studied the monolithic architecture and the client-server architecture and implemented the simple calculator.