ASSIGNMENT NO. A4

Problem Statement:

In an embedded system application Dining Philosopher's problem algorithm is used to design a software that uses shared memory between neighboring processes to consume the data. The Data is generated by different Sensors/WSN system Network and stored in MOngoDB (NoSQL).Implementation be done using Scala/ Python/ C++/ Java. Design using Client-Server architecture. Perform Reliability Testing. Use latest open source software modeling, Designing and testing tool/Scrum-it/KADOS, NoSQLUnit and Camel.

Learning Objectives:

1. To Understand the Dining Philosopher's problem.
2. To learn about working and commands of MongoDB.

Learning Outcomes:

Learnt about the Dining philosopher's solution and connection with MongoDB for storing data.

Software and Hardware Requirements:

1. 64-bit operating System(Linux)
2. MongoDB
3. Terminal
4. java Compiler
5. Eclipse
6. Junit(For testing)
7. Modelio 3.6.1

Theory

Dining Philosopher's:

In computer science, the dining philosophers problem is an example problem often used in concurrent algorithm design to illustrate synchronization issues and techniques for resolving them.

Problem statement of Dining Philosopher:

Five silent philosophers sit at a round table with bowls of spaghetti. Forks are placed between each pair of adjacent philosophers. Each philosopher must alternately think and eat. However, a philosopher can only eat spaghetti when he has both left and right forks. Each fork can be held by only one philosopher and so a philosopher can use the fork only if it is not being used by another philosopher. After he finishes eating, he needs to put down both forks so they become available to others. A philosopher can take the fork on his right or the one on his left as they become available, but cannot start eating before getting both of them. Eating is not limited by the remaining amounts of spaghetti or

stomach space; an infinite supply and an infinite demand are assumed. The problem is how to design a discipline of behavior (a concurrent algorithm) such that no philosopher will starve; i.e., each can forever continue to alternate between eating and thinking, assuming that no philosopher can know when others may want to eat or think.

Problems:

The problem was designed to illustrate the challenges of avoiding deadlock, a system state in which no progress is possible. To see that a proper solution to this problem is not obvious, consider a proposal in which each philosopher is instructed to behave as follows: think until the left fork is available; when it is, pick it up; think until the right fork is available; when it is, pick it up; when both forks are held, eat for a fixed amount of time; then, put the right fork down; then, put the left fork down; repeat from the beginning. This attempted solution fails because it allows the system to reach a deadlock state, in which no progress is possible. This is a state in which each philosopher has picked up the fork to the left, and is waiting for the fork to the right to become available. With the given instructions, this state can be reached, and when it is reached, the philosophers will eternally wait for each other to release a fork.[4]

MongoDB:

MongoDB is an open-source document database, and leading NoSQL database. MongoDB is written in C++. MongoDB (from humongous) is a cross-platform document-oriented database. Classified as a NoSQL database, MongoDB eschews the traditional table-based relational database structure in favor of JSON-like documents with dynamic schemas (MongoDB calls the format BSON), making the integration of data in certain types of applications easier and faster. MongoDB is developed by MongoDB Inc. and is published as free and open-source software under a combination of the GNU Affero General Public License and the Apache License. As of July 2015, MongoDB is the fourth most popular type of database management system, and the most popular for document stores.

Main features of MongoDB:

1. Document-oriented

Instead of taking a business subject and breaking it up into multiple relational structures, MongoDB can store the business subject in the minimal number of documents. For example, instead of storing title and author information in two distinct relational structures, title, author, and other title-related information can all be stored in a single document called Book.

2. Ad hoc queries

MongoDB supports field, range queries, regular expression searches. Queries can return specific fields of documents and also include user-defined JavaScript functions.

3. Indexing

Any field in a MongoDB document can be indexed ? including within arrays and embedded documents (indices in MongoDB are conceptually similar to those in RDBMSes). Primary and secondary indices are available.

4. Replication

MongoDB provides high availability with replica sets.[6] A replica set consists of two or more copies of the data. Each replica set member may act in the role of primary or secondary replica at any time. The primary replica performs all writes and reads by default. Secondary replicas maintain a copy of the data of the primary using built-in replication. When a primary replica fails, the replica set automatically conducts an election process to determine which secondary should become the primary. Secondaries can optionally perform read operations, but that data is eventually consistent by default.

5. Load balancing

MongoDB scales horizontally using sharding. The user chooses a shard key, which determines how the data in a collection will be distributed. The data is split into ranges (based on the shard key) and distributed across multiple shards. (A shard is a master with one or more slaves.). Alternatively, the shard key can be hashed to map to a shard ? enabling an even data distribution. MongoDB can run over multiple servers, balancing the load and/or duplicating data to keep the system up and running in case of hardware failure. MongoDB is easy to deploy, and new machines can be added to a running database.

6. File storage

MongoDB can be used as a file system, taking advantage of load balancing and data replication features over multiple machines for storing files. This function, called Grid File System,is included with MongoDB drivers and available for many development languages (see "Language Support" for a list of supported languages). MongoDB exposes functions for file manipulation and content to developers.

Mathematical Model

```
S = {s, e, I, o, f, DD, NDD, success, failure}
s = {Initial state of system}
  s = Mongodb server started

I = {input of system}
I = (I1)
where,
   I1 = {x such tha x= 0-9 is a number of philosophers }
```

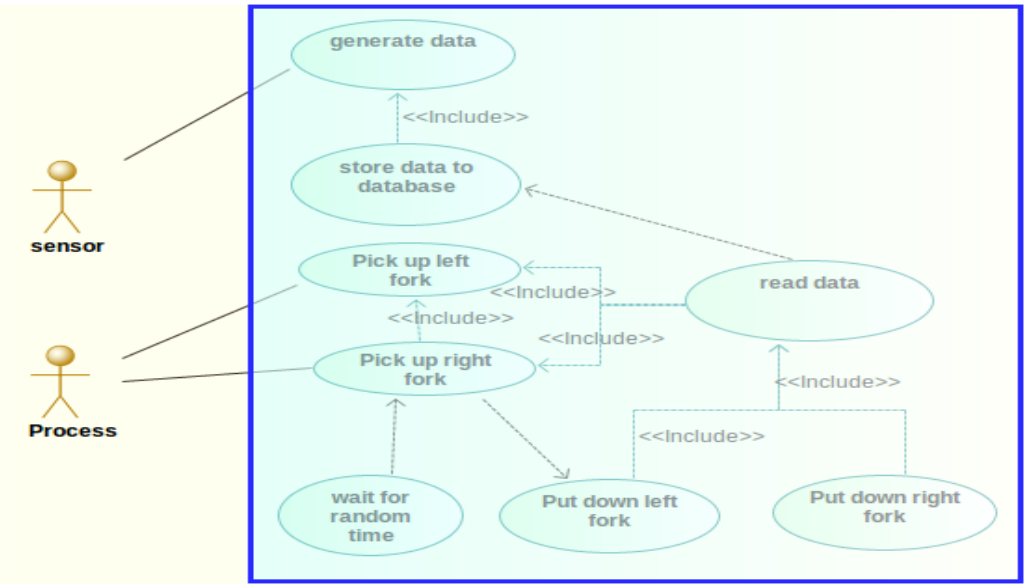
o = {Output of system}
o = Philosopher's problem get solved.

DD = {Deterministic Data }
= Number of philosopher's and fork.
NDD = {Non-Deterministic Data }
= Which philosopher will put which fork.
NDD = {a-z, A-Z}

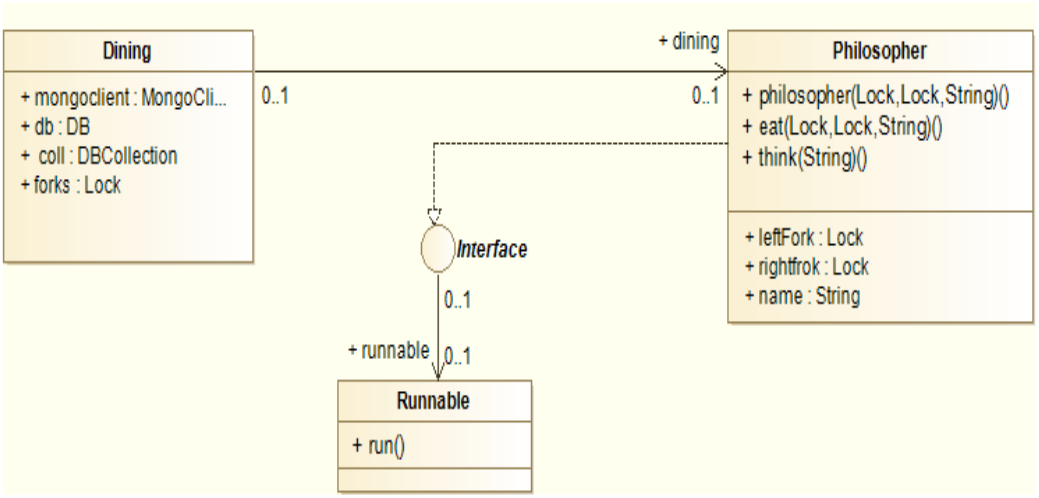
f = {start(),think(),eat(),pick(), free(),exe()}
where,
start()={ function which used to start the process}
think()={function is used to think for while}
eat()={function is used to eat process}
pick()={function is used to pick up the spoon.}
free()={function is used to free the the spoon.}

success = desired output is generated i.e. Required operation is performed.
failure = desired output is not generated.

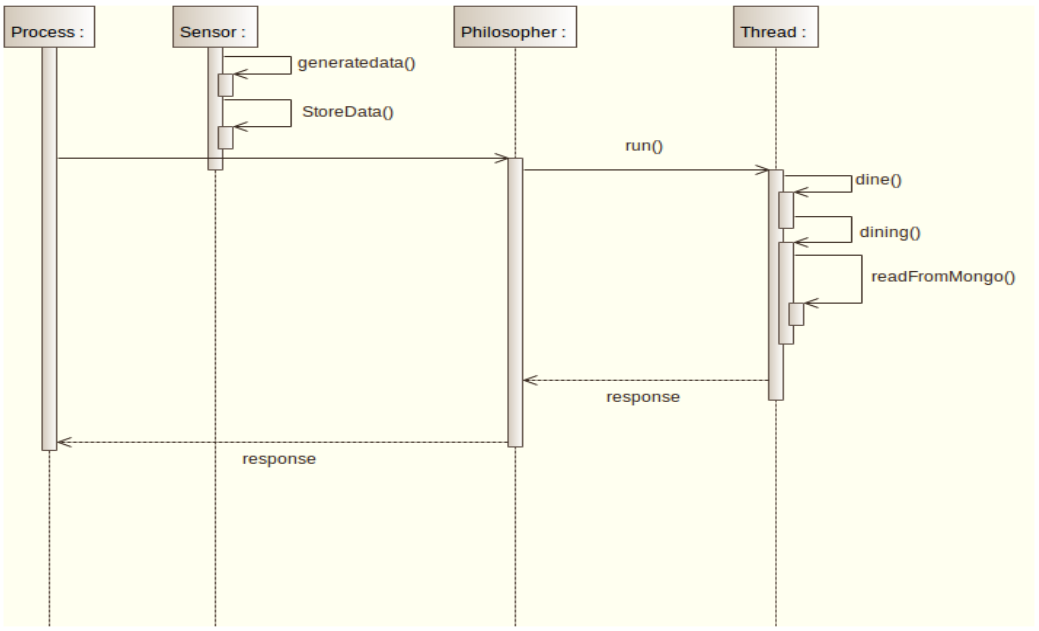
Use case Diagram



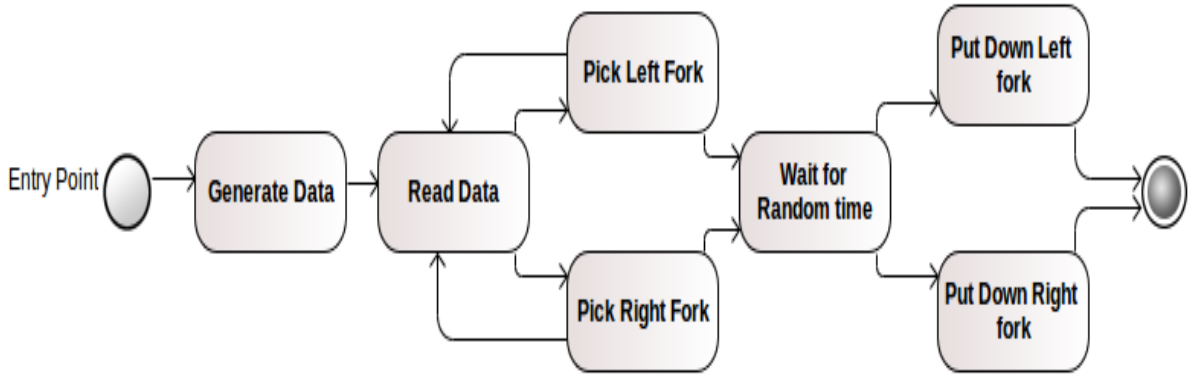
Class Diagram



Sequence Diagram



State Diagram



Algorithm

1. Start
2. Input the no of philosophers and iteration.
3. philosophers think for while.
4. pick up the spoon
5. eat for particular time.
6. free the spoon.
7. again repeat the process form step 3 upto no of iteration.
8. end

Black Box Testing

Black-box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings. This method of test can be applied to virtually every level of software testing: unit, integration, system and acceptance. PRECONDITIONS:

1. Input by philosopher i.e which fork taken by philosopher.

White Box testing

White-box testing is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality (i.e. black-box testing). White box testing is a testing technique, that examines the program structure and derives test data from the program logic/code. The other names of glass box testing are clear box testing, open box testing, logic driven testing or path driven testing or structural testing.

1. Number of Philosopher's and forks known.
2. No one is idle.

Positive Testing

Input	**Expected Output**	**Actual Output**
1. No Conflicts for resource	No deadlock	Same as Expected
2. During conflict one philosopher puts resource down	No deadlock	Same as expected

Negative Testing

Input	**Expected Output**	**Actual Output**
1. Conflicts for resource	Deadlock	Same as Expected
2. During conflict no philosopher puts resources down	Deadlock	Same as expected

Conclusion

We have successfully implemented the Dining Philosopher's algorithm.