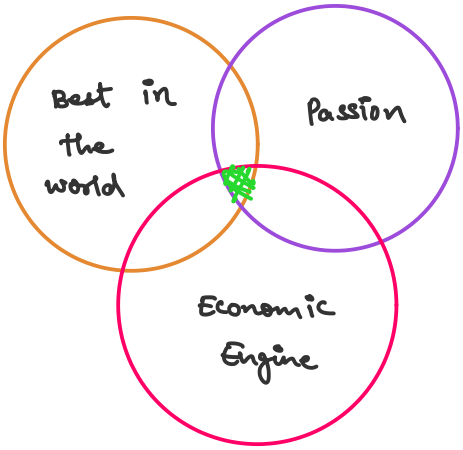
21__18-09-2022

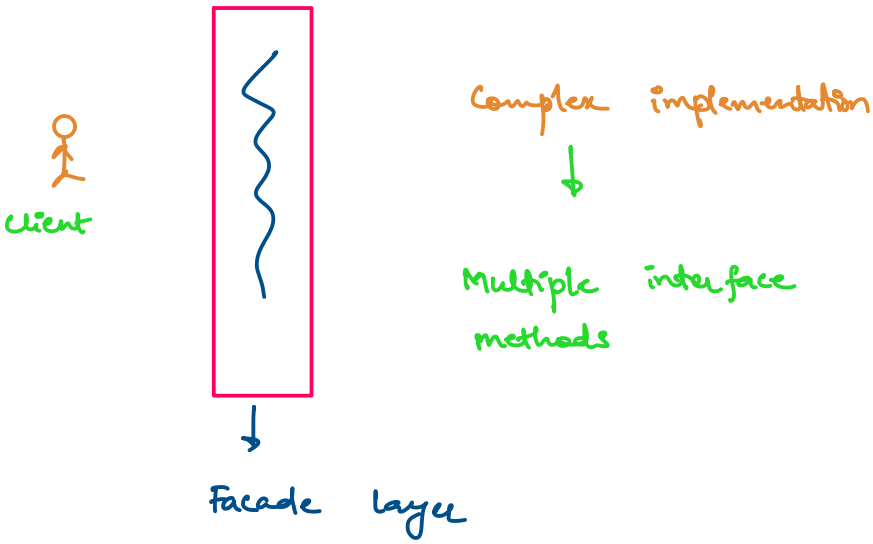
Sunday, 18 September 2022 8:03 PM

Hedgehog concept

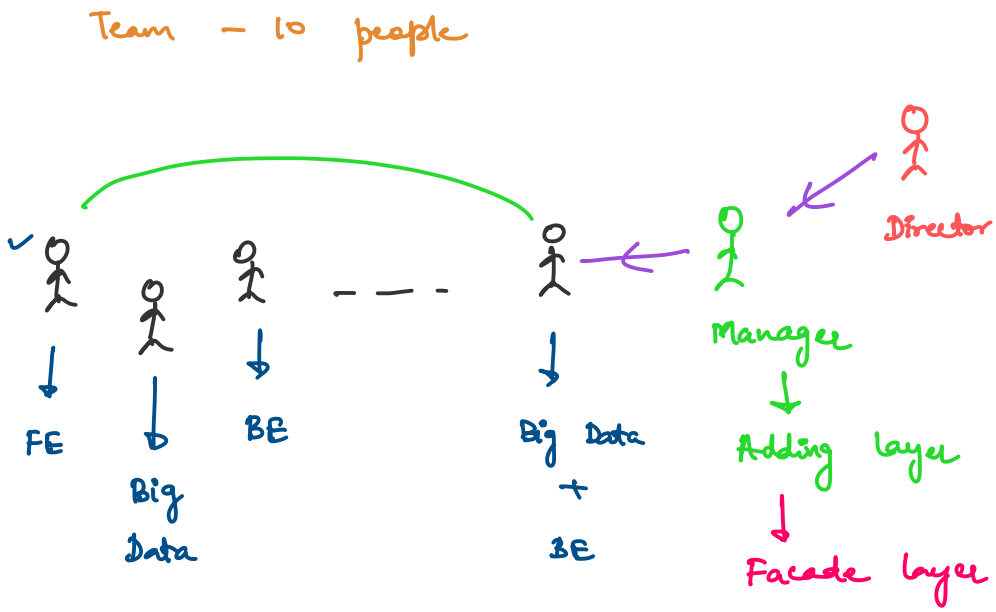
The fox knows many things but the hedgehog knows one big thing.

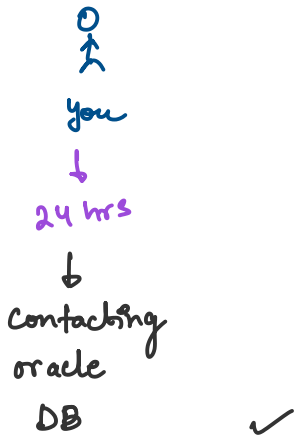


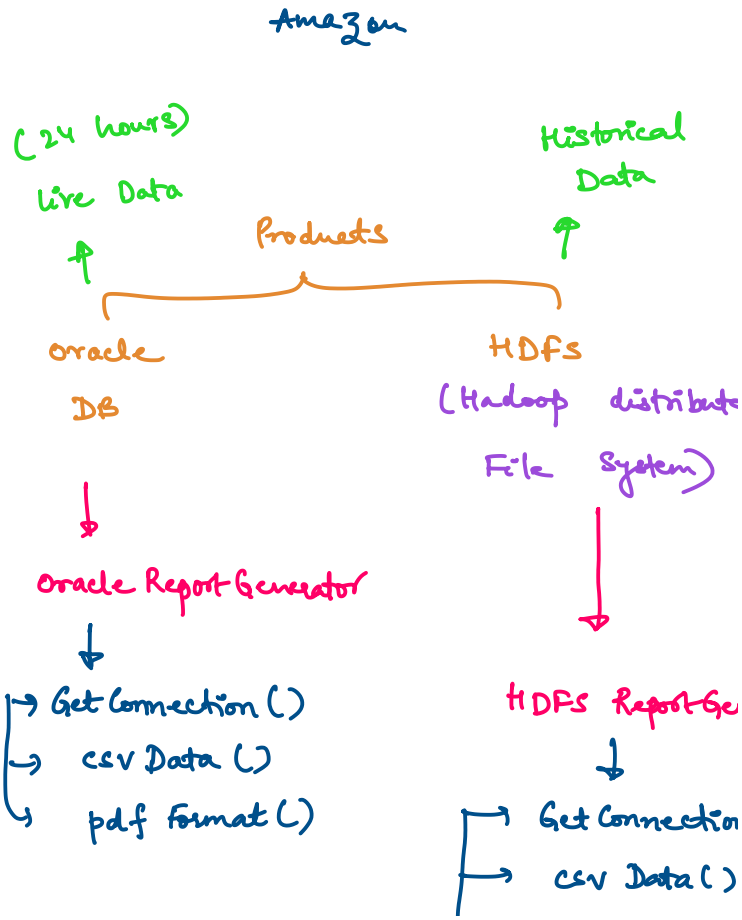
Facade Design pattern



The main aim of facade is to simplify the complex system.

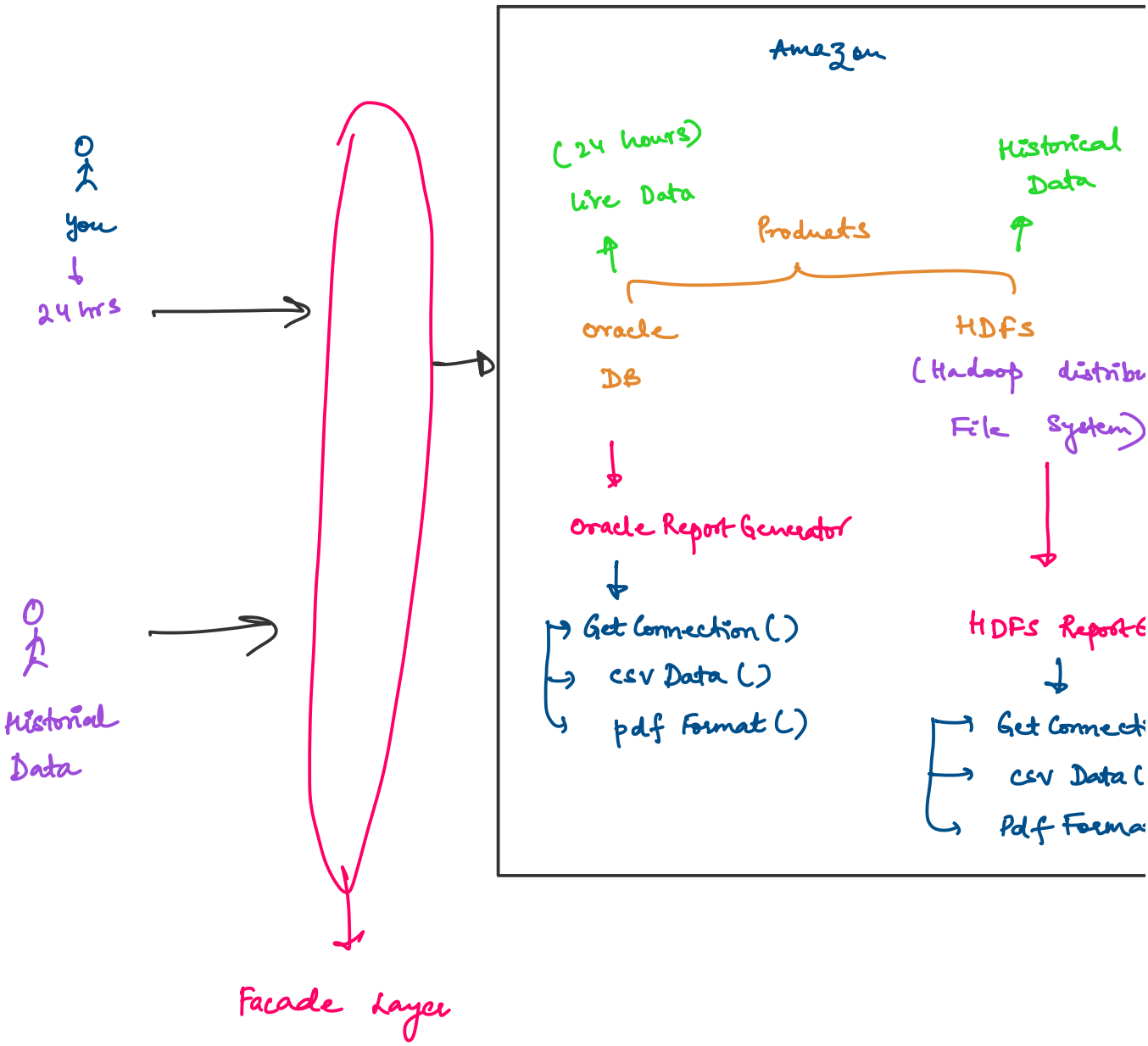






↳ Pdf Format

↳ contacting HDFS



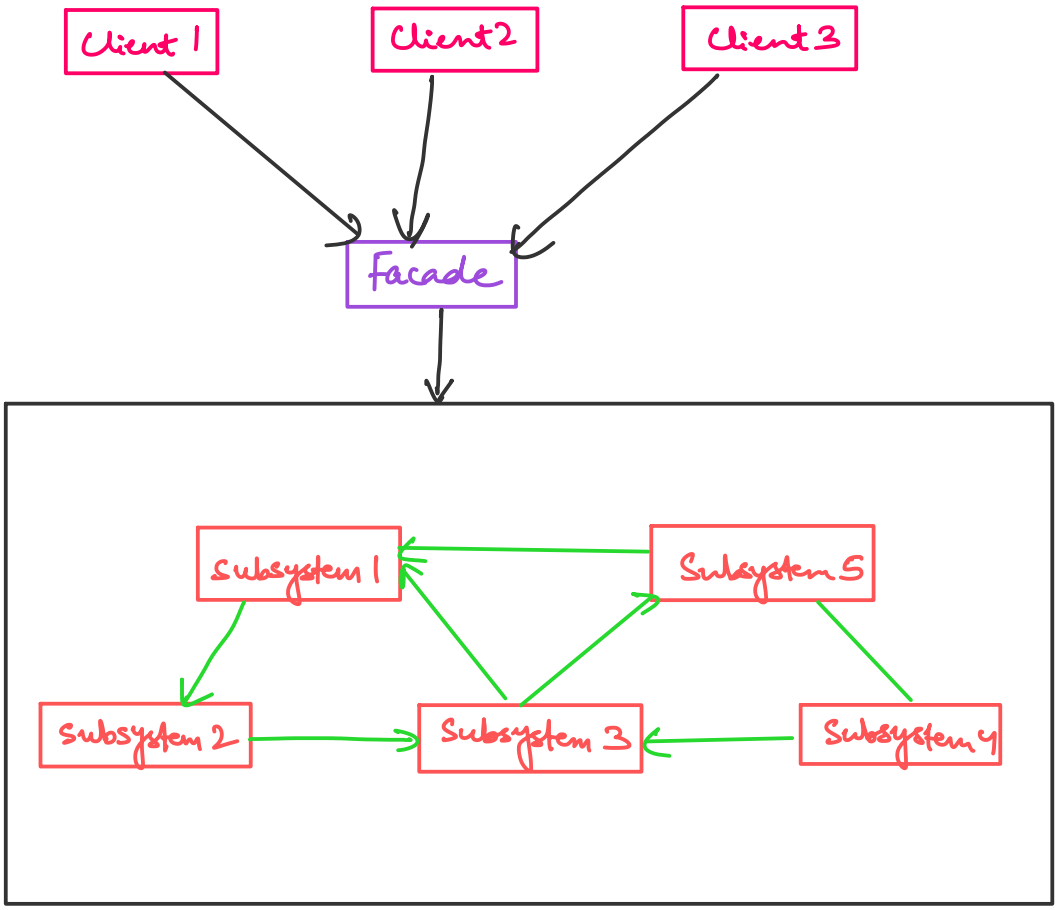
Facade layer is going to provide the required information (or details) to all the customers.

As per the Gang of four, facade design pattern states that we need to provide unified interface to a set of interfaces in a subsystem.

Facade defines a higher level interface that makes the subsystem easier to use.

A facade pattern says that it just provides a unified and simplified interface to set of interfaces in a subsystem, therefore it hides the complexities of the subsystem from the client.

Practically every abstract factory is a type of facade.



Difference between facade pattern vs abstract factory pattern.

A facade is a class or a group of classes hiding internal implementation / services from the user.

An abstract factory encapsulates a group of factories which are used for creating objects, whereas facade can be used to provide abstraction to all kinds of operations, not just creation.

Advantages of facade design pattern

1. It shields the clients from the complexities of the subsystem components.

2. It promotes loose coupling between subsystems and its clients.

Implementation guidelines –

We need to use facade design pattern when –

1. We want to provide a simple interface to a complex subsystem. Subsystems often get more complex as they evolve.

2. There are many dependencies between clients and implementation classes of an abstraction.

3. We want to layer the subsystems, use facade to define an entry point to each subsystem level.

Uses of facade design pattern –

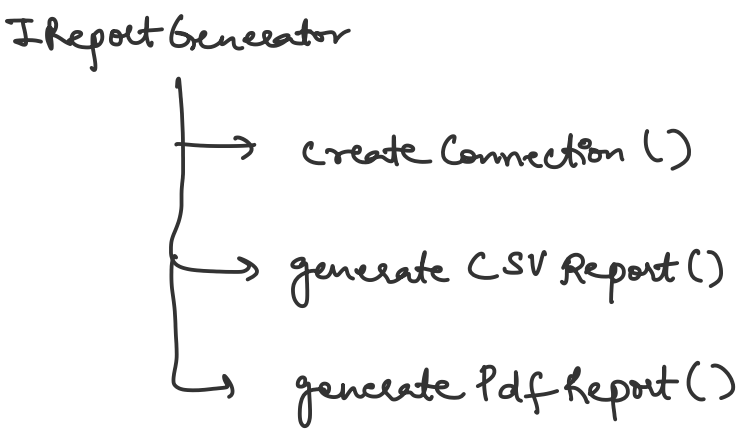
1. It simplifies the complex implementation.

2. Helps in the situations when

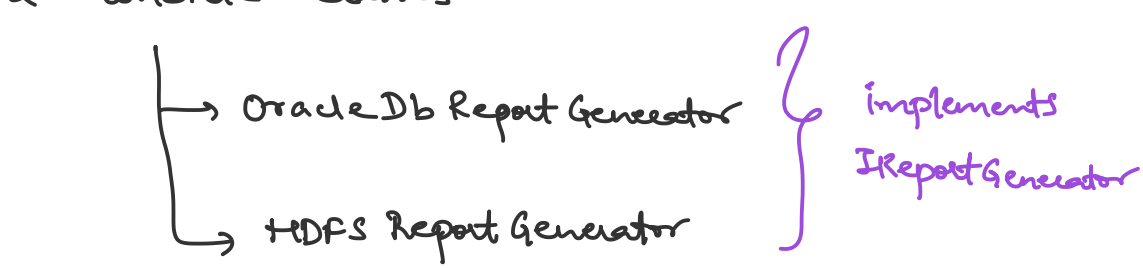
4. Helps in the [illegible] when you have multiple dependencies.

Steps –

1. Create an interface.



2. Create concrete classes implementing the IReport Interface.

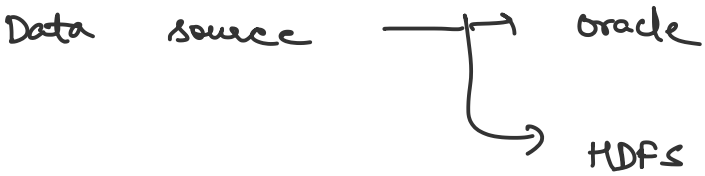


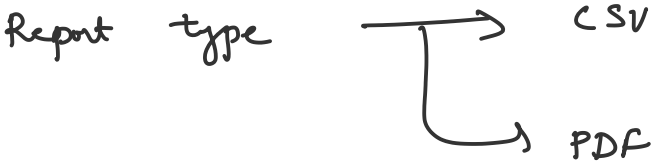
3. Create a facade class.

↳ ReportFacade

↓

Generating report based upon data source type and report type.





4. Use facade to provide support to the client.

↳ main method.

```
package FacadeDesignPattern;

public interface IReportGenerator {
   public void createConnection();
   public void generateCsvReport();
   public void generatePdfReport();
```

```
}
```

```
package FacadeDesignPattern;

public class OracleDbReportGenerator implements IReportGenerator {

    @Override
    public void createConnection() {
       System.out.println("Creating connection with oracle DB");
    }

    @Override
    public void generateCsvReport() {
       System.out.println("Generating the CSV report from oracle DB");
    }

    @Override
    public void generatePdfReport() {
       System.out.println("Generating the PDF report from oracle DB");
    }
}
```

```
package FacadeDesignPattern;

public class HdfsReportGenerator implements IReportGenerator {
   @Override
   public void createConnection() {
      System.out.println("Creating connection with HDFS");
   }

   @Override
   public void generateCsvReport() {
      System.out.println("Generating the CSV report from HDFS");
   }

   @Override
   public void generatePdfReport() {
      System.out.println("Generating the PDF report from HDFS");
```

```
    }
}
```

```
package FacadeDesignPattern;

public class ReportFacade {
   public void generateReport(String dataSourceType, String reportType)
   {
      if(dataSourceType == "HDFS")
      {
         HdfsReportGenerator hdfsReportGenerator = new HdfsReportGenerator();
         hdfsReportGenerator.createConnection();

         if(reportType == "CSV")
         {
            hdfsReportGenerator.generateCsvReport();
         }
         else
         {
            hdfsReportGenerator.generateCsvReport();
         }
      }
      else
      {
         OracleDbReportGenerator oracleDbReportGenerator = new OracleDbReportGenerato
         oracleDbReportGenerator.createConnection();

         if(reportType == "CSV")
         {
            oracleDbReportGenerator.generateCsvReport();
         }
         else
         {
            oracleDbReportGenerator.generateCsvReport();
         }
      }
   }
}
```

```
package FacadeDesignPattern;

public class Client {
   public static void main(String[] args) {
      ReportFacade reportFacade = new ReportFacade();

      //know historical reports in PDF format
      reportFacade.generateReport("HDFS", "PDF");
      System.out.println();

      //know the report from last24 hours in CSV format
      reportFacade.generateReport("OracleDb", "CSV");
   }
}
```