Core- JAVA

Select Statement

Write a program to retrieve all the records present in the Book table and display those records in the specified format using the SELECT select.

Strictly adhere to the Object-Oriented specifications given in the problem statement. All class names, attribute names and method names should be the same as specified in the problem statement.

Create a class named **Book** with the following private attributes/variables.

Data type	Variable
Integer	id
String	title
String	category
String	author
Double	price

Include appropriate **getters** and **setters**.
Include **default** and **parameterized constructors** in the order **public Book(Integer id, String title, String category, String author, Double price)**

Create a class **BookdDAO** with the following method.

Method Name	Description
ArrayList<Book> listBooks()	This method is used to list the all the books in the database.

Create a class **DBConnection** with following method.

Method	Description
public static Connection getConnection()	This method is used to connect the java application with oracle database. Here register the JDBC driver for the application, configure the database properties(fetch from oracle.properties) and return the connection object.

Create a class **Main** with main method. In the method, create instances of the above classes and test the above classes.

Use the below format to print the details in table :
System.out.format(**"%-5s %-20s %-20s %-10s %s\n"**,"Id","Title","Category","Author","Price");

oracle.properties :

db.url = jdbc:oracle:thin:@localhost:1521:xe
db.username = root
db.password = student

Use the below code to retrieve the connection details from oracle.properties to establish connection

```
ResourceBundle rb = ResourceBundle.getBundle("oracle");
String url = rb.getString("db.url");
String user = rb.getString("db.username");
String pass = rb.getString("db.password");
```

Table Properties:

create table book(
id number(10) not null,
title VARCHAR2(45) not null,
category VARCHAR2(45) not null,
author VARCHAR2(45) not null,
price binary_double not null,
primary key(id));

Download the oracle jar file in the below link.
Oracle jar

Sample Input and Output:

List of Books

Id	Title	Category	Author	Price
1	Vampire Dairy	Fiction	Chetan	150.0
2	Harry potter	Witchcraft	Rowling	450.0

```
import java.sql.*;
import java.util.ResourceBundle;
public class DBConnection {
    public static Connection getConnection() throws ClassNotFoundException, SQLException {
        ResourceBundle rb = ResourceBundle.getBundle("oracle");
        String url = rb.getString("db.url");
        String username = rb.getString("db.username");
        String password = rb.getString("db.password");
        //fill your code here
        Class.forName("oracle.jdbc.driver.OracleDriver");

        //step2 define connection url
        Connection con=DriverManager.getConnection(url,username,password);

        return con;
    }
}
```

```
public class Book {
    Integer id;
    String title;
    String category;
    String author;
    Double price;
    public Book() {
        super();
        // TODO Auto-generated constructor stub
    }
    public Book(Integer id, String title, String category, String author, Double price) {
        super();
        this.id = id;
        this.title = title;
        this.category = category;
        this.author = author;
        this.price = price;
    }
```

```
    public Integer getId() {
        return id;
    }
    public void setId(Integer id) {
        this.id = id;
    }
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    public String getCategory() {
        return category;
    }
    public void setCategory(String category) {
        this.category = category;
    }
    public String getAuthor() {
        return author;
    }
    public void setAuthor(String author) {
        this.author = author;
    }
    public Double getPrice() {
        return price;
    }
    public void setPrice(Double price) {
        this.price = price;
    }

}
```

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;

public class BookDAO {
    public ArrayList<Book> listBooks() throws ClassNotFoundException, SQLException{
        ArrayList<Book> bookList = new ArrayList<Book>();
        Connection con = DBConnection.getConnection();

    try{
        Statement st = con.createStatement();

        ResultSet r = st.executeQuery("select * from book");
        while(r.next()){
 bookList.add(new Book(r.getInt(1),r.getString(2),r.getString(3),r.getString(4), r.getDouble(5)));
        }
        con.close();
    }catch(Exception e){
        System.out.println(e);
```

```
    }
        return bookList;
    }
}
```

```
import java.util.*;
import java.io.*;
public class Main {
    public static void main(String[] args) throws Exception
    {
        ArrayList<Book> list = new ArrayList<Book>();
        BookDAO b = new BookDAO();

        list = b.listBooks();
        System.out.println("List of Books");
        System.out.format("%-5s %-20s %-20s %-10s %s\n","Id","Title","Category","Author","Price");
        for(Book book :list){
            System.out.format("%-5s %-20s %-20s %-
10s %s\n",book.getId(), book.getTitle(),book.getCategory(),book.getAuthor(),book.getPrice());
        }

    }
}
```

Update details of Travel Classes

Write a java program to update the details of Travel class available in the database and display the list of travel class details in the descending order of names using JDBC drivers.

[Note: Strictly adhere to the object-oriented specifications given as a part of the problem statement. Follow the naming conventions as mentioned. Create separate classes in separate files.]

Create a class **TravelClass** with the following attributes.

Data Type	Variable Name
String	name
String	description

Include appropirate **getters**, **setters**, **default** and **parameterized constructors** for the above class

Create a class **TravelClassDAO** with a following method

Method name	Description
ArrayList<TravelClass> listAllTravelClassess()	This method retrieves the list of travel classes available in the database in the descending order of the travel class name and returns the same.
void updateDetail(String name, String description)	This method update the given description into the database for the given travel class name.

Create a class **DBConnection** with following method.

Method	Description

public static Connection getConnection()	This method is used to connect the java application with oracle database. Here register the JDBC driver for the application, configure the database properties(fetch from oracle.properties) and return the connection object.

Create a class **Main** with main method and call the methods of TravelClassDAO and display the list as shown in the main method.

Use the below format to print the details in table :
System.out.format(**"%-25s %s\n",**"Name","Description");

oracle.properties :

db.url = jdbc:oracle:thin:@localhost:1521:xe
db.username = root
db.password = student

Use the below code to retrieve the connection details from oracle.properties to establish connection

ResourceBundle rb = ResourceBundle.getBundle("oracle");
String url = rb.getString("db.url");
String user = rb.getString("db.username");
String pass = rb.getString("db.password");

Table Properties:

create table travel_class(
id number(10) not null,
name VARCHAR2(45) not null,
description CLOB not null,
primary key(id)
);

Download the oracle jar file in the below link.
Oracle jar

Sample Input and Output:
Enter the name of TravelClass :
Economy Class
Enter the description to update :
Lowest travel class of seating in flight travel.
Updated List of Travel Classes
Name Description
Premium Economy Class Positioning in price, comfort, and amenities, this travel class is leveled between economy class and business class.
Economy Class Lowest travel class of seating in flight travel.
Business Class Intermediate level of service between economy class and first class.

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;

public class TravelClassDAO {
    ArrayList<TravelClass> listAllTravelClassess() throws ClassNotFoundException, SQLException {

        ArrayList<TravelClass> travelClassList= new ArrayList<TravelClass>();
```

```
        try{
            Connection con = DBConnection.getConnection();
            Statement st = con.createStatement();

            ResultSet r = st.executeQuery("Select name, description From travel_class Order by nam
e DESC");
            while(r.next()){
                travelClassList.add(new TravelClass(r.getString(1), r.getString(2)));
            }
            con.close();
        }catch(Exception e){
            System.out.println(e);
        }
        return travelClassList;
    }
    public void updateDetail(String name, String description) throws ClassNotFoundException, SQLEx
ception {
        try{
            Connection con = DBConnection.getConnection();
            String query =String.format("Update travel_class set description = '%s' where name = '
%s' ",description,name);
            PreparedStatement p = con.prepareStatement(query);
            p.execute();
            con.close();
        }catch(Exception e){
             System.out.println(e);
        }
    }
}
```

```
import java.io.*;
import java.sql.SQLException;
import java.util.ArrayList;


public class Main {
    public static void main(String args[]) throws Exception{
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Enter the name of TravelClass :");
        String name = br.readLine();
        System.out.println("Enter the description to update :");
        String description = br.readLine();
        TravelClassDAO dao = new TravelClassDAO();
        dao.updateDetail(name, description);

        System.out.println("Updated List of Travel Classes");
        System.out.format("%-25s %s\n","Name","Description");
        ArrayList<TravelClass> list = new ArrayList<TravelClass>();
        list = dao.listAllTravelClassess();
        for(TravelClass t : list){
             System.out.format("%-25s %s\n",t.getName(), t.getDescription());
        }
    }
}
```

Insert new Airport

There is conjusted space in airport because of airplane's increased count.So Government decided to build new airport in their country.So help the government by implementing the scenario using a program. Add the details of new airport for a country.

[Note: Strictly adhere to the object-oriented specifications given as a part of the problem statement. Follow the naming conventions as mentioned. Create separate classes in separate files.]

Create a class **Airport** with the following attributes:

Data type	Variable name
String	iataAirportCode
String	name
String	city
String	country

Include appropirate **getters**, **setters**, **default** and **parameterized constructors** for the above class

Create a class **AirportDAO** with a following methods

Method name	Description
ArrayList<Airport> listAirport()	This method will return the list of airports in the database
void insertAirport(Airport airportIns)	This method will insert the new airport into the database

Create a class **DBConnection** with following method.

Method	Description
public static Connection getConnection()	This method is used to connect the java application with oracle database. Here register the JDBC driver for the application, configure the database properties(fetch from oracle.properties) and return the connection object.

Create a class **Main** with main method. In the method, create instances of the above classes and test the above classes.

Use the below format to print the details in table :
System.out.format(**"%-10s %-40s %-10s %s\n"**,"IATA Code","Name","City","Country");

oracle.properties :

db.url = jdbc:oracle:thin:@localhost:1521:xe
db.username = root
db.password = student

Use the below code to retrieve the connection details from oracle.properties to establish connection

ResourceBundle rb = ResourceBundle.getBundle("oracle");
String url = rb.getString("db.url");
String user = rb.getString("db.username");
String pass = rb.getString("db.password");

Table Properties:

```
create table airport(
id number(10) GENERATED ALWAYS AS IDENTITY not null,
iata_airport_code VARCHAR2(45) not null,
name VARCHAR2(45) not null,
city VARCHAR2(45) not null,
country_name VARCHAR2(45) not null,
primary key(id));
```

Download the oracle jar file in the below link.
Oracle jar

Sample Input and Output:
Enter the Airport Code :
MDZ
Enter the Airport name :
Francisco Gabriell International Airport
Enter the City :
Mendoza
Enter the Country name :
Arizona

IATA Code	Name	City	Country
COK	Cochin International Airport	Cochin	India
BNE	Brisbane Airport	Brisbane	Australia
SXR	Srinagar International Airport	Srinagar	India
MDZ	Francisco Gabriell International Airport	Mendoza	Arizona

```
public class Airport {
    String iataAirportCode,name,city,country;

    public Airport() {
        super();
        // TODO Auto-generated constructor stub
    }

    public Airport(String iataAirportCode, String name, String city, String country) {
        super();
        this.iataAirportCode = iataAirportCode;
        this.name = name;
        this.city = city;
        this.country = country;
    }

    public String getIataAirportCode() {
        return iataAirportCode;
    }

    public void setIataAirportCode(String iataAirportCode) {
        this.iataAirportCode = iataAirportCode;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
```

```
    }

    public String getCity() {
        return city;
    }

    public void setCity(String city) {
        this.city = city;
    }

    public String getCountry() {
        return country;
    }

    public void setCountry(String country) {
        this.country = country;
    }



}
```

```
import java.sql.*;
import java.util.ArrayList;


public class AirportDAO {
    public void insertAirport(Airport airportIns) throws ClassNotFoundException, SQLException{

        try{
            String query = String.format("insert into airport(iata_airport_code, name, city, count
ry_name) VALUES('%s','%s','%s','%s')",airportIns.getIataAirportCode(),airportIns.getName(),airport
Ins.getCity(),airportIns.getCountry());

            Connection con = DBConnection.getConnection();
            PreparedStatement p = con.prepareStatement(query);


            p.execute();
            con.close();

        }catch(Exception e){
            System.out.println(e);
        }
    }
    public ArrayList<Airport> listAirport() throws ClassNotFoundException, SQLException{
        ArrayList<Airport> list = new ArrayList<Airport>();
        try{
        Connection con = DBConnection.getConnection();

        Statement st = con.createStatement();

        ResultSet r = st.executeQuery("Select iata_airport_code, name, city, country_name from air
port");
        while(r.next()){
```

```
            list.add(new Airport(r.getString(1), r.getString(2), r.getString(3), r.getString(4)));
            }
            con.close();
        }catch(Exception e){
            System.out.println(e);
        }

        return list;
    }
}
```

```
import java.io.*;
import java.util.ArrayList;

public class Main {
    public static void main(String args[]) throws Exception{
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String countryName,airportCode,airportName,city;
        System.out.println("Enter the Airport Code :");
        airportCode = br.readLine();
        System.out.println("Enter the Airport name :");
        airportName = br.readLine();
        System.out.println("Enter the City :");
        city = br.readLine();
        System.out.println("Enter the Country name :");
        countryName = br.readLine();

        Airport airport = new Airport(airportCode, airportName, city, countryName);
        AirportDAO a = new AirportDAO();
        a.insertAirport(airport);

        ArrayList<Airport> list = new ArrayList<Airport>();
        list = a.listAirport();
        System.out.format("%-10s %-40s %-10s %s\n","IATA Code","Name","City","Country");
        for(Airport at : list){
            System.out.format("%-10s %-40s %-
10s %s\n",at.getIataAirportCode() ,at.getName() ,at.getCity() ,at.getCountry());
        }

    }
}
```

User-Delete

The users who enter into 20 Ideas for Vision 2020 must be active and must keep posting their ideas.If the administrator finds out that the user is not active the admin can delete the user record from the table. Help the admin to delete the user record from the table.

[Note: Strictly adhere to the object-oriented specifications given as a part of the problem statement. Follow the naming conventions as mentioned. Create separate classes in separate files.]

Create a **User** with following private attributes.

Data Type	Variable
Integer	id
String	name
String	email
String	password
Integer	age
String	role
Date	createdDate
String	status

Include appropriate **getters**, **setters**, **default** and **parameterized constructors** for the above class.

Create a class **UserDAO** with following methods

Method Name	Method Description
ArrayList<User> listUsers()	This method returns the list of users available in the database
void deleteUser(Integer id)	This method will delete the user from the database using the given id

Create a class DBConnection with following method

Method	Description
public static Connection getConnection()	This method is used to connect the java application with oracle database. Here register the JDI driver for the application,configure the database properties(fetch from oracle.properties) and return the connection object.

Create a class **Main** with main method and call the methods of UserDAO and display the list as shown.

Output Format:
Use Java Format Specifier to display the user details
System.out.format("%-15s %-15s %-15s %-15s %-15s %-15s %-15s %s\n","Id","Name","Email","Password","Age","Role","CreatedDate","Status");

Table Properties:

```
create table "user"(
id number(10) GENERATED ALWAYS AS IDENTITY not null,
name varchar2(50) not null,
email varchar2(50) not null,
password varchar2(50) not null,
age number(10) not null,
role varchar2(50) not null,
created_date date not null,
status varchar2(50) not null,
primary key(id));
```

oracle.properties :

```
db.url = jdbc:oracle:thin:@localhost:1521:xe
db.username = root
db.password = student
```

Use the below code to retrieve the connection details from oracle.properties to establish connection

```
ResourceBundle rb = ResourceBundle.getBundle("oracle");
String url = rb.getString("db.url");
```

String user = rb.getString("db.username");
String pass = rb.getString("db.password");

Download the oracle jar file in the below link.
Oracle jar

Sample Input and Output
[All text in bold are input and the remaining are output]
Before the Delete

Id	Name	Email	Password	Age	Role	CreatedDate	Status
1	Asha	ash@a.com	as123	18	user	2015-10-13	Approved
2	Rahul	rh@a.com	rh@123	15	user	2015-10-14	Approved
3	Ravi	rv@a.com	rv@98	20	user	2015-10-14	pending

Enter the Id :
1
After the Delete

Id	Name	Email	Password	Age	Role	CreatedDate	Status
2	Rahul	rh@a.com	rh@123	15	user	2015-10-14	Approved
3	Ravi	rv@a.com	rv@98	20	user	2015-10-14	pending

```
import java.util.Date;
import java.text.*;
public class User {
    Integer id;
    String name;
    String email;
    String password;
    Integer age;
    String role;
    Date createdDate;
    String status;

    public User() {
        super();
        // TODO Auto-generated constructor stub
    }

    public User(Integer id, String name, String email, String password, Integer age, String role,
Date createdDate,
            String status) {
        super();
        this.id = id;
        this.name = name;
        this.email = email;
        this.password = password;
        this.age = age;
        this.role = role;
        this.createdDate = createdDate;
        this.status = status;
    }

    public Integer getId() {
        return id;
    }
```

```java
    public void setId(Integer id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public Integer getAge() {
        return age;
    }

    public void setAge(Integer age) {
        this.age = age;
    }

    public String getRole() {
        return role;
    }

    public void setRole(String role) {
        this.role = role;
    }

    public Date getCreatedDate() {
        return createdDate;
    }

    public void setCreatedDate(Date createdDate) {
        this.createdDate = createdDate;
    }

    public String getStatus() {
        return status;
    }
```

```
    public void setStatus(String status) {
        this.status = status;
    }
    public String toString(){
        SimpleDateFormat sdf = new SimpleDateFormat("YYYY-MM-dd");
        return String.format("%-15s %-15s %-15s %-15s %-15s %-15s %-
15s %s",id,name,email,password,age,role,sdf.format(createdDate),status);
    }
}
```

```
import java.text.*;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.sql.*;
public class UserDAO{

    public ArrayList<User> listUsers() throws Exception{
        ArrayList<User> userList = new ArrayList<User>();
        //fill your code here
      Connection con = DBConnection.getConnection();
      Statement st = con.createStatement();
      ResultSet rs = st.executeQuery("select * from \"user\"");
    //   SimpleDateFormat sdf = new SimpleDateFormat("dd-MON-YYYY");
      while(rs.next()) {
      userList.add(new User(rs.getInt(1),rs.getString(2),rs.getString(3),rs.getString(4),rs.getInt
(5),rs.getString(6),rs.getDate(7),rs.getString(8)));
      }
      return userList;
    }
    public void deleteUser(Integer id) throws Exception{
        //fill your code here
        Connection con = DBConnection.getConnection();
      Statement st = con.createStatement();
       int updateCount=st.executeUpdate("delete from  \"user\" where id ="+id);

    }

}
```

```
import java.util.*;
import java.io.*;
import java.text.SimpleDateFormat;

public class Main{

    public static void main(String [] args) throws Exception{
        //fill your code here
        Scanner sc = new Scanner(System.in);
        System.out.println("Before the Delete");
        System.out.format("%-15s %-15s %-15s %-15s %-15s %-15s %-
15s %s\n","Id","Name","Email","Password","Age","Role","CreatedDate","Status");
        UserDAO udao = new UserDAO();
        List<User> userList = udao.listUsers();
        for(User x : userList) System.out.println(x);
```

```
        System.out.println("Enter the Id :");
        udao.deleteUser(sc.nextInt());
        System.out.println("After the Delete");
        System.out.format("%-15s %-15s %-15s %-15s %-15s %-15s %-
15s %s\n","Id","Name","Email","Password","Age","Role","CreatedDate","Status");

        userList = udao.listUsers();
        for(User x : userList) System.out.println(x);
    }

}
```

Create a class **Department** with the following private attributes,

Attribute	Datatype
departmentName	String
staff	Staff

Create a class **Staff** with the following private attributes,

Attribute	Datatype
staffName	String
designation	String

Include the following method in the **Department** class,

Method	Description
public displayStaff()	This method displays "**staffName** is working in the **departmentName** department as **desi**

Create a driver class **Main**, in the main method get the inputs from user, create the objects and call the methods.

Input and Output format:
Refer to sample Input and Output for formatting specifications.

Sample Input and Output:
[All text in bold corresponds to the input and rest corresponds to the output]

Enter the name of the staff:
Jane
Enter the staff designation:
Associate Professor
Enter the department name:
Physics
Jane is working in the Physics department as Associate Professor

```
public class Staff {

    private String staffName;

    private String designation;

    Staff(String staffName,String designation){
```

```
        this.staffName = staffName;

        this.designation = designation;


    }

    public String getStaffName() {
        return staffName;
    }


    public String getDesignation() {
        return designation;
    }


}
```

```
public class Department extends Staff{

    private String departmentName;

    Department(String staffName,String designation, String departmentName){
        super(staffName,designation);
        this.departmentName = departmentName;

    }

    public void displayStaff() {

        System.out.println(super.getStaffName()+" is working in the "+this.departmentName+" depart
ment as "+super.getDesignation());

    }

}
```

```
import java.util.Scanner;
public class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);     //oops i-design (1)

        System.out.println("Enter the name of the staff:");

        String name = sc.nextLine();

        System.out.println("Enter the staff designation:");

        String designation = sc.nextLine();

        System.out.println("Enter the department name:");

        String department_name = sc.nextLine();
```

```
        Department dept = new Department(name,designation,department_name);

        dept.displayStaff();
    }
}
```

Inheritance in Java is same as that of inheritance in real Life. A class which inherits another class obtains all the latter's attributes and methods. The former is called Child class whilst the latter is called Parent class. This phenomenon would be very promising in applications dealing with multiple classes that are constituted by similar or more likely same attributes. You 'll get to know the importance of inheritance from the following problem. All type of accounts in a bank have common attributes which can be inherited from an Account class.

Create a class Account with the following protected attributes

Attributes	Datatype
accName	String
accNo	String
bankName	String

Include appropriate getters and setters.

Include the following protected methods.

Method	Description
void display()	This protected method displays the account details

Create a class CurrentAccount with following private attributes which extends Account class

Attributes	Datatype
tinNumber	String

Create default constructor and a parameterized constructor with arguments in order **CurrentAccount(String accName,String accNo,String bankName,String tinNumber)**.
Include appropriate getters and setters.

Include the following public methods.

Method	Description
void display()	This method calls the super class display(). This public method displays the TIN number. Call this method with the reference of base class.

Create a class SavingsAccount with following private attributes which extends Account class

Attributes	Datatype
orgName	String

Create default constructor and a parameterized constructor with arguments in order **SavingsAccount(String accName,String accNo,String bankName,String orgName)**.
Include appropriate getters and setters.

Include the following public methods.

Method	Description
void display()	This method calls the super class display(). This public method displays the Organisation Name. Call this method with the reference of base class.

Create a driver class named **Main** to test the above class.

Note:
Strictly adhere to the Object-Oriented Specifications given in the problem statement.All class names, attribute names and method names should be the same as specified in the problem statement.

Input Format:

The first input corresponds to choose current or savings account
The next line consists of account name,account number,bank name,org name or tin number (according to chosen account type)

Output Format

The output consists of account details and TIN number or Organisation name
Refer sample output for formatting specifications.

Sample Input/Output-1:

Choose Account Type
1.Savings Account
2.Current Account
1
Enter Account details in comma separated(Account Name,Account Number,Bank Name,Organisation Name)
Morsh,033808020000879,Baroda,Amphisoft
Account Name:Morsh
Account Number:033808020000879
Bank Name:Baroda
Organisation Name:Amphisoft

Sample Input/Output-2:

Choose Account Type
1.Savings Account
2.Current Account
2
Enter Account details in comma separated(Account Name,Account Number,Bank Name,TIN Number)
Krish,131231451,ICICI,798902
Account Name:Krish
Account Number:131231451
Bank Name:ICICI
TIN Number:798902

```
public class Account {
    protected String accName;
    protected String accNo;
    protected String bankName;
    Account(){

    }
```

```
    Account(String accName, String accNo, String bankName){
        this.accName = accName;
        this.accNo = accNo;
        this.bankName = bankName;
    }

    public String getAccName() {
        return accName;
    }
    public void setAccName(String accName) {
        this.accName = accName;
    }
    public String getAccNo() {
        return accNo;
    }
    public void setAccNo(String accNo) {
        this.accNo = accNo;
    }
    public String getBankName() {
        return bankName;
    }
    public void setBankName(String bankName) {
        this.bankName = bankName;
    }

    protected void display() {
        System.out.println("Account Name:"+this.accName);
        System.out.println("Account Number:"+this.accNo);
        System.out.println("Bank Name:"+this.bankName);
    }

}
```

```
public class SavingsAccount extends Account{
    private String orgName;
    SavingsAccount(){

    }
    SavingsAccount(String accName, String accNo, String bankName, String orgName){
        super(accName, accNo, bankName);
        this.orgName = orgName;
    }
    public String getOrgName() {
        return orgName;
    }
    public void setOrgName(String orgName) {
        this.orgName = orgName;
    }

    public void display() {
        super.display();
        System.out.println("Organisation Name:"+this.orgName);
    }
}
```

```java
public class CurrentAccount extends Account{
    private String tinNumber;
    CurrentAccount(){

    }
    CurrentAccount(String accName, String accNo, String bankName, String tinNumber){
        super(accName, accNo, bankName);
        this.tinNumber = tinNumber;
    }
    public String getTinNumber() {
        return tinNumber;
    }
    public void setTinNumber(String tinNumber) {
        this.tinNumber = tinNumber;
    }

    public void display() {
        super.display();
        System.out.println("TIN Number:"+this.tinNumber);
    }
}
```

```java
import java.util.Scanner;
public class Main {
    public static void main(String[] args){
        Scanner s= new Scanner(System.in);
        System.out.println("Choose Account Type\n1.Savings Account\n2.Current Account");
        //String ch=s.nextLine();
        int n= s.nextInt();

        switch(n){
            case 2:
                System.out.println("Enter Account details in comma separated(Account Name,Account
Number,Bank Name,TIN Number)");
                String c=s.nextLine();
                String[] cur=c.split(",");
                CurrentAccount current=new CurrentAccount(cur[0],cur[1],cur[2],cur[3]);
                current.display();
                break;
            case 1:
                System.out.println("Enter Account details in comma separated(Account Name,Account
Number,Bank Name,Organisation Name)");
                String m=s.nextLine();
                String[] sav=m.split(",");


                SavingsAccount savings=new SavingsAccount(sav[0],sav[1],sav[2],sav[3]);
                savings.display();
                break;
        }
    }
}
```

Abstract Class - FundTransfer

Let's try an application like a fund transfer for our larger application. So, in fund transfer, there are 3 types of NEFT/IMPS/RGTS. We can create an abstract class FundTransfer. And extend it in the child classes. Create an abstract method transfer and implement it in all the child classes.

Strictly adhere to the Object-Oriented specifications given in the problem statement. All class names, attribute names and method names should be the same as specified in the problem statement.

Create an abstract class **FundTransfer** with following private attributes,

Attributes	Datatype
accountNumber	String
balance	Double

Include the following methods in the class **FundTransfer.**

Method	Description
Boolean validate(Double transfer)	To check if the accountNumber is 10 digits, the transfer amount is non- If all cases are satisfied then return true, if not return false
Boolean transfer(Double transfer)	It is an abstract method with no definition

Create a class **NEFTTransfer** which extends **FundTransfer.**

Include the following method in the class **NEFTTransfer**.

Method	Description
Boolean transfer(Double transfer)	To check whether the transfer amount+5% of the transfer am If then subtracts transfer amount and 5% service charge from balance and r

Create a class **IMPSTransfer** which extends **FundTransfer.**

Include the following method in the class **IMPSTransfer.**

Method	Description
Boolean transfer(Double transfer)	To check whether transfer amount+2% of transfer amoun If then subtracts transfer amount and 2% service charge from balance and r

Create a class **RTGSTransfer** which extends **FundTransfer.**

Include the following method in the class **RTGSTransfer.**

Method	Description
Boolean transfer(Double transfer)	To check whether the transfer amount is If then subtracts transfer amount from balance and return true, if no

Include appropriate getters/setters, constructors with super() to create objects. Write a driver class **Main** to test them.

Input format:
Refer to sample Input and Output for the details and for the formatting specifications.

Output format:

Print **"Transfer occurred successfully"** in the main method. If the transfer function returns true.
Print "**Account number or transfer amount seems to be wrong**" in the main method. If validate function returns false.
Print "**Transfer could not be made**" in the main method. If the transfer function returns false.
Refer to sample Input and Output for formatting specifications.

Note: All Texts in bold corresponds to the input and rest are output

Sample Input and Output 1:

```
Enter your account number:
1234567890
Enter the balance of the account:
10000
Enter the type of transfer to be made:
1.NEFT
2.IMPS
3.RTGS
1
Enter the amount to be transferred:
2000
Transfer occurred successfully
Remaining balance is 7900.0
```

Sample Input and Output 2:

```
Enter your account number:
1111111
Enter the balance of the account:
10000
Enter the type of transfer to be made:
1.NEFT
2.IMPS
3.RTGS
2
Enter the amount to be transferred:
1000
Account number or transfer amount seems to be wrong
```

Sample Input and Output 3:

```
Enter your account number:
1234567890
Enter the balance of the account:
50000
Enter the type of transfer to be made:
1.NEFT
2.IMPS
3.RTGS
3
Enter the amount to be transferred:
7500
Transfer could not be made
```

```
public abstract class FundTransfer {
    //write your code here
    private String accountNumber;
    private Double balance;

    public FundTransfer(String acc, Double balance)
    {
        this.accountNumber = acc;
        this.balance = balance;
    }
    public FundTransfer()
    {

    }

    public String getAccountNumber()
    {
        return accountNumber;
    }
    public void setAccountNumber()
    {
        this.accountNumber = accountNumber;
    }
    public Double getBalance()
    {
        return balance;
    }
    public void setBalance(Double balance)
    {
        this.balance = balance;
    }

    Boolean validate(Double amount)
    {
        int n = accountNumber.length();
        if(n==10 && amount<balance && amount>0)
            return true;
        return false;
    }
     abstract Boolean transfer(Double transfer);
}
```

```
class NEFTTransfer extends FundTransfer {
    //write your code here
    public NEFTTransfer(String acc,Double balance)
    {
        super(acc,balance);
    }
    public NEFTTransfer()
    {

    }

    Boolean transfer(Double trans)
    {
        double am = trans + 0.05*trans;
```

```
        if(am<super.getBalance()){
            double bal = super.getBalance() - am;
            super.setBalance(bal);
            return true;
        }
        return false;
    }
}
```

```
public class IMPSTransfer extends FundTransfer{
    //write your code here
    IMPSTransfer(String acc,Double balance)
    {
        super(acc,balance);
    }
    IMPSTransfer()
    {

    }
    Boolean transfer(Double transfer)
    {
        double am = transfer + 0.02*transfer;
        if(am<super.getBalance()){
            double bal = super.getBalance() - am;
            super.setBalance(bal);
            return true;
        }
        return false;
    }
}
```

```
public class RTGSTransfer  extends FundTransfer{
    //write your code here
    RTGSTransfer(String acc,Double balance)
    {
        super(acc,balance);
    }
    RTGSTransfer()
    {

    }
    Boolean transfer(Double transfer)
    {

        if(transfer>10000){
            double bal = super.getBalance() - transfer;
            super.setBalance(bal);
            return true;
        }
        return false;
    }
}
```

```
import java.util.*;
public class Main {
```

```
    public static void main(String args[]) throws Exception {
        //write your code here
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter your account number:");
        String acc = sc.next();
        System.out.println("Enter the balance of the account:");
        Double balance = sc.nextDouble();
        System.out.println("Enter the type of transfer to be made:");
        System.out.println("1.NEFT\n" +"2.IMPS\n" +"3.RTGS");
        int choice = sc.nextInt();
        System.out.println("Enter the amount to be transferred:");
        Double amount = sc.nextDouble();
        FundTransfer ft;

        if(choice == 1)
        {
            FundTransfer nt = new NEFTTransfer(acc,balance);
            boolean val=nt.validate(amount);
            if(val==true)
                {
                    boolean ans = nt.transfer(amount);
            if(ans == true)
                {
                    System.out.println("Transfer occurred successfully");
                    System.out.println("Remaining balance is "+nt.getBalance());
                }
                else
                    System.out.println("Transfer could not be made");
                }
            else
                System.out.println("Account number or transfer amount seems to be wrong");

        }
        else if(choice == 3)
        {
            RTGSTransfer rt = new RTGSTransfer(acc,balance);
            boolean val=rt.validate(amount);
            if(val==true)
                {
                    boolean ans = rt.transfer(amount);
            if(ans ==true)
                {
                    System.out.println("Transfer occurred successfully");
                    System.out.println("Remaining balance is "+rt.getBalance());
                }
                else
                    System.out.println("Transfer could not be made");
                }
            else
                System.out.println("Account number or transfer amount seems to be wrong");

        }
        else if (choice ==2){
            IMPSTransfer it = new IMPSTransfer(acc,balance);
            boolean val=it.validate(amount);
            if(val==true)
```

```
                {
                    boolean ans = it.transfer(amount);
            if(ans == true)
                {
                    System.out.println("Transfer occurred successfully");
                    System.out.println("Remaining balance is "+it.getBalance());
                }
                else
                    System.out.println("Transfer could not be made");
                }
            else
                System.out.println("Account number or transfer amount seems to be wrong");

        }
    }
}
```

super() method

So far you have learned about the basics of inheritance and created 2 child classes for a parent class and displayed the details. Now let's go for simple manipulation along with superclass. This will be needed in our application as some class share common attributes so they can be grouped as child classes of some superclass.

To try this let's create a parent class **Event** with following attributes,

Attributes	**Datatype**
name	String
detail	String
type	String
ownerName	String
costPerDay	Double

Then create child class **Exhibition** that extends Event with the following attribute,

Attributes	**Datatype**
noOfStall	Integer

And create another child class **StageEvent** that extends Event with the following attribute,

Attributes	**Datatype**
noOfSeats	Integer

Add suitable constructor (with super() if necessary) and getters/setters for the classes.

Get the starting and ending date of the event from the user and calculate the total cost for the event. Then calculate GST for the event according to the event type.

GST is 5% for Exhibition and 15% for StageEvent.

Input format:
The first line of input is an integer which corresponds to the choice of event.
The second line of input is the details of the event in the CSV format **(name, detail, type, owner name,costPerDay,noOfStall)** for the Exhibition.
(name, detail, type, owner name,costPerDay,noOfSeats) for the stage event.
The next line is the starting date of the event.

The last line of the input is the ending date of the event.

Output format:
The output is the GST to be paid for the event.
Refer to sample input/output for other further details and format of the output.

[All Texts in bold corresponds to the input and rest are output]
Sample Input/Output 1:

Enter your choice:
1.Exhibition event
2.Stage event
1
Enter the details of exhibition:
Science Fair,Exciting experiments,Fair,John,10000.00,10
Enter the starting date of the event:
03-01-2018
Enter the ending date of the event:
06-01-2018
The GST to be paid is Rs.1500.0

Sample Input/Output 2:

Enter your choice:
1.Exhibition event
2.Stage event
2
Enter the details of stage event:
Movie Award Function,Awards for all category,Award function,Joe,100000,10000
Enter the starting date of the event:
07-01-2018
Enter the ending date of the event:
09-01-2018
The GST to be paid is Rs.30000.0

```
public class Event {
    private String name;
    private String detail;
    private String type;
    private String ownerName;
    private double costPerDay;
    private double gst;


    public Event(String name, String detail, String type, String ownerName, double costPerDay, dou
ble gst) {
        this.name = name;
        this.detail = detail;
        this.type = type;
        this.ownerName = ownerName;
        this.costPerDay = costPerDay;
        this.gst = gst;
    }

    public double getCostPerDay() {
```

```
        return costPerDay;
    }

    public double getGst() {
        return gst;
    }

    protected double getGST(long days) {
        return (this.getGst() * this.getCostPerDay() * days)/100;
    }
}
```

```
public class StageEvent extends Event {
    private int noOfSeats;

    public StageEvent(String name, String detail, String type, String ownerName, double costPerDay
, int noOfSeats) {
        super(name, detail, type, ownerName, costPerDay, 15);
        this.noOfSeats = noOfSeats;
    }
}
```

```
public class Exhibition extends Event {
    private int noOfStalls;

    public Exhibition(String name, String detail, String type, String ownerName, double costPerDay
, int noOfStalls) {
        super(name, detail, type, ownerName, costPerDay, 5);
        this.noOfStalls = noOfStalls;
    }
}
```

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) throws ParseException {
        Scanner in = new Scanner(System.in);

        System.out.println("Enter your choice:" +
                "\n1.Exhibition event" +
                "\n2.Stage event");
        int choice = in.nextInt();
        in.nextLine();

        System.out.print("Enter the details of ");
        if (choice == 1)
            System.out.println("exhibition:");
        else
            System.out.println("stage event:");
        String[] input = in.nextLine().split(",");

        System.out.println("Enter the starting date of the event:");
```

```
        String start = in.nextLine();

        System.out.println("Enter the ending date of the event:");
        String end = in.nextLine();

        SimpleDateFormat sdf = new SimpleDateFormat("dd-MM-yyyy");
        Date startDate = sdf.parse(start);
        Date endDate = sdf.parse(end);

        long milliDiff = endDate.getTime() - startDate.getTime();
        long days = milliDiff / 86400000;

        Event e;
        if(choice == 1)
            e = new Exhibition(input[0], input[1], input[2], input[3], Double.parseDouble(input[4]
), Integer.parseInt(input[5]));
        else
            e = new StageEvent(input[0], input[1], input[2], input[3], Double.parseDouble(input[4]
), Integer.parseInt(input[5]));

        System.out.println("The GST to be paid is Rs." + e.getGST(days));
    }
}
```

Interface

The Interface defines a rule that any classes that implement it should override all the methods. Let's implement Interface in our application. We'll start simple, by including display method in the Stall interface. Now all types of stalls that implement the interface should override the method.

Strictly adhere to the Object-Oriented specifications given in the problem statement. All class names, attribute names and method names should be the same as specified in the problem statement.

Create an interface **Stall** with the following method

Method	Description
void display()	abstract method.

Create a class **GoldStall** which implements **Stall** interface with the following private attributes

Attribute	Datatype
stallName	String
cost	Integer
ownerName	String
tvSet	Integer

Create default constructor and a parameterized constructor with arguments in order **GoldStall(String stallName, Integer cost, String ownerName, Integer tvSet)**. Include appropriate getters and setters.

Include the following method in the class **GoldStall**

Method	Description
void display()	To display the stall name, cost of the stall, owner name and the number of tv sets.

Create a class **PremiumStall** which implements **Stall** interface with following private attributes

Attribute	Datatype
stallName	String
cost	Integer
ownerName	String
projector	Integer

Create default constructor and a parameterized constructor with arguments in order **PremiumStall(String stallName, Integer cost, String ownerName, Integer projector)**. Include appropriate getters and setters.

Include the following method in the class **PremiumStall.**

Method	Description
void display()	To display the stall name, cost of the stall, owner name and the number of projectors.

Create a class **ExecutiveStall** which implements **Stall** interface with following private attributes

Attribute	Datatype
stallName	String
cost	Integer
ownerName	String
screen	Integer

Create default constructor and a parameterized constructor with arguments in order **ExecutiveStall(String stallName, Integer cost, String ownerName, Integer screen)**. Include appropriate getters and setters.

Include the following method in the class **ExecutiveStall.**

Method	Description
void display()	To display the stall name, cost of the stall, owner name and the number of screens.

Create a driver class named **Main** to test the above class.

Input Format:
The first input corresponds to choose the stall type.
The next line of input corresponds to the details of the stall in CSV format according to the stall type.

Output Format:
Print "**Invalid Stall Type**" if the user has chosen the stall type other than the given type
Otherwise, display the details of the stall.
Refer to sample output for formatting specifications.

Note: All Texts in bold corresponds to the input and rest are output

Sample Input and Output 1:

```
Choose Stall Type
1)Gold Stall
2)Premium Stall
3)Executive Stall
1
```

Enter Stall details in comma separated(Stall Name,Stall Cost,Owner Name,Number of TV sets)
The Mechanic,120000,Johnson,10
Stall Name:The Mechanic
Cost:120000.Rs
Owner Name:Johnson
Number of TV sets:10

Sample Input and Output 2:

ChooseStall Type
1)Gold Stall
2)Premium Stall
3)Executive Stall
2
Enter Stall details in comma separated(Stall Name,Stall Cost,Owner Name,Number of Projectors)
Knitting plaza,300000,Zain,20
Stall Name:Knitting plaza
Cost:300000.Rs
Owner Name:Zain
Number of Projectors:20

Sample Input Output 3:

ChooseStall Type
1)Gold Stall
2)Premium Stall
3)Executive Stall
3
Enter Stall details in comma separated(Stall Name,Stall Cost,Owner Name,Number of Screens)
Fruits Hunt,10000,Uber,7
Stall Name:Fruits Hunt
Cost:10000.Rs
Owner Name:Uber
Number of Screens:7

Sample Input Output 4:

ChooseStall Type
1)Gold Stall
2)Premium Stall
3)Executive Stall
4
Invalid Stall Type

```
public interface Stall {
    void display();
}
```

```
public class GoldStall implements Stall {
    private String stallName;
    private int cost;
    private String ownerName;
    private int tvSet;

    public GoldStall() {
```

```
        this.stallName = this.ownerName = "";
        this.cost = this.tvSet = 0;
    }


    public void setStallName(String stallName) {
        this.stallName = stallName;
    }

    public void setCost(int cost) {
        this.cost = cost;
    }

    public void setOwnerName(String ownerName) {
        this.ownerName = ownerName;
    }

    public void setTvSet(int tvSet) {
        this.tvSet = tvSet;
    }

    public GoldStall(String stallName, int cost, String ownerName, int tvSet) {
        this.stallName = stallName;
        this.cost = cost;
        this.ownerName = ownerName;
        this.tvSet = tvSet;
    }

    @Override
    public void display() {
        System.out.println("Stall Name:" + this.stallName
                + "\nCost:" + this.cost + ".Rs"
                + "\nOwner Name:" + this.ownerName
                + "\nNumber of TV sets:" + this.tvSet);
    }

    public String getStallName() {
        return stallName;
    }

    public int getCost() {
        return cost;
    }

    public String getOwnerName() {
        return ownerName;
    }

    public int getTvSet() {
        return tvSet;
    }
}
```

```
public class PremiumStall implements Stall{
    private String stallName;
    private int cost;
```

```
    private String ownerName;
    private int projector;

    public PremiumStall() {
        this.stallName = this.ownerName = "";
        this.cost = this.projector = 0;
    }


    public PremiumStall(String stallName, int cost, String ownerName, int projector) {
        this.stallName = stallName;
        this.cost = cost;
        this.ownerName = ownerName;
        this.projector = projector;
    }

    public void setStallName(String stallName) {
        this.stallName = stallName;
    }

    public void setCost(int cost) {
        this.cost = cost;
    }

    public void setOwnerName(String ownerName) {
        this.ownerName = ownerName;
    }

    public void setProjector(int projector) {
        this.projector = projector;
    }

    @Override
    public void display() {
        System.out.println("Stall Name:" + this.stallName
                + "\nCost:" + this.cost + ".Rs"
                + "\nOwner Name:" + this.ownerName
                + "\nNumber of Projectors:" + this.projector);
    }

    public String getStallName() {
        return stallName;
    }

    public int getCost() {
        return cost;
    }

    public String getOwnerName() {
        return ownerName;
    }

    public int getProjector() {
        return projector;
    }
}
```

```java
public class ExecutiveStall implements Stall{
    private String stallName;
    private int cost;
    private String ownerName;
    private int screen;

    public ExecutiveStall() {
        this.stallName = this.ownerName = "";
        this.cost = this.screen = 0;
    }

    public ExecutiveStall(String stallName, int cost, String ownerName, int screen) {
        this.stallName = stallName;
        this.cost = cost;
        this.ownerName = ownerName;
        this.screen = screen;
    }

    @Override
    public void display() {
        System.out.println("Stall Name:" + this.stallName
                           + "\nCost:" + this.cost + ".Rs"
                           + "\nOwner Name:" + this.ownerName
                           + "\nNumber of Screens:" + this.screen);
    }

    public String getStallName() {
        return stallName;
    }

    public int getCost() {
        return cost;
    }

    public String getOwnerName() {
        return ownerName;
    }

    public int getScreen() {
        return screen;
    }

    public void setStallName(String stallName) {
        this.stallName = stallName;
    }

    public void setCost(int cost) {
        this.cost = cost;
    }

    public void setOwnerName(String ownerName) {
        this.ownerName = ownerName;
    }

    public void setScreen(int screen) {
        this.screen = screen;
```

```
    }
}
```

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);

        System.out.println("Choose Stall Type\n" +
                "1)Gold Stall\n" +
                "2)Premium Stall\n" +
                "3)Executive Stall");
        int choice = in.nextInt();

        if (choice == 1) {
            in.nextLine();
            System.out.println("Enter Stall details in comma separated(Stall Name,Stall Cost,Owner
 Name,Number of TV sets)");
            String[] input = in.nextLine().split(",");
            GoldStall stall = new GoldStall();
            stall.setStallName(input[0]);
            stall.setCost(Integer.parseInt(input[1]));
            stall.setOwnerName(input[2]);
            stall.setTvSet(Integer.parseInt(input[3]));
            stall.display();
        } else if (choice == 2) {
            in.nextLine();
            String[] input;
            System.out.println("Enter Stall details in comma separated(Stall Name,Stall Cost,Owner
 Name,Number of Projectors)");
            input = in.nextLine().split(",");
            PremiumStall stall = new PremiumStall();
            stall.setStallName(input[0]);
            stall.setCost(Integer.parseInt(input[1]));
            stall.setOwnerName(input[2]);
            stall.setProjector(Integer.parseInt(input[3]));
            stall.display();
        } else if (choice == 3) {
            in.nextLine();
            String[] input;
            System.out.println("Enter Stall details in comma separated(Stall Name,Stall Cost,Owner
 Name,Number of Screens)");
            input = in.nextLine().split(",");
            ExecutiveStall stall = new ExecutiveStall();
            stall.setStallName(input[0]);
            stall.setCost(Integer.parseInt(input[1]));
            stall.setOwnerName(input[2]);
            stall.setScreen(Integer.parseInt(input[3]));
            stall.display();
        } else {
            System.out.println("Invalid Stall Type");
        }
    }
}
```

Using contains() and trim() method

Sunil has finished most of his application for the fair, it's time to focus on minor details that went wrong during a test run of his application in this module. Accidentally some gibberish text with leading and trailing got copied to the clipboard and got pasted in some of the text documents. But, still. He has the gibberish text with him, he can manually load each document, and find the text and delete it. Think it will take ages, no he can think of a time saver. He can use his programming skills, he can load each document in a program and find in which files the text got copied. Even though, he feels difficult to identify it. So let him.

write a program to find whether the gibberish text is present in the string. Assume text of the document is given as the input to the program.

Create a driver class called **Main**. In the Main method, obtain the inputs from the console and prompt whether the gibberish text is present in the main text.

Problem Constraints:

Use contains() and trim()

Input and Output format:

The first line of the input consists of the text of the document.
The second line of the input consists of a string that has to be found in the given text.

Note: All text in bold corresponds to the input and rest corresponds to the output.

Sample Input and output 1:

Enter the text from the document
One fine morning, a minister from Emperor Akbar's court had gathered in the assembly hall. He informed the Emperor that all his valuables had been stolen by a thief the previous night.
Enter the string to be found in the data
stolen
String is found in the document

Sample Input and output 2:

Enter the text from the document
One fine morning, a minister from Emperor Akbar's court had gathered in the assembly hall. He informed the Emperor that all his valuables had been stolen by a thief the previous night.
Enter the string to be found in the data
Birbal
String is not found in the document

```
import java.util.*;
class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the text from the document");
        String str = sc.nextLine();
        System.out.println("Enter the string to be found in the data");
        String word = sc.nextLine();
        word = word.trim();
        if(str.contains(word)) {
            System.out.println("String is found in the document");
        }
```

```
        else {
            System.out.println("String is not found in the document");
        }
    }

}
```

String Builder

StringBuilder class is used to create mutable (modifiable) string. StringBuilders are used when the String has to be modified constantly. We recently observed that the availability of items for the stall is hard to know. So we are gonna program that requirement here. Get the availability of the items from the vendors. They are providing the list separated by "$". So use string builder and display the details of the items along with their availability. The list provided by vendors only have the available numbers of the item. We don't want that, so just display whether the item is available or not.

Create a class **Item** with following private attributes

Attributes	Datatype
name	String
type	String
cost	Integer
availableQuantity	Integer

Include appropriate getters and setters
Create default constructor and a parameterized constructor with arguments in order **Item(String name, String type, Integer cost, Integer availableQuantity)**.

Create a driver class named **Main** to test the above class.

Note:
Strictly adhere to the Object-Oriented Specifications given in the problem statement.All class names, attribute names and method names should be the same as specified in the problem statement.

Input Format:

The first line input corresponds to the number of items 'n'.
The next 'n' line of inputs corresponds to the item details in the format of **(Item Name$Item Type$Item Cost$Item Availability)**.
Refer sample input for formatting specifications.

Output Format:

The output consists Item details in the CSV format.If the available quantity of the item is 0 then make it as "**Not Available**" else "**Available**".
Refer sample output for formatting specifications.

Sample Input/Output-1:

Enter the number of items:
3
Enter the item details in the format(Item Name$Item Type$Item Cost$Item Availability)
Wallets$Leather$1200$10
Notebooks$Papers$200$0
Headphones$Electronics$800$3
Items:
Wallets,Leather,1200,Available
Notebooks,Papers,200,Not Available
Headphones,Electronics,800,Available

```java
public class Item {
    private String name;
    private String type;
    private int cost;
    private int availableQuantity;

    public Item(String name, String type, int cost, int availableQuantity) {
        this.name = name;
        this.type = type;
        this.cost = cost;
        this.availableQuantity = availableQuantity;
    }

    public String getName() {
        return name;
    }

    public String getType() {
        return type;
    }

    public int getCost() {
        return cost;
    }

    public int getAvailableQuantity() {
        return availableQuantity;
    }

    @Override
    public String toString() {
        return name + ',' +
                type + ',' +
                cost + ',';
    }
}
```

```java
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);

        System.out.println("Enter the number of items:");
        int n = in.nextInt();
        in.nextLine();

        System.out.println("Enter the item details in the format(Item Name$Item Type$Item Cost$Ite
m Availability)");
        Item[] array = new Item[n];
        for (int i = 0; i < n; i++) {
            String[] input = in.nextLine().split("[$]");
            Item item = new Item(input[0], input[1], Integer.parseInt(input[2]), Integer.parseInt(
input[3]));
            array[i] = item;
```

```
        }

        System.out.println("items:");
        for(int i=0;i<n;i++){
            Item item = array[i];
            StringBuilder sb = new StringBuilder(item.toString());
            if(item.getAvailableQuantity() > 0)
                System.out.println(sb.append("Available"));
            else
                System.out.println(sb.append("Not Available"));
        }
    }
}
```

Date Formats

So far, we have got the dates from the user in MM-dd-yyyy format. but in some reports, we want them in some other formats. So write a program to convert the dates gotten from the user into different formats.

Create a driver class named Main and do all manipulation in the main method.

Input Format:

The first line input corresponds to the date that is to be processed in various date formats.
Refer sample input for formatting specifications.

Output Format:

The output consists of date in the format of ("EEE, MMM d, yy" , "dd.MM.yyyy" , "dd/MM/yyyy").
Refer sample output for formatting specifications.

Sample Input/Output-1:

Enter the date to be formatted:(MM-dd-yyyy)
10-20-1996
Date Format with EEE, MMM d, yy : Sun, Oct 20, 96
Date Format with dd.MM.yyyy : 20.10.1996
Date Format with dd dd/MM/yyyy : 20/10/1996

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);

        System.out.println("Enter the date to be formatted:(MM-dd-yyyy)");
        String input = in.nextLine();

        SimpleDateFormat sdf = new SimpleDateFormat("MM-dd-yyyy");
        SimpleDateFormat sdf1 = new SimpleDateFormat("EEE, MMM d, yy");
        SimpleDateFormat sdf2 = new SimpleDateFormat("dd.MM.yyyy");
        SimpleDateFormat sdf3 = new SimpleDateFormat("dd/MM/yyyy");

        Date date = null;
```

```
        try {
            date = sdf.parse(input);
        } catch (ParseException e) {
            e.printStackTrace();
        }

        System.out.println("Date Format with EEE, MMM d, yy : " + sdf1.format(date) +
                "\nDate Format with dd.MM.yyyy : " + sdf2.format(date) +
                "\nDate Format with dd dd/MM/yyyy : " + sdf3.format(date));
    }
}
```

StringBuilder

For more practice in StringBuilder let's try the exercise below. Given a code, try to change the code to expected format. The first 2 letters correspond to the city code, DH - Delhi MB - Mumbai KL - Kolkata. The digits following the city code correspond to the reference number. In the output code, the city code changes as given below.
DH - DEL
MB - MUM
KL - KOL

Also, need to maintain uniformity in the number of digits in the reference number be 5 digits.
Create a driver class called Main. In the Main method, obtain the input from the console and print the formatted code.
Note: Use StringBuilder

[All text in bold corresponds to the input and rest corresponds to the output]

Sample Input/Output 1:

Enter the code
DH1
Formatted code
DEL00001

Sample Input/Output 2:

Enter the code
MB12345
Formatted code
MUB12345

```
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Map<String, String> map = new HashMap<>();
        map.put("DH", "DEL");
        map.put("MB", "MUB");
        map.put("KL", "KOL");
```

```
        Scanner in = new Scanner(System.in);
        System.out.println("Enter the code");
        String code = in.nextLine();
        StringBuilder sb = new StringBuilder(code);
        String s = sb.substring(0,2);
        int i = Integer.parseInt(code.substring(2));
        sb.replace(0,2,map.get(s));
        sb.replace(3, sb.length(), String.format("%05d", i));

        System.out.println("Formatted code\n" + sb);
    }
}
```

Date Processing --- Compare dates

We realized that there are many events held on different dates and lasts for the different duration. And we want the count of events that are held only for one day so the arrangements of the hall can be made. Now we have a list of events with the name along with starting and ending date in CSV format. So write a program to split them and display the number of 1-day events.

Create class Event with following private attributes

Attributes	Datatype
eventName	String
startDate	Date
endDate	Date

Include appropriate getters and setters.
Create default constructor and a parameterized constructor with arguments in order **Event(String eventName, Date startDate, Date endDate).**

Note:
Strictly adhere to the Object-Oriented Specifications given in the problem statement.All class names, attribute names and method names should be the same as specified in the problem statement.

Create a driver class named **Main** to test the above class.

Input Format:

The first line input corresponds to the number of events 'n'.
The next 'n' lines of input corresponding to the event details in CSV format(Event Name,Start Date,End Date).
Refer sample input for formatting specifications.

Output Format:

The output consists of 1-day event names. If no 1-day event present, then display "**No Events**".
Refer sample output for formatting specifications.

[All text in bold corresponds to input and rest corresponds to output]
Sample Input/Output-1:

Enter the number of Events
3
Enter event details in CSV(Event Name,Start Date,End Date) Date:dd/MM/yyyy
Reunion,28/02/2017,28/02/2017
Party,30/06/2017,05/07/2017
Wedding,23/10/2017,24/10/2017
1-day Events:

Reunion

Sample Input/Output-2:
Enter the number of Events
2
Enter event details in CSV(Event Name,Start Date,End Date) Date:dd/MM/yyyy
Celebration,30/06/1998,31/06/1998
Christmas Party,22/01/2000,25/02/2000
1-day Events:
No Events

```
import java.util.Date;

public class Event {
    private String eventName;
    private Date startDate;
    private Date endDate;

    public Event(String eventName, Date startDate, Date endDate) {
        this.eventName = eventName;
        this.startDate = startDate;
        this.endDate = endDate;
    }

    public Event() {
    }

    public String getEventName() {
        return eventName;
    }

    public void setEventName(String eventName) {
        this.eventName = eventName;
    }

    public Date getStartDate() {
        return startDate;
    }

    public void setStartDate(Date startDate) {
        this.startDate = startDate;
    }

    public Date getEndDate() {
        return endDate;
    }

    public void setEndDate(Date endDate) {
        this.endDate = endDate;
    }
}
```

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
```

```
import java.util.Date;
import java.util.Scanner;
import java.util.concurrent.TimeUnit;

public class Main {
    public static void main(String[] args) throws ParseException {
        Scanner in = new Scanner(System.in);
        ArrayList<String> list = new ArrayList<>();
        System.out.println("Enter the number of Events");
        int n = in.nextInt();
        in.nextLine();
        System.out.println("Enter event details in CSV(Event Name,Start Date,End Date) Date:dd/MM/
yyyy");


        SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
        for (int i = 0; i < n; i++) {
            String[] details = in.nextLine().split(",");
            Date start = sdf.parse(details[1]);
            Date end = sdf.parse(details[2]);
            long noOfDays = TimeUnit.DAYS.convert(end.getTime() - start.getTime(), TimeUnit.DAYS);

            if (noOfDays == 0) {
                list.add(details[0]);
            }
        }

        System.out.println("1-day Events:");
        if (!list.isEmpty())
            for(String s : list)
                System.out.println(s);
        else
            System.out.println("No Events");
    }
}
```

contains() & indexOf() methods in ArrayList

Write a program to get the hall details and store in the ArrayList and search the hall and display it's position details.

Get hall names in the **Main** class and store it an ArrayList.

Input format:

The first line of input is an integer which corresponds to the number ‘n’ of halls.
The n lines of input are the string which corresponds to the hall name.
The last line of input is the string which corresponds to the hall name to be searched.

Output format:

The output is the hall position.
It is the position at which the hall is present in the list starting from 0.
If the hall to be searched is not present in the list, then print "**[Hall name] hall is not found**"
Refer to sample Input and Output for formatting specifications.

[All Texts in bold corresponds to the input and rest are output]
Sample Input and Output 1:

Enter the number of halls:
3
Enter the Hall Name 1
SPK
Enter the Hall Name 2
DFG
Enter the Hall Name 3
TRE
Enter the hall name to be searched:
DFG
DFG hall is found in the list at position 1

Sample Input/Output 2:

Enter the number of halls:
3
Enter the Hall Name 1
SPJ
Enter the Hall Name 2
RWE
Enter the Hall Name 3
HFG
Enter the hall name to be searched:
SPK
SPK hall is not found

```
import java.util.*;
public class Main {
    public static void main(String args[]) throws Exception{
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number of halls:");
        int n = sc.nextInt();
        sc.nextLine();

        ArrayList<String> a = new ArrayList<String>(n);
        for(int i = 0; i<n; i++){
            System.out.println("Enter the Hall Name "+(i+1));
            a.add(sc.nextLine());
        }
        System.out.println("Enter the hall name to be searched:");
        String word = sc.nextLine();

        if(a.contains(word))
            System.out.println(word+" hall is found in the list at position "+a.indexOf(word));
        else
            System.out.println(word+ " hall is not found");
    }
}
```

Email Search

In your application let's dive deep into Set and explore its inbuilt functions. In this problem experiment with containsAll() method. Write a program to search all the email addresses which are given as CSV format.

Create a **Main** class. Obtain email addresses from the user and add them to a Set. At last, get a String that has multiple email addresses in CSV format. Print "**Email addresses are present**" if all email addresses are present in the Set, else print "**Email addresses are not present**".

Input and Output format:
Refer to sample Input and Output for formatting specifications.

Note: All Texts in bold corresponds to the input and rest are output

Sample Input and Output 1:

```
Enter Email address
Merry@gmail.com
Do you want to Continue?(yes/no)
yes
Enter Email address
Peter@yahoo.com
Do you want to Continue?(yes/no)
yes
Enter Email address
Christian@hotmail.com
Do you want to Continue?(yes/no)
yes
Enter Email address
Merry@gmail.com
Do you want to Continue?(yes/no)
no
Enter the email addresses to be searched separated by comma
Merry@gmail.com,Peter@yahoo.com
Email addresses are present
```

Sample Input and Output 2:

```
Enter Email address
Manikandan@yahoo.com
Do you want to Continue?(yes/no)
yes
Enter Email address
bala@google.co.in
Do you want to Continue?(yes/no)
no
Enter the email addresses to be searched separated by comma
bala@google.co.in,jeryy@gmail.com
Email addresses are not present
```

```java
import java.util.*;
public class Main {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        HashSet<String> emails = new HashSet<String>();
        String response = "yes";
        while(response .equals("yes")){
            System.out.println("Enter Email address");
            String em = sc.nextLine();
```

```
            emails.add(em);
            System.out.println("Do you want to Continue?(yes/no)");
            response = sc.nextLine();
        }
        System.out.println("Enter the email addresses to be searched separated by comma");
        String input[] = sc.nextLine().split(",");

        ArrayList<String> email = new ArrayList<String>();
        Collections.addAll(email, input);


        if(emails.containsAll(email))
            System.out.println("Email addresses are present");
        else
            System.out.println("Email addresses are not present");
    }

}
```

Address details are displayed in tabular format.(Use "%-15s %-15s %-15s %-15s %s\n" for formatting Address details.)

[All text in bold corresponds to the input and rest corresponds to the output]
Sample Input/Output:

Enter the number of address
4
Enter the address 1 detail
22nd lane,RR nagar,Chennai,Tamil nadu,600028
Enter the address 2 detail
3rd street,KRK nagar,Visak,Andhrapradesh,745601
Enter the address 3 detail
1/45 8th street,KK nagar,Chennai,Tamil nadu,600021
Enter the address 4 detail
5/15 7th lane,RK nagar,Madurai,Tamil nadu,625001
Enter the city to be searched
Chennai
Line 1 Line 2 City State Pincode
22nd lane RRnagar Chennai Tamilnadu 600028
1/45 8th street KKnagar Chennai Tamilnadu 600021

```
public class Address {
    private String addressLine1;
    private String addressLine2;
    private String city;
    private String state;
    private int pincode;


    public Address(String addressLine1, String addressLine2, String city, String state, int pincod
e) {
        super();
        this.addressLine1 = addressLine1;
```

```
        this.addressLine2 = addressLine2;
        this.city = city;
        this.state = state;
        this.pincode = pincode;
    }

    public String getAddressLine2() {
        return addressLine2;
    }

    public String getCity() {
        return city;
    }

    public String getState() {
        return state;
    }

    public int getPincode() {
        return pincode;
    }

    public String getAddressLine1() {
        return addressLine1;
    }
    public void setAddressLine1(String addressLine1) {
        this.addressLine1 = addressLine1;
    }

}
```

```
public class Address {
    private String addressLine1;
    private String addressLine2;
    private String city;
    private String state;
    private int pincode;


    public Address(String addressLine1, String addressLine2, String city, String state, int pincod
e) {
        super();
        this.addressLine1 = addressLine1;
        this.addressLine2 = addressLine2;
        this.city = city;
        this.state = state;
        this.pincode = pincode;
    }

    public String getAddressLine2() {
        return addressLine2;
    }

    public String getCity() {
        return city;
    }
```

```
    public String getState() {
        return state;
    }

    public int getPincode() {
        return pincode;
    }

    public String getAddressLine1() {
        return addressLine1;
    }
    public void setAddressLine1(String addressLine1) {
        this.addressLine1 = addressLine1;
    }

}
```

User Search

In your application, it is time to experiment with a Set of user-defined Objects. Just like List of objects, Set of objects is also relatively simple.
Consider a Set of bank users. The set contains users who are inactive as well. The bank wants to retain only the Users who are active given a list of active users.

Strictly adhere to the Object-Oriented specifications given in the problem statement. All class names, attribute names and method names should be the same as specified in the problem statement.

Create a **User** class with the following private attributes

Attributes	Datatype
username	String
bankname	String

Include appropriate getters and setters.
Create default constructor and a parameterized constructor with arguments in order **User(String username, String bankname).**

Note:

Override **equals method** to compare the username, to improve the comparison of username use **hashCode method** to check the strings are same.
Override the **Comparable** interface to sort Users in Alphabetical order of Username.

In the **Main** class, Create a Set of user objects from the user input. Now obtain input in CSV format, create User objects and add to a List but with null values in bankName. Now Use **retainAll()** to remove elements in Set that are not in the present in the List.

Input and output format:
Refer to sample Input and Output for formatting specifications.

Sample Input and Output 1:
[All Texts in bold corresponds to the input and rest are output]

Enter number of users:
3
Enter details of user1

Username:
Rikhra
Bank name:
Bank of Baroda
Enter details of user2
Username:
Krish
Bank name:
Lakshmi vilas Bank
Enter details of user3
Username:
Saroja
Bank name:
Karur vysya Bank
Enter username(Expire in one month) seperated by comma
Saroja,Anil,Krish
Users going to expire within a month
User 1
User Name = Krish
Bank Name = Lakshmi vilas Bank
User 2
User Name = Saroja
Bank Name = Karur vysya Bank

```
import java.util.Objects;

public class User implements Comparable<User>{
    private String username;
    private String bankname;

    public User(String username, String bankname) {
        this.username = username;
        this.bankname = bankname;
    }

    public User() {
        this.username = "";
        this.bankname = "";
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getBankname() {
        return bankname;
    }

    public void setBankname(String bankname) {
        this.bankname = bankname;
    }
```

```
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        User user = (User) o;
        return Objects.equals(username, user.username);
    }

    @Override
    public int hashCode() {
        return Objects.hash(username);
    }

    @Override
    public int compareTo(User u) {
        return this.getUsername().compareTo(u.getUsername());
    }

    @Override
    public String toString() {
        return "User{" +
                "username='" + username + '\'' +
                ", bankname='" + bankname + '\'' +
                '}';
    }
}
```

```
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.println("Enter number of users:");
        int n = in.nextInt();
        in.nextLine();

        Set<User> set = new TreeSet<>();
        for (int i = 0; i < n; i++) {
            System.out.println("Enter details of user" + (i + 1));

            System.out.println("Username:");
            String username = in.nextLine();
            System.out.println("Bank name:");
            String bankname = in.nextLine();

            set.add(new User(username, bankname));
        }

        System.out.println("Enter username(Expire in one month) seperated by comma");
        String[] input = in.nextLine().split(",");

        System.out.println("Users going to expire within a month");
        ArrayList<User> toRetain = new ArrayList<>();

        for (int i = 0; i < input.length; i++) {
```

```
            toRetain.add(new User(input[i], null));
        }

        set.retainAll(toRetain);

        int i = 1;
        for (User u : set) {
            System.out.println("User " + (i++));
            System.out.println("User Name = " + u.getUsername());
            System.out.println("Bank Name = " + u.getBankname());
        }
    }
}
```

List removeRange() and addAll()

We have a lot of ArrayList methods. Let's see a few of the important ones namely **removeRange()** and **addAll()**. You will be provided with a template function that returns ArrayList of User objects.

Write a program to get the User in the CSV format and display the details as specified in the Sample Input and Output.

Strictly adhere to the object Oriented specifications given in the problem statement. All class names, attribute names and method names should be the same as specified in the problem statement.

Create a class named **User** class with the following private attributes.

Attributes	Datatype
name	String
contactNumber	String
userName	String
email	String

Include getters and setters for the above attributes and the default and parameterized constructors. Format for the parameterized constructor is **User(String name, String contactNumber, String userName, String email)**

Include the following method in the **User** class.

Method	Description
void display()	This method prints the details of the User object.

Create a class named **UserBO** which extends **ArrayList**.

Include the following methods in the class **UserBO**.

Method	Description
void removeUser(int n1,int n2)	This method accepts from and to index to delete the objects f Use **removeRange()** method.
static UserBO getList()	This method returns an UserBO object that contains pre-coded object va

Create a driver class called **Main**. In the Main method, obtain inputs from the console. Create an object for UserBO and add the User objects to it. For displaying, Iterate through the ArrayList and call the display() method of User class.

Input format:
The first line of input is an integer which corresponds to the number 'n' of user details to be added. The next 'n' line input consists of the user details in the CSV format **[name, contact number, username, email].** Then the last line of inputs consists of two integers that correspond to the from and to index.

Output format:
Use **System.out.printf("%-20s%-20s%-20s%-20s")** for displaying the User details. Refer to sample Input and Output for formatting specifications.

Sample Input and Output:
[All text in bold corresponds to the input and rest corresponds to output]

Enter the number of User details to be added
1
Enter the user 1 detail in csv format
Ram ganesh,9874587457,Ram,ramg@abc.in
Name Contact Number Username Email
mohan Raja 9874563210 mohan mohan@abc.in
arjun Ravi 4324237 arjun arjun@abc.in
Arun kumar 9845671230 arun arun@abc.in
prakash raj 7548921445 prakash raj@abc.in
Ram ganesh 9874587457 Ram ramg@abc.in
Enter the range to be removed from array list
1
3
Name Contact Number Username Email
mohan Raja 9874563210 mohan mohan@abc.in
prakash raj 7548921445 prakash raj@abc.in
Ram ganesh 9874587457 Ram ramg@abc.in

```
public class User implements Comparable<User> {
    private String name;
    private String contactNumber;
    private String userName;
    private String email;

    public User(String name, String contactNumber, String userName, String email) {
        this.name = name;
        this.contactNumber = contactNumber;
        this.userName = userName;
        this.email = email;
    }

    public User() {
        this.name = null;
        this.contactNumber = null;
        this.userName = null;
        this.email = null;
    }

    public String getName() {
        return name;
    }
```

```
    public void setName(String name) {
        this.name = name;
    }

    public String getContactNumber() {
        return contactNumber;
    }

    public void setContactNumber(String contactNumber) {
        this.contactNumber = contactNumber;
    }

    public String getUserName() {
        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    void display() {
        System.out.printf("%-20s%-20s%-20s%-20s\n", name, contactNumber, userName, email);
    }

    @Override
    public int compareTo(User o) {
        return this.getName().compareTo(o.getName());
    }
}
```

```
import java.util.ArrayList;

public class UserBO extends ArrayList<User> {
    public static UserBO getList() {
        UserBO u = new UserBO();
        u.add(new User("mohan Raja", "9874563210", "mohan", "mohan@abc.in"));
        u.add(new User("arjun Ravi", "4324237", "arjun", "arjun@abc.in"));
        u.add(new User("Arun kumar", "9845671230", "arun", "arun@abc.in"));
        u.add(new User("prakash raj", "7548921445", "prakash", "raj@abc.in"));
        return u;
    }

    void removeUser(int n1, int n2) {
        this.removeRange(n1 , n2);
    }

}
```

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.println("Enter the number of User details to be added");
        int n = in.nextInt();
        in.nextLine();

        UserBO list = new UserBO();
        list.addAll(UserBO.getList());

        for (int i = 0; i < n; i++) {
            System.out.println("Enter the user " + (i + 1) + " detail in csv format");
            String[] input = in.nextLine().split(",");
            User u = new User(input[0], input[1], input[2], input[3]);
            list.add(u);
        }

        System.out.println("Name                Contact Number      Username            Email
          ");

        for (User u : list) {
            u.display();
        }

        System.out.println("Enter the range to be removed from array list");
        int n1 = in.nextInt();
        int n2 = in.nextInt();

        list.removeUser(n1, n2);

        System.out.println("Name                Contact Number      Username            Email
          ");
        for (User u : list) {
            u.display();
        }


    }
}
```

Map of Maps

In our application, we gonna implement Map of Map. The value of the map is gonna be another map. This can be helpful in making counts of linked variables. We have the Address class. The address belongs to a city and state. So if we separate addresses based on each state and city maps of maps will be useful. The first map will have the state as key with a map as value. The inner map will have the city as key and count of addresses in that city as value.

Map<String,Map<String,Integer>> is the general format and Map<state,Map<city,count>> is the format for this problem.

Create a driver class **Main** and using the main method get the details, create a map, and display the details.

Input Format:
The first line has the number of address n.
The next n lines have the addresses in CSV format. (area,city,state, pincode)
Refer to the sample Input and Output for the formatting specifications.

Output Format:
Output has State name in the first line and each city name along with the count of address in the city in the next lines. A new line between 2 different states.
The order of output must be in lexicographical order for both state and city.

[All Texts in bold corresponds to the input and rest are output]

Sample Input/Output 1:

Enter the number of address:
3
Enter the address:
Annanagar,Madurai,Tamil Nadu,666666
Enter the address:
Gandhipuram,Coimbatore,Tamil Nadu,123456
Enter the address:
KKnagar,Madurai,Tamil Nadu,678901
Number of entries in city/state wise:

State:Tamil Nadu
City:Coimbatore Count:1
City:Madurai Count:2

Sample Input/Output 2:

Enter the number of address:
5
Enter the address:
Annanagar,Madurai,Tamil Nadu,666666
Enter the address:
Gandhipuram,Coimbatore,Tamil Nadu,123456
Enter the address:
KKnagar,Madurai,Tamil Nadu,678901
Enter the address:
Marathahalli,Banglore,Karnataka,123456
Enter the address:
Electronic City,Banglore,Karnataka,123457
Number of entries in city/state wise:

State:Karnataka
City:Banglore Count:2

State:Tamil Nadu
City:Coimbatore Count:1
City:Madurai Count:2

```
import java.util.TreeMap;
import java.util.Map;
import java.util.Scanner;
```

```
public class Main {
    public static void main(String[] args) {
        // Map<state, Map<city, count>>
        Map<String, Map<String, Integer>> map = new TreeMap<>();

        Scanner in = new Scanner(System.in);
        System.out.println("Enter the number of address:");
        int n = in.nextInt();
        in.nextLine();

        for (int i = 0; i < n; i++) {
            System.out.println("Enter the address:");
            String[] input = in.nextLine().split(",");
            String state = input[2];
            String city = input[1];

            if (map.containsKey(state)) {
                Map<String, Integer> m = map.get(state);
                if (m.containsKey(city))
                    m.replace(city, m.get(city) + 1);
                else
                    m.put(city, 1);
            } else {
                Map<String, Integer> m = new TreeMap<>();
                m.put(city, 1);
                map.put(state, m);
            }
        }

        System.out.println("Number of entries in city/state wise:");
        for (String state : map.keySet()) {
            System.out.println("\nState:" + state);
            Map<String, Integer> m = map.get(state);
            for (String city : m.keySet()) {
                System.out.println("City:" + city + " " + "Count:" + m.get(city));
            }
        }
    }
}

/*

qq,ccc,dd,555
qq,ppp,zz,455
qq,eee,dd,555
qq,eee,dd,555
qq,eee,zz,544
qq,ttt,aa,252


* */
```

NumberFormatException

Let's learn a common type of exception which you would have come across already. When you use BufferedReader to read input you need to parse String it into various datatype like Integer, Double. For

example, If you try to parse a String ("abc") into Integer, it throws NumberFormatException. So let's try to handle this **NumberFormat exception.**

In our application, while acquiring attributes for classes like **ItemType**, this exception may occur. So try to handle it in this program.

Create a class **ItemType** with the following attribute,

Attributes	**Data type**
name	String
deposit	Double
costPerDay	Double

Add appropriate getter/setter, default and parameterized constructor. **public ItemType(String name, Double deposit, Double costPerDay)**.

Override **toString()** and print the details as in the specified format.

Create a **Main** class and test the above class. Display "**The details of the item type are:**" in the main method. Handle the NumberFormatException in the Main Class.

Input and Output format:
Refer to sample Input and Output for formatting specifications.

[All Texts in bold corresponds to the input and rest are output]

Sample Input and Output 1:

```
Enter the Item type details:
Enter the name:
Electronics
Enter the deposit:
1000
Enter the cost per day:
100
The details of the item type are:
Name:Electronics
Deposit:1000.0
Cost Per Day:100.0
```

Sample Input and Output 2:

```
Enter the Item type details:
Enter the name:
Electronics
Enter the deposit:
One thousand
java.lang.NumberFormatException: For input string: "One thousand"
```

```
public class ItemType {
    public String name;
    public Double deposit;
    public Double costPerDay;

    public ItemType(String name, Double deposit, Double costPerDay){
```

```
        this.name = name;
        this.deposit = deposit;
        this.costPerDay = costPerDay;

    }

    @Override
    public String toString(){
        return "The details of the item type are:\nName:"+this.name+"\nDeposit:"+this.deposit+"\nC
ost Per Day:"+this.costPerDay;
    }
}
```

```
import java.util.Scanner;
public class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);

        double deposit =0;
        double costPerDay = 0;
        String d = "";
        String c = "";

        System.out.println("Enter the Item type details:");
        System.out.println("Enter the name:");
        String name = sc.nextLine();

        try{
        System.out.println("Enter the deposit:");
        d= d + sc.nextLine();
        deposit = Double.parseDouble(d);
        }catch(NumberFormatException e){
            System.out.println("java.lang.NumberFormatException: For input string: \"" + d + "\"")
;
            return;
        }

        try{
        System.out.println("Enter the cost per day:");
        c= c+ sc.nextLine();
        costPerDay = Double.parseDouble(c);
        }catch(NumberFormatException e){
            System.out.println("java.lang.NumberFormatException: For input string: \"" + c + "\"")
;
            return;
        }

        ItemType item = new ItemType(name, deposit, costPerDay);
        System.out.println(item);

    }
}
```

ArrayIndexOutOfBoundsException

The next prominent exception which you will see is ArrayIndexOutOfBoundsException. It occurs when the program tries to access the array beyond its size. As we know arrays have fixed size. So when you try to use array beyond its size it throws this exception. Let's try to handle this exception.

Handling this exception will also prove to be good for our application. For example, if there are only 100 seats in the event and the user tries to book the 105th seat, it will throw this exception. So you must handle it to do a specific job.

Create an array of size 100 and assume it as seat array. Get the tickets to be booked from the user and handle any exception that occurs in **Main** Class. At last display all the tickets booked.

Input and Output format:
The first line of input consists of an integer which corresponds to the number of seats to be booked. The next n lines of input consist of the integer which corresponds to the seat number. Refer to sample Input and Output for formatting specifications.

Note: All Texts in bold corresponds to the input and rest are output.

Sample Input and Output 1:

```
Enter the number of seats to be booked:
5
Enter the seat number 1
23
Enter the seat number 2
42
Enter the seat number 3
65
Enter the seat number 4
81
Enter the seat number 5
100
The seats booked are:
23
42
65
81
100
```

Sample Input and Output 2:

```
Enter the number of seats to be booked:
4
Enter the seat number 1
12
Enter the seat number 2
101
java.lang.ArrayIndexOutOfBoundsException: 100
```

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
```

```
        System.out.println("Enter the number of seats to be booked:");
        int n = in.nextInt();
        int s = 0;

        try {
            if (n > 100 || n < 0)
                throw new ArrayIndexOutOfBoundsException();
            int[] array = new int[n];
            for (int i = 0; i < n; i++) {
                System.out.println("Enter the seat number " + (i + 1));
                s = in.nextInt();
                if (s > 100 || s < 0)
                    throw new ArrayIndexOutOfBoundsException();
                else
                    array[i] = s;
            }

            System.out.println("The seats booked are:");
            for (int e : array)
                System.out.println(e);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("java.lang.ArrayIndexOutOfBoundsException: " + (s - 1));
        }
    }
}
```

Duplicate mobile number exception

Write a java program to find the duplicate mobile number using the exception handling mechanism.

Strictly adhere to the Object-Oriented specifications given in the problem statement. All class names, attribute names and method names should be the same as specified in the problem statement.

Create a Class called **ContactDetail** with the following private attributes.

Attributes	Datatype
mobile	String
alternateMobile	String
landLine	String
email	String
address	String

Include getters and setters.
Include default and parameterized constructors.
Format for a parameterized constructor is **ContactDetail(String mobile, String alternateMobile,String landLine, String email, String address)**

Override the **toString()** method to display the Contact details as specified.

Create a class called **ContactDetailBO** with following methods

Method	Description

static void validate(String mobile,String alternateMobile)	This method throws DuplicateMobileNu if the mobile and alternateMobile are the sam

Create a driver class called **Main**. In the Main method, obtain inputs from the user. Validate the mobile and alternateMobile and display the ContactDetail if no exception occurs else handle the exception.

Pass the exception message as "**Mobile number and alternate mobile number are same**". If mobile and alternateMobile are the same.

Input and Output format:
Refer to sample Input and Output for formatting specifications.

Note: All text in bold corresponds to the input and rest corresponds to the output.

Sample Input and Output 1:

Enter the contact details
9874563210,9874563210,0447896541,johndoe@abc.in,22nd street kk nagar chennai
DuplicateMobileNumberException: Mobile number and alternate mobile number are same

Sample Input and Output 2:

Enter the contact details
9874563210,9876543211,0447896541,johndoe@abc.in,22nd lane RR nagar kovai
Mobile:9874563210
Alternate mobile:9876543211
LandLine:0447896541
Email:johndoe@abc.in
Address:22nd lane RR nagar kovai

```
public class ContactDetail {
    String mobile;
    String alternateMobile;
    String landLine;
    String email;
    String address;

    public ContactDetail() {

    }
    public ContactDetail(String mobile, String alternateMobile, String landLine, String email, Str
ing address) {
        this.mobile = mobile;
        this.alternateMobile = alternateMobile;
        this.landLine = landLine;
        this.email = email;
        this.address = address;
    }

    public String getMobile() {
        return mobile;
    }

    public void setMobile(String mobile) {
        this.mobile = mobile;
```

```
    }

    public String getAlternateMobile() {
        return alternateMobile;
    }

    public void setAlternateMobile(String alternateMobile) {
        this.alternateMobile = alternateMobile;
    }

    public String getLandLine() {
        return landLine;
    }

    public void setLandLine(String landLine) {
        this.landLine = landLine;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    @Override
    public String toString() {
        return "Mobile:" + this.mobile +
                "\nAlternate mobile:" + this.alternateMobile +
                "\nLandLine:" + this.landLine +
                "\nEmail:" + this.email +
                "\nAddress:" + this.address;
    }
}
```

```
public class ContactDetailBO {
        //your code here
    public static void validate(String mobile,String AlternateMobile) throws DuplicateMobileNumber
Exception {
        if (mobile.equals(AlternateMobile))
            throw new DuplicateMobileNumberException();
    }
}
```

```
public class DuplicateMobileNumberException extends Exception{
    @Override
```

```
    public String toString(){
        return "DuplicateMobileNumberException: Mobile number and alternate mobile number are same
";
    }
}
```

```
import java.io.*;
public class Main {
    public static void main(String[] args)throws Exception{
        //Your code here
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Enter the contact details");
        String st=br.readLine();
        String[] input=st.split(",");
        // fill the code
        String mobile = input[0];
        String alternateMobile = input[1];
        ContactDetailBO cbo = new ContactDetailBO();
        try {
            cbo.validate(mobile, alternateMobile);
            ContactDetail contact = new ContactDetail(mobile, alternateMobile, input[2], input[3],
 input[4]);
            System.out.println(contact);
        } catch (DuplicateMobileNumberException e) {
            System.out.println(e.toString());
        }
    }
}
```

HTML- JS

Write a JavaScript program to create a class named 'SuperHero' with name, alias &
planet as members.
Create instances of above class like superman and batman.
Invoke the function WhoAmI which will be added to above class's prototype to display the name & other properties of
the instance.

Problem Constraints:
Class name should be capitalized '**SuperHero**'.
SuperHero should have the members like name, alias and planet.
Use – function SuperHero(name, alias, planet) { ..... }
It also has the prototype method **WhoAmI( )** to display all the properties.
Use new keyword to create instances like superman and batman .
Invoke WhoAmI( ) method to display all the properties.

Input and Output Format:

Refer sample input and output for formatting specifications.

Input corresponds to name, alias and planet separated by a comma.
First line of input corresponds to Superman instance.
Second line of input corresponds to BatMan instance.
Output should display all the properties as mentioned below.

Note: Input should be read from input.txt

Input 1:

Superman,Clark Kent,Krypton
Batman,Bruce Wayne,Earth

Output 1:

Superman Clark Kent is from the planet Krypton
Batman Bruce Wayne is from the planet Earth

```
var fs = require('fs');
var input=fs.readFileSync('input.txt').toString().trim().split('\n');

var input1 = input[0].trim().split(',');
var input2 = input[1].trim().split(',');

//console.log(input1);
//console.log(input2);

//Fill your code here

function SuperHero(name, alias, planet){
    this.name = name;
    this.alias = alias;
    this.planet = planet;
}

SuperHero.prototype.WhoAmI = function(){
    console.log(this.name + " " + this.alias + " is from the planet " + this.planet);
}

//console.log(input[0], input[1], input[2]);

var Superman = new SuperHero(input1[0], input1[1], input1[2]);
var BatMan = new SuperHero(input2[0], input2[1], input2[2]);

Superman.WhoAmI();
BatMan.WhoAmI();
```

Check Name Length

Check the given user name length using Javascript class.

Problem Description:
1) Read input file 'input.txt'
2) Create class named as User.
3) Inside the class create the argument as user name.
4) Now create getter for name.
5) Create the function checkNameLength() inside the class.
6) Inside the function check the length of the user name.
7) If the user name length is greater than 4 then, print the name as it is. Else print the message 'Name is too short'.

6) Create object for the class using new Classname() and pass the input to the constructor.
7) And call the object method, using object.methodName() to print the output.

Input and Output Format:

Refer to sample input and output.

Input is user name.

Output is user name.(If the length is greater than 4), Else 'Name is too short'.

Input:

Harina

Output:

Harina

Input:

Jim

Output:

Name is too short

```
var fs = require('fs');
var input=fs.readFileSync('input.txt').toString().trim().split('\n');
//fill your code

class User {
    constructor(name){
        this.name = name;
    }

    getName(){
        return this.name;
    }

}

function checkNameLength(person){

        let name = person.getName();

        if (name.length > 4)
            console.log(name);
        else
            console.log("Name is too short");
    }

var person = new User(input[0]);
checkNameLength(person);
```

Inheritance_Shapes

Write a program using inheritance.

Problem Description:
1) Read input file 'input.txt'

2) Create class named as Shape.
3) Inside the class create the argument as name,nos(number of side) and color.
4) Inside the constructor assign the name to this.name, nos to this.nos, color to this.color
5) Create displayShapeColor() method inside the class Shape.
6) Create another three classs named as Circle, Square, Triangle which inherits the class Shape.
7) Inside the child classes create method named as displayShapeSides().
8) Create objects for child classes using the inputs given, and access both parent and child class methods.

Input and Output Format:

Refer to sample input and output.

Input consists of three lines. Each line has a comma-separated value of name,nos,color

Output is shape with color/ shape with sides (Ex: Circle is in color red/ Square has 4 sides)

Input :

Circle,0,red
Square,4,red
Triangle,3,red

Output :

Circle is in color red // invokes parent class method
Circle has 0 sides //invokes child class method
Square is in color red // invokes parent class method
Square has 4 sides //invokes child class method
Triangle is in color red // invokes parent class method
Triangle has 3 sides //invokes child class method

```
var fs = require('fs');
var input=fs.readFileSync('input.txt').toString().trim().split('\n');

var input1 = input[0].trim().split(',');
var input2 = input[1].trim().split(',');
var input3 = input[2].trim().split(',');
//fill your code

class Shape{
    constructor(name, nos, color){
        this.name = name;
        this.nos = nos;
        this.color = color;
    }

    displayShapeColor(){
        console.log(this.name + " is in color " + this.color);
    }
}

class Circle extends Shape{
    displayShapeSides(){
        console.log("Circle has " + this.nos + " sides");
    }
}
```

```
class Square extends Shape{
    displayShapeSides(){
        console.log("Square has " + this.nos + " sides");
    }
}

class Triangle extends Shape{
    displayShapeSides(){
        console.log("Triangle has " + this.nos + " sides");
    }
}

var circle = new Circle(input1[0], input1[1], input1[2]);
var square = new Square(input2[0], input2[1], input2[2]);
var triangle = new Triangle(input3[0], input3[1], input3[2]);

circle.displayShapeColor();
circle.displayShapeSides();

square.displayShapeColor();
square.displayShapeSides();

triangle.displayShapeColor();
triangle.displayShapeSides();
```

Write a JavaScript console program to create two objects like Person and Employee respectively. Employee should inherit the Person's prototype. Display all the properties of Person and Employee using Employee instance.

Problem Constraints:
Create an object '**Person**' with property name.
Create an another object **Employee** with property employeeId.
Employee should be inherited from Person.
Using the instance of **Employee**, display all the properties.

Refer the below hint to define inherited property.
Hint:
let obj = {
…
…
}
let inheritedObj={
…
…
}
inheritedObj.__proto__=obj

Input and Output Format:

Refer sample input and output for formatting specifications.

Input corresponds to name and employeeId separated by a comma.
Output should display all the properties as mentioned below.

Note: Input should be read from input.txt

Input 1:

Shera,43

Output 1:

Using inherited instance
Name : Shera
Employee Id : 43

```
var fs = require('fs');
var name = fs.readFileSync('input.txt').toString();
let str =name.trim().split(",");

//Fill your code here

let Person = {
    name : str[0]
}

let Employee = {
    employeeId : str[1]
}

Employee.__proto__=Person;

console.log("Using inherited instance");
console.log("Name : " + Employee.name)
console.log("Employee Id : " + Employee.employeeId);
```

Print Employee Details

Print employee details using javascript class.

Problem Description:
1) Read input file 'input.txt'
2) Create a class named as Employee.
3) Inside the class create arguments employee name,dept and DOJ.
4) Now create the method as displayEmployee() and inside the method print the employee details.
5) Create object for the class using new Classname() and pass the input to the constructor.
6) And call the object method, using object.methodName() to print the employee details.

Input and Output Format:

Refer to sample input and output.

Input and output is employee details.

Input:

John
HR
23-8-1997

Output:

Name : John
Department : HR
DOJ : 23-8-1997

```
var fs = require('fs');
var input=fs.readFileSync('input.txt').toString().trim().split('\n');
//fill your code

class Employee{
    constructor(name, dept, DOJ){
        this.name = name;
        this.dept = dept;
        this.DOJ = DOJ;
    }

    displayEmployee(){
        console.log("Name : " + this.name);
        console.log("Department : " + this.dept);
        console.log("DOJ : " + this.DOJ);
    }
}

let obj = new Employee(input[0], input[1], input[2]);
obj.displayEmployee();
```

Welcome Customer

Print welcome customer message using Javascript class.

Problem Description:
1) Read input file 'input.txt'
2) Create class named as Customer.
3) Inside the class create the argument as customer name.
4) Inside the constructor assign the customer name to this.name.
5) Now create a method as sayWelcome() and inside the method print the welcome message.
6) Create object for the class using new Classname() and pass the input to the constructor.
7) And call the object method, using object.methodName() to print the welcome message.

Input and Output Format:

Refer to sample input and output.

Input is the customer name.

Output is the Welcome message.

Input:

John

Output:

Welcome John.

```
var fs = require('fs');
var input=fs.readFileSync('input.txt').toString().trim().split('\n');
//fill your code

class Customer{
```

```
    constructor(name){
        this.name = name;
    }

    sayWelcome(){
        console.log("Welcome " + this.name);
    }
}

var cust = new Customer(input[0]);
cust.sayWelcome();
```

Changing property using Javascript

Design an HTML page as shown in the screenshot and apply the style using Javascript function.

Property marginTop

Set the top margin of a element.

Syntax :

```
document.getElementById("myDiv").style.marginTop = "50px";
```

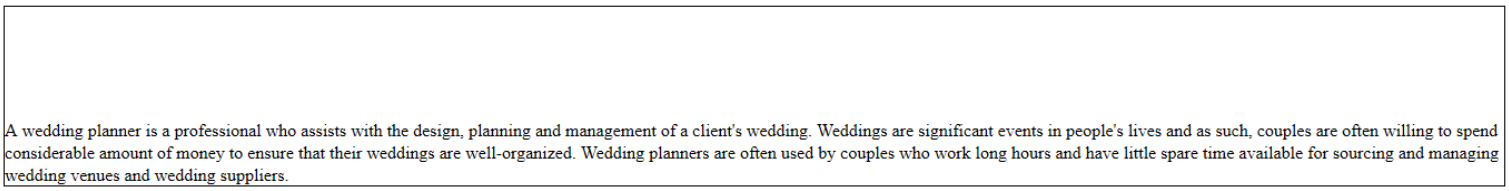
Property border

Add a border to a element.

Syntax :

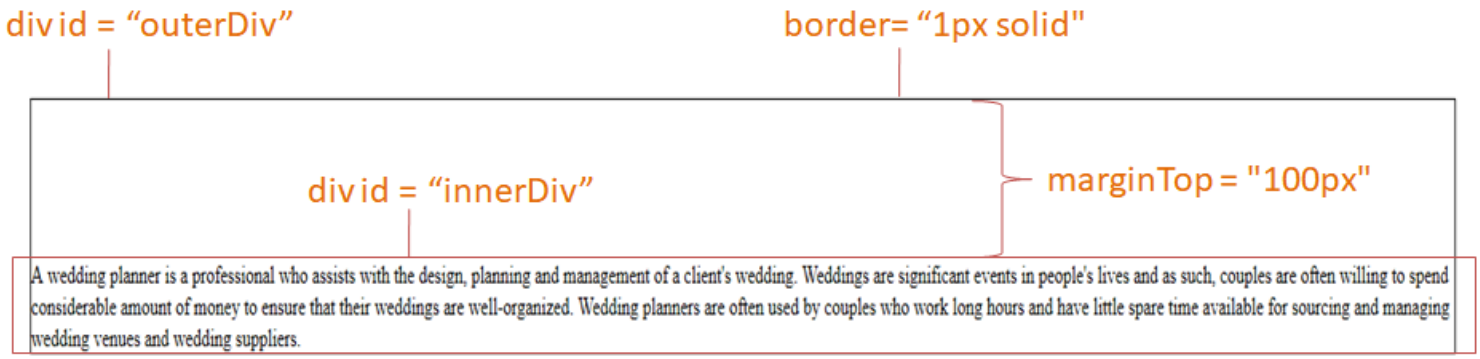
```
document.getElementById("myDiv").style.border= "1px solid";
```

Sample Screenshot :



Constraints:

The outer div should have a border and the inner div should have margin-top property



Additional Constraints :

The file name shall be index.html.
Have two div's, one inside the other.

The outer div shall have id "**outerDiv**" and apply **border** property in the script.
The inner div shall have id "**innerDiv**" and apply **marginTop** property in the script.
Change the properties inside the apply() function.
Load the function **apply()** on body **onload** attribute.

Note :

Content of the page should be present as shown in the screenshot.

Content :

A wedding planner is a professional who assists with the design, planning and management of a client's wedding. Weddings are significant events in people's lives and as such, couples are often willing to spend considerable amount of money to ensure that their weddings are well-organized. Wedding planners are often used by couples who work long hours and have little spare time available for sourcing and managing wedding venues and wedding suppliers.

Template Code :

Click here to download the template code.

```
<html>

<head>

    <script>
        function apply() {
            document.getElementById("innerDiv").style.marginTop = "100px";
            document.getElementById("outerDiv").style.border = "1px solid";
        }
    </script>

</head>

<body onload="apply()">

    <div id="outerDiv">
        <div id="innerDiv">

A wedding planner is a professional who assists with the design, planning and management of a clie
nt's wedding. Weddings are significant events in people's lives and as such, couples are often wil
ling to spend considerable amount of money to ensure that their weddings are well-
organized. Wedding planners are often used by couples who work long hours and have little spare ti
me available for sourcing and managing wedding venues and wedding suppliers
        </div>
    </div>




</body>

</html>
```

Diagonal Coloring

The DOM allows us to do anything with elements and their contents, but first we need to reach the corresponding DOM object. All operations on the DOM start with the document object. That's the main "entry point" to DOM. From it we can access any node. Elements that are direct children of a node are known as child nodes. Design a page to paint all the diagonal cells of the table with the given color.

```
Hint:
firstElementChild
The firstElementChild property returns the first child element of the
specified element
```

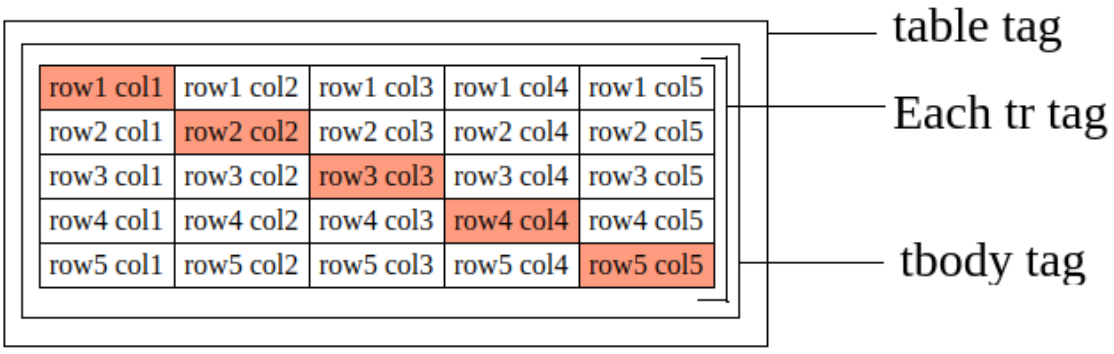
Constraints :

File name should be index.html.
Create a table with the content shown in the screenshot.
The color of the diagonal cells should be "FFA07A"
Table tag should be the first child of the document body.
In the script, table should be obtained using firstElementChild.
The background color should be changed using style.backgroundColor.
The row length should be 5.
Script tag should be inlined with index.html

Screenshot 1:

row1 col1	row1 col2	row1 col3	row1 col4	row1 col5
row2 col1	row2 col2	row2 col3	row2 col4	row2 col5
row3 col1	row3 col2	row3 col3	row3 col4	row3 col5
row4 col1	row4 col2	row4 col3	row4 col4	row4 col5
row5 col1	row5 col2	row5 col3	row5 col4	row5 col5

Screenshot 2:




```
<html>
<head>
    <style>
```

```
        td {
            border: 1px solid black;
        }
    </style>
    <script>
        function load() {
            var x = document.firstElementChild.innerHTML;
            document.body.style.backgroundColor = "";
        }
    </script>
</head>
<body onload="load()">
    <table>
        <tr>
            <td style="background-color:#ffa07a;">row1 col1</td>
            <td>row1 col2</td>
            <td>row1 col3</td>
            <td>row1 col4</td>
            <td>row1 col5</td>
        </tr>
        <tr>
            <td>row2 col1</td>
            <td style="background-color:#ffa07a;">row2 col2</td>
            <td>row2 col3</td>
            <td>row2 col4</td>
            <td>row2 col5</td>

        </tr>
        <tr>
            <td>row3 col1</td>
            <td>row3 col2</td>
            <td style="background-color:#ffa07a;">row3 col3</td>
            <td>row3 col4</td>
            <td>row3 col5</td>

        </tr>
        <tr>
            <td>row4 col1</td>
            <td>row4 col2</td>
            <td>row4 col3</td>
            <td style="background-color:#ffa07a;">row4 col4</td>
            <td>row4 col5</td>

        </tr>
        <tr>
            <td>row5 col1</td>
            <td>row5 col2</td>
            <td>row5 col3</td>
            <td>row5 col4</td>
            <td style="background-color:#ffa07a;">row5 col5</td>

        </tr>
    </table>

</body>
</html>
```

JS - Welcome page

Vaibhav has registered for "Kalostav", the event for exhibiting young talents in the field of Web designing. The sponsors of the event, Rofl Technologies has announced a cash reward of about of Rs.10,000 for the winners. In the first level of the contest, the participants were asked to create an artistic front page for Rofl Technologies which displays a welcome message for its users.

Vaibhav decided to design his page which reads the name of the user as input and displays a welcome message along with the username on the same page. Help him to win over the contest by creating a webpage that displays a welcome message.

Hints :DOM document.getElementById() Method The getElementById() method returns the element that has the ID attribute with the specified value. It is used almost whenever you want to manipulate, or get informations from, an element on your document. Returns null if no elements with the specified ID exists.Syntax : document.getElementById(elementID) **value property** The value property sets or returns the value of the value attribute of a text field. The value property contains the default value or the custom input (or a value set by a script).Syntax :textObject.value = text**innerHTML property** The innerHTML property sets or returns the HTML content (inner HTML) of an element. Syntax : HTMLElementObject.innerHTML = text

Constraints :

File name should be index.html.
The first input field must have the attribute id as "name".
Create the input field of type 'button' must have the attribute id as "post".
The message should be displayed inside the div element with id "result".
The function displaying the welcome message should be named as "welcome()".

Note :

Content of the page should be present as shown in the screenshot.

Sample Screenshot 1 :

Sample Screenshot 2 :


```html
<!DOCTYPE html>
<head>
    <script>
        function welcome() {
            const x = document.getElementById("name").value;
            document.getElementById("result").innerHTML = `<h2> Welcome ${x}</h2>`;


        }
    </script>


</head>

<body>
    <h1>Welcome Page</h1>
    <p>Name
        <input type="text" id="name" name="Name"></p>
    <input type="button" id="post" value="Post" onclick="welcome()"><br>
    <div id="result">

    </div>


</body>

</html>
```

JS - Grade

Maple Educational Institutions has conducted an Entrance Examination to select prospective students for admissions of the upcoming academic year. The students who have cleared the entrance exam with 50% and above i.e more than 'E' grade are considered for the post-admission process, where other students have to re-appear for the examinations.

The Institution Management decided to announce the exam results with the grade obtained by each of the students on their official website and has approached you for help. Create a webpage which will get the marks as input from the students and display the grade obtained by them.

Hints : if/else statement Use if to specify a block of code to be executed, if a specified condition is true.
Use else to specify a block of code to be executed, if the same condition is false.
Syntax :

```
if (condition) {
// if the condition is true block of code to be executed
} else {
// if fails else part gets executed
}
```

Constraints :
The Html page must contain a h1 tag.
The name and id of the text field should be "mark".

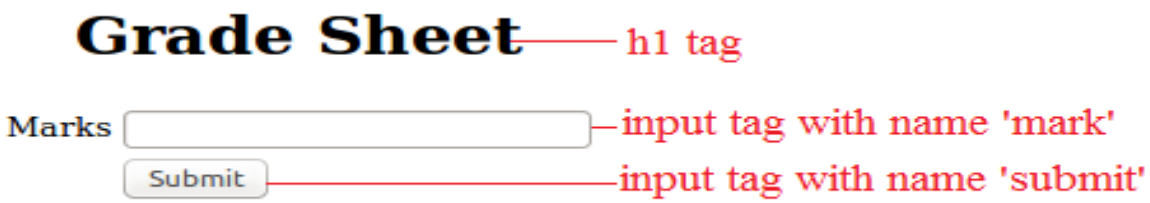
The name of the button should be "submit".
onclick attribute should have the value "myFunction()".
Grade sheet is as mentioned below.

S.No	Marks	Grade
1	100	A+
2	90 - 99	A
3	80 - 89	B
4	70 - 79	C
5	60 - 69	D
6	50 - 59	E
7	below 50	RA

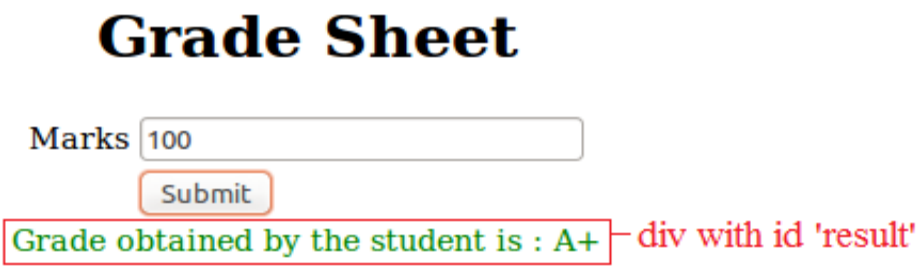
The grade of the student must be displayed inside the div element with id "result".

Note :
Content of the page should be present as shown in the screenshot.
Kindly refer the content which is given as a part of description.

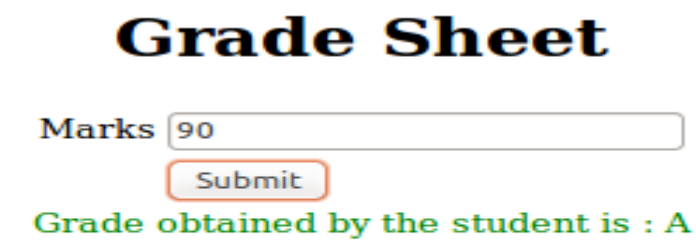
Sample Screenshot 1 :



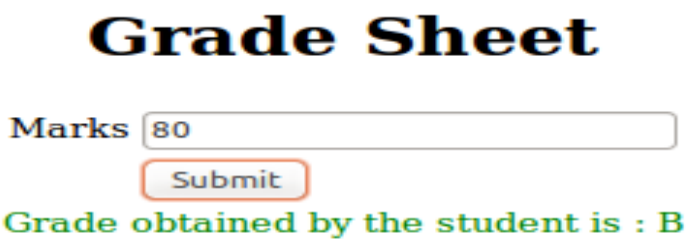
Sample Screenshot 2 :



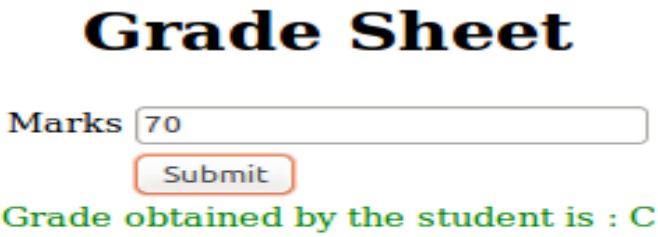
Sample Screenshot 3 :



Sample Screenshot 4 :

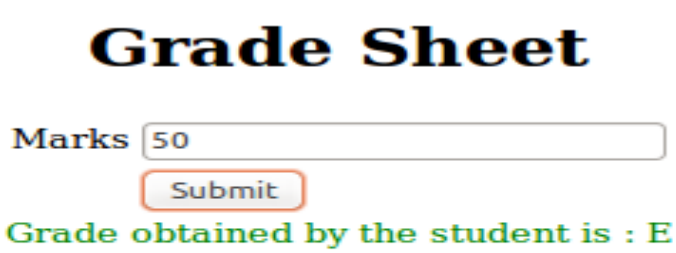


Sample Screenshot 5 :

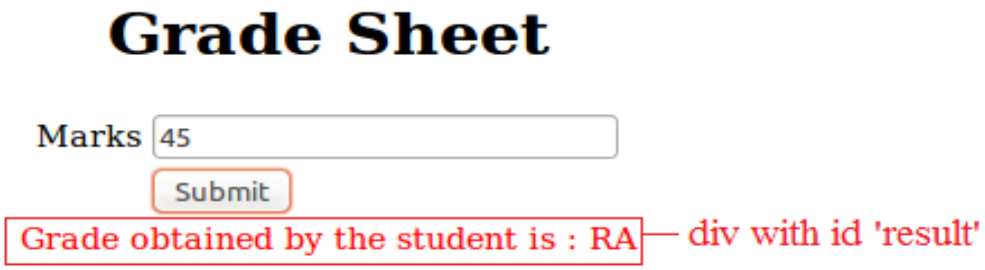


Sample Screenshot 6 :

Sample Screenshot 7 :



Sample Screenshot 8 :




```
<html>
<head>
    <title>MyPage</title>
    <script>
        function myFunction(){
            var x = document.getElementById("mark").value;
            if(x==100){
                var res = document.getElementById("result");
                res.innerHTML = 'Grade obtained by the student is : A+';
                res.style.color = "green";
            }
            else if(x>=90 && x<=99) {
                var res = document.getElementById("result");
                res.innerHTML = 'Grade obtained by the student is : A';
                res.style.color = "green";
            }
            else if(x>=80 && x<=89) {
                var res = document.getElementById("result");
                res.innerHTML = 'Grade obtained by the student is : B';
                res.style.color = "green";
            }
            else if(x>=70 && x<=79) {
                var res = document.getElementById("result");
                res.innerHTML = 'Grade obtained by the student is : C';
                res.style.color = "green";
            }
            else if(x>=60 && x<=69) {
```

```
                var res = document.getElementById("result");
                res.innerHTML = 'Grade obtained by the student is : D';
                res.style.color = "green";
            }
            else if(x>=50 && x<=59) {
                var res = document.getElementById("result");
                res.innerHTML = 'Grade obtained by the student is : E';
                res.style.color = "green";
            }
            else if(x<50){
                var res = document.getElementById("result");
                res.innerHTML = 'Grade obtained by the student is : RA';
                res.style.color = "red";
            }
        }
    </script
</head>
<body>
<!--  fill your code here -->
<h1>Grade Sheet</h1>
<p>Marks
    <input type="number" name="mark" id="mark">
    </p>
    <input type="button" name="submit" value="Submit" onclick="myFunction()">
    <div id="result">
        </div>
</body>
</html>
```

DOM modification

DOM modification is the key to creating “live” pages. New elements can be created and inserted at the adjacent position. A node can be cloned or removed from the DOM. Design a web page to show the mothly income rate.

```
Hint:
document.createElement(tag) creates a new DOM element with give tag.
document.createTextNode(text) creates a new text node with the given text.
elem.insertAdjacentHTML(where, html) creates new adjacent html element to the curent element
node.remove() removes the node from the DOM
element.cloneNode(boolean) clones the node
```

Constraints:

File name should be index.html.
Initally, the income rate for first three months of a year will be given in a table.
Create a button with id ‘button’ and onclicking the button call a function getMonths().
Use insertAdjacentHTML() with beforebegin to create a h2 tag with content "Rate table of monthly income".
Insert rows and cells to the existing table.
Use createTextNode() and appendChild() methods to inser values to the table cells.
Once the entire table is loaded, button should not be displayed.

Note:

Kindly use the var keyword instead of let/const for variable declaration.
Content of the page should be present as shown in the screenshot.
The monthly rate of income is:
January - 5.5

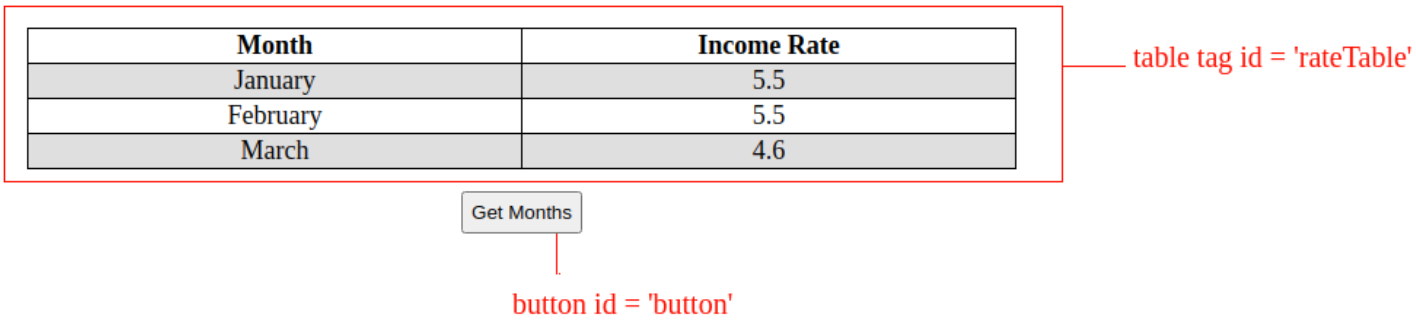
February - 5.5
March - 4.6
April - 8
May - 6
June - 7.2
July - 4.6
August - 6.9
September - 8.5
October - 9.4
November - 5.1
December - 7.5

Screenshot 1:

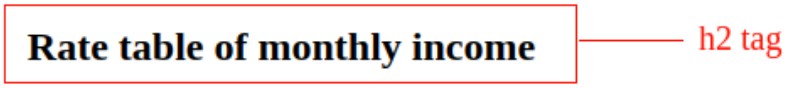
Month	Income Rate
January	5.5
February	5.5
March	4.6

Get Months

Screenshot 2:



Screenshot 3:



Month	Income Rate
January	5.5
February	5.5
March	4.6
April	8
May	6
June	7.2
July	4.6
August	6.9
September	8.5
October	9.4
November	5.1
December	7.5

```
<!DOCTYPE html>
<html>
<style>
    body {
        text-align: center;
    }

    table,th,td {
        width: 50%;
        font-size: 18px;
        border: 1px solid black;
        border-collapse: collapse;
    }

    tbody tr:nth-child(odd) {
        background-color: #e0dfdf;
    }

    table.center {
        margin-left: auto;
        margin-right: auto;
    }

    button {
        padding: 5px;
        margin: 15px;
        cursor: pointer;
    }
</style>

<body>
    <table class="center" id="rateTable">
```

```
    <thead>
      <tr>
        <th>Month</th>
        <th>Income Rate</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>January</td>
        <td>5.5</td>
      </tr>
      <tr>
        <td>February</td>
        <td>5.5</td>
      </tr>
      <tr>
        <td>March</td>
        <td>4.6</td>
      </tr>
    </tbody>
  </table>
  <button id="button" onclick="getMonths()">Get Months</button>
  <script>
      function getMonths(){
      document.getElementById("button").style.display = "none";

      var table = document.getElementById("rateTable");
      var tbody = table.lastElementChild;

      table.insertAdjacentHTML("beforebegin", "<h2>Rate table of monthly income</h2>");

      var month = ["April", "May", "June", "July", "August", "September", "October", "November",
"December"];
      var values = ["8", "6", "7.2", "4.6", "6.9", "8.5", "9.4", "5.1", "7.5"];

      function tableExtension(month,val){
        var tr = document.createElement("tr");
        var td = document.createElement("td");
        var mon = document.createTextNode(month);
        var monval = document.createTextNode(val);

        td.appendChild(mon);
        tr.appendChild(td);

        var td = td.cloneNode(false);

        td.appendChild(monval);
        tr.appendChild(td);

        tbody.appendChild(tr);
      }

      for(var i in month)
        tableExtension(month[i],values[i]);
  }
  </script>
```

```
  </body>

</html>
```

JS - Add / Remove class

The creative team of "Pink Frag Event Organizers" has planned to create an innovative UI for the new blog to be uploaded in their official website. The main framework is to display the interesting facts about the events organized by the firm in order grab the attention of millions of people through internet.

Being one among the team, you are assigned the task of creating the blog which will display the effects over the elements on clicking the 'add' button and remove the effects on clicking the 'remove' button and should toggle the current effect on clicking 'toggle' button using javascript.

Toggle - If the element has effect, it removes the effect. If doesn't have effect, it adds the effect.

Hints : classList property The classList property returns the class name(s) of an element, as a DOMTokenList object. This property is useful to add, remove and toggle CSS classes on an element. The classList property is read-only, however, you can modify it by using the add() and remove() methods. Syntax :

```
element = document.getElementById("idName");
element.classList.add("classProperty");
```

Similarly, remove() and toggle() methods were used.

Constraints :

Design the html page as given in the sample screenshot1.
Tag 'h1' must be present in the html page.
Button tag must be present in the html page.
Tag 'div' must be present in the html page as given in the screenshot.
The onClick functions should be specified same as given in the screenshot.
Use add, remove and toggle methods.

Initially all div should be present with the below style
- border - 1px solid black
- width - 200px
- height - 100px
- background-color - none
- text-align - left

The class 'class1' should be present with the below style
- font-size - 30px
- background color - #20c5b5
- color - white
- text-align - center

The class 'class2' should be present with the below style
font-size - 30px
background-color - #FA8072
color - white
text-align - center

The class 'class3' should be present with the below style
font-size - 30px
background color - #7d8c9b
color - white
text-align - center

The class 'class4' should be present with the below style
font-size - 30px
background color - #7ba428
color - white
text-align - center

Conditions :

When clicking the Add button, apply effects for all the div's.
When clicking the Remove button, remove effects for all the div's.
When clicking the Toggle button, remove the applied effect if it has effects and vice versa.

Note :

Content of the page should be present as shown in the screenshot.
Kindly refer the content which is given as a part of description.

Sample Screenshot 1 :

Sample Screenshot 2 :

Sample Screenshot 3 :
Use width attribute for all the div is '200px'

Sample Screenshot 4 :

Sample Screenshot 5 :

Pink Frag Event Organization


```
<!DOCTYPE html>
<html>
<head>
    <style>
        #div1 {
            border: 1px solid black;
            width: 200px;
            height: 100px;
            background-color: none;
            text-align: center;
        }

        #div2 {
            border: 1px solid black;
            width: 200px;
            height: 100px;
            background-color: none;
            text-align: center;
        }

        #div3 {
            border: 1px solid black;
            width: 200px;
            height: 100px;
            background-color: none;
            text-align: left;
        }

        #div4 {
            border: 1px solid black;
            width: 200px;
            height: 100px;
            background-color: none;
            text-align: left;
        }

        .class1 {
            font-size: 30px;
            background-color: #20c5b5;
            color: white;
            text-align: center;
```

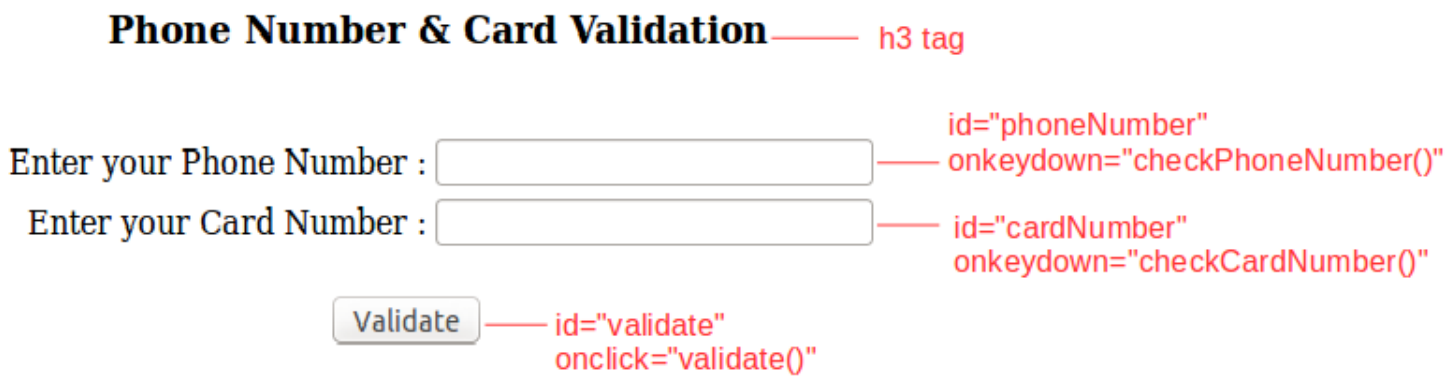
```
        }

        .class2 {
            font-size: 30px;
            background-color: #fa8072;
            color: white;
            text-align: center;
        }

        .class3 {
            font-size: 30px;
            background-color: #7d8c9b;
            color: white;
            text-align: center;
        }

        .class4 {
            font-size: 30px;
            background-color: #7ba428;
            color: white;
            text-align: center;
        }
    </style>
</head>

<body>
    <h1>Pink Frag Event Organization</h1>
    <button id="add" onclick="addClass()">Add</button>
    <button id="remove" onclick="removeClass()">Remove</button>
    <button id="toggle" onclick="toggleClass()">Toggle</button>
    <div id="div1" >Wedding Event </div>
    <div id="div2" >Conferences</div>
    <div id="div3" >Birthday Party</div>
    <div id="div4" >Trade Fairs</div>

</body>
<script>
    function addClass() {
        var a = document.getElementById("div1");
        var b = document.getElementById("div2");
        var c = document.getElementById("div3");
        var d = document.getElementById("div4");
        a.classList.add("class1");
        b.classList.add("class2");
        c.classList.add("class3");
        d.classList.add("class4");
    }

    function removeClass() {
        var a = document.getElementById("div1");
        var b = document.getElementById("div2");
        var c = document.getElementById("div3");
        var d = document.getElementById("div4");
        a.classList.remove("class1");
        b.classList.remove("class2");
        c.classList.remove("class3");
```

```
        d.classList.remove("class4");

    }

    function toggleClass() {
        var a = document.getElementById("div1");
        var b = document.getElementById("div2");
        var c = document.getElementById("div3");
        var d = document.getElementById("div4");
        a.classList.toggle("class1");
        b.classList.toggle("class2");
        c.classList.toggle("class3");
        d.classList.toggle("class4");

    }
</script>
</html>
```

JS - Credit card validation

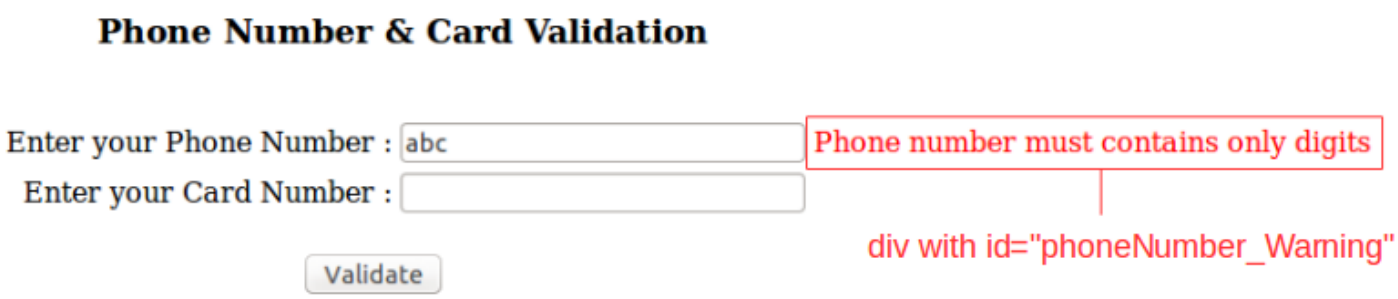
Design an HTML page to perform phone number and credit card validation using javascript as shown in the screenshot.

Sample Screenshot 1:



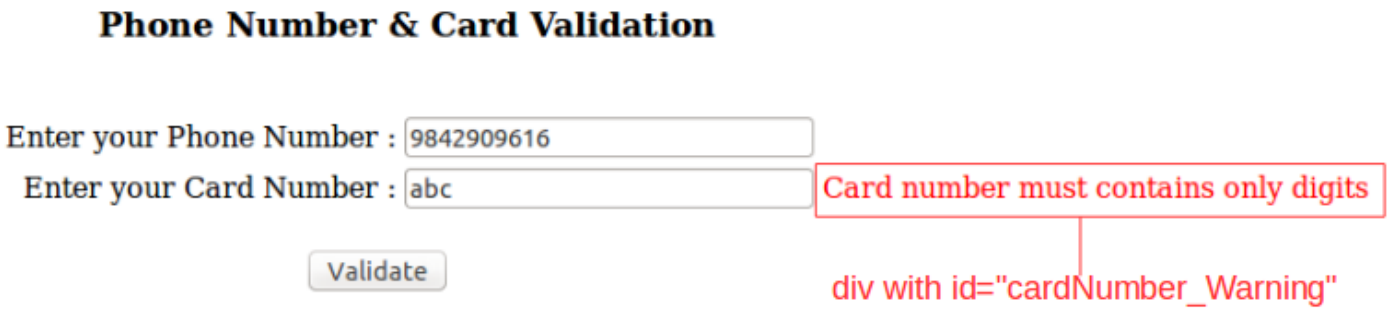
Sample Screenshot 2 :

If the phone number contains anything other than digits it must print 'Phone number must contain only digits' in the div with id 'phoneNumber_Warning'.

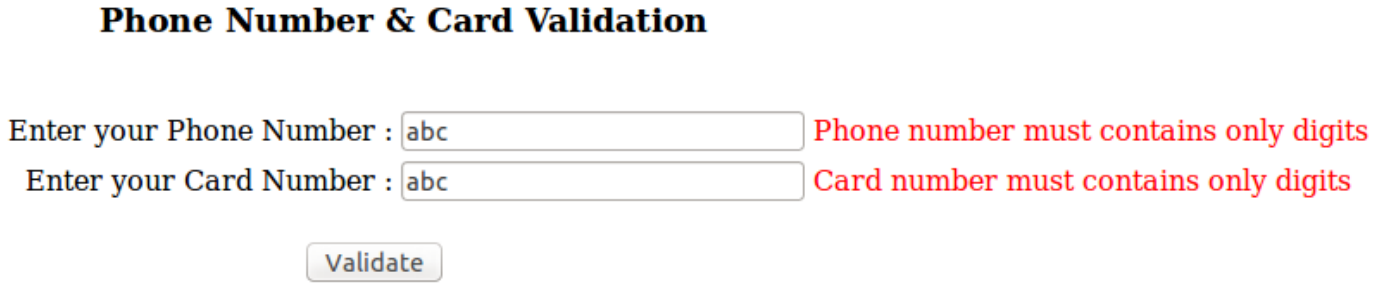


Sample Screenshot 3 :

If the card number contains other than digits it must print 'Card number must contain only digits' in the div with id 'cardNumber_Warning'.

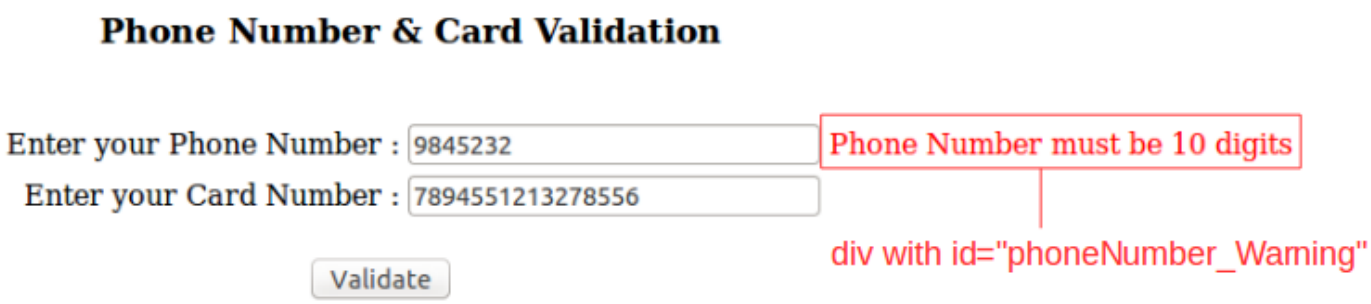


Sample Screenshot 4 :



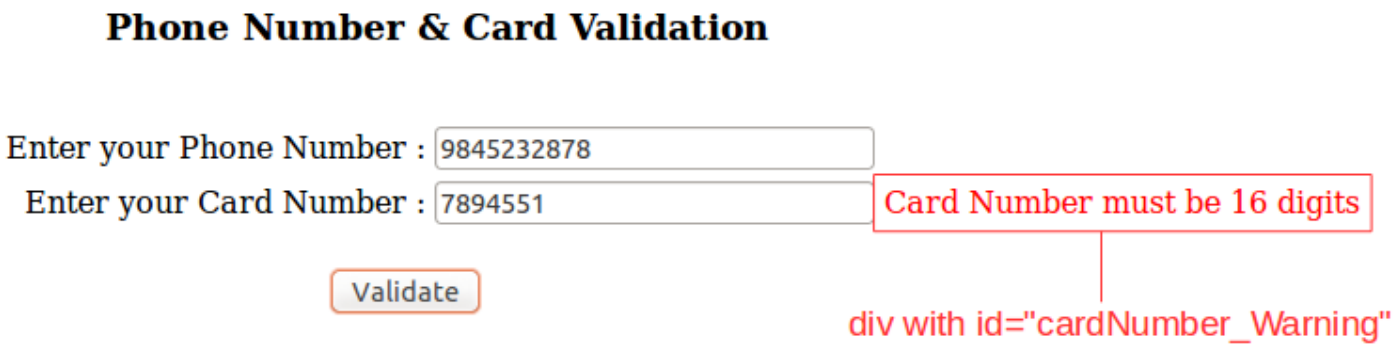
Sample Screenshot 5 :

If the Phone number does not contain 10 digits it must print 'Phone Number must be 10 digits' in the div with id 'phoneNumber_Warning'.



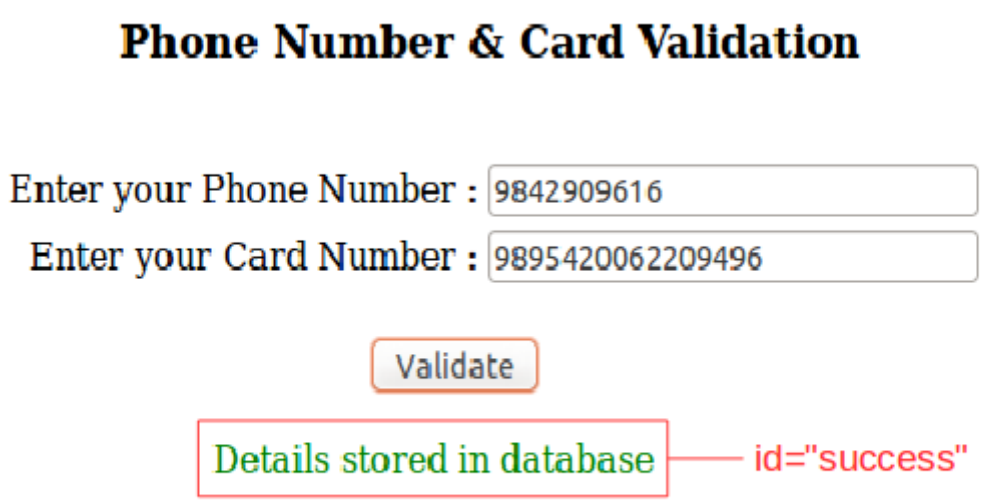
Sample Screenshot 6 :

If the card number does not contain 16 digits it must print 'Card Number must be 16 digits' in the div with id 'cardNumber_Warning'.



Sample Screenshot 7 :

If both phone and card numbers are validated, then print 'Details stored in database' in the div with id 'success'



Note :

Content of the page should be present as shown in the screenshot.

```
<html>
    <head>
        <script type="text/javascript">
            function validate()
            {
               phone = document.getElementById("phoneNumber").value
               card = document.getElementById("cardNumber").value
               phonewarning = document.getElementById("phoneNumber_Warning")
               cardwarning = document.getElementById("cardNumber_Warning")

               if(phone.length!==10){
                   phonewarning.innerHTML = "Phone Number must be 10 digits"
                   phonewarning.style.display="inline";
                   document.getElementById("success").style.display = "none"

               }
               if(card.length!==16){
                   cardwarning.innerHTML = "Card Number must be 16 digits"
                   cardwarning.style.display="inline";
                   document.getElementById("success").style.display = "none"

               }

               if(phone.length===10 && card.length===16){
                   phonewarning.innerHTML = ""
                   phonewarning.style.display="none"
                   cardwarning.innerHTML = ""
                   cardwarning.style.display="none";
                   document.getElementById("success").style.display = "block"
               }
            }
            function checkPhoneNumber()
            {
```

```
                value = document.getElementById("phoneNumber").value
                warning = document.getElementById("phoneNumber_Warning")


                for(var i=0;i<value.length;i++){
                    if(value.charCodeAt(i)<48 || value.charCodeAt(i)>57)
                    {
                        warning.innerHTML = "Phone number must contains only digits"
                        warning.style.display="inline";
                        return;
                    }
                }
                warning.innerHTML = ""
                warning.style.display="none";

            }
            function checkCardNumber()
            {

                value = document.getElementById("cardNumber").value
                warning = document.getElementById("cardNumber_Warning")

                for(var i=0;i<value.length;i++){
                    if(value.charCodeAt(i)<48 || value.charCodeAt(i)>57)
                    {
                        warning.innerHTML = "Card number must contains only digits"
                        warning.style.display="inline";
                        return;
                    }
                }
                warning.innerHTML = ""
                warning.style.display="none";

            }
        </script>
    </head>
    <body>
        <h3>Phone Number & Card Validation</h3>
        Enter your Phone Number:<input type="text" onkeydown="checkPhoneNumber()" id="phoneNumber"
 oninput="checkPhoneNumber()">
        <div id="phoneNumber_Warning"></div></br>

        Enter your Card Number:<input type="text" onkeydown="checkCardNumber()" id="cardNumber" on
input="checkCardNumber()">
        <div id="cardNumber_Warning"></div></br>
        <button type="button" id="validate" onclick="validate()">Validate</button>
       <div id="success" style="display:none">Details stored in database</div>
    </body>
</html>
```

JS - Oninput Event

"Eventious" is one of the most popular online application being used by the people of all age groups as it would include knowledgeable games. They have planned to launch a new game that would display the content simultaneously when the user types the values in the specific field. This game would estimate the speed of the contents being typed in the content box.

As the developers of Eventious are busy in playing support roles for the existing games, the technnical team has assigned you the task of creating the web-app for repeating the contents in the text area using oninput events in javascript.

Hints : The oninput event occurs when an element gets user input.
This event occurs when the value of an <input> or <textarea> element is changed.Syntax :<element oninput="myScript">

Content :

h3 - Oninput Event
Have one element with id 'result'
Have a <textarea> field with id 'myContent' and oninput attribute value 'displayContent()'
Display the whole content inside the tag.
Include the following functions/methods in the script.

Function Name	Description
displayContent()	This method is used to display the content of textarea in the resulting element while the user types and clears the content.

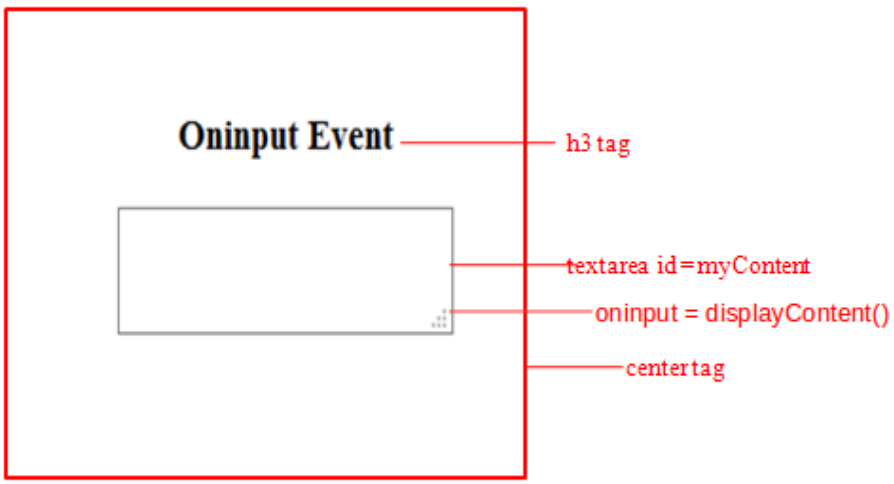
Constraints:

File name should be 'index.html'.
Enter the content in the textarea with id 'myContent'.
The content of textarea must be displayed inside the div element with id 'result' while the user types and clears the content.

Note :

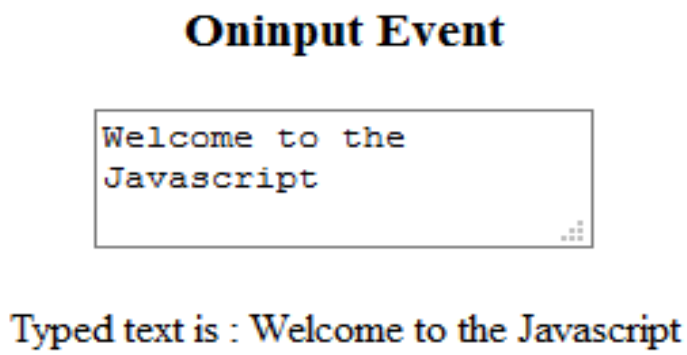
Content of the page should be present as shown in the screenshot.
Kindly refer the content which is given as a part of description.

Sample Screenshot 1 :



Sample Screenshot 2 :

On entering the contents in textarea , it should be displayed in the result div as mentioned below,



Sample Screenshot 3 :

Sample Screenshot 4 :
On clearing the content in textarea , the change should be reflected in the result div.


```
<!DOCTYPE html>
<html>
<body>
 <center>
   <h3>Oninput Event</h3>
 <textarea id="myContent" oninput="displayContent()"></textarea>

<div id="result"></div>
</center>
</body>
 <script>
 function displayContent(){
text = document.getElementById("myContent").value
if(text.length===0){
document.getElementById("result").innerHTML = "";
return
}
document.getElementById("result").innerHTML = "Typed text is : " + text;
}
 </script>
  </html>
```

Event Listeners

PGG Institutions has planned to organize the national level hackathon competition as a part of the Technical symposium. The institution decided to create their own editor console for the event using the hi-tech learners of their institution. This would encourage young talents in their college towards a new field of technology. Being a nationalized competition, there would be higher chances for plagiarism and its the responsibility of the developers to create a editor page which would restrict the functions such as cut, copy and paste.

The restrictions of these functions can be made by using javascript code, hence being one of them, help your team in creating the editor console which would avoid the usage of such functions which would lead to plagiarism.

Hints :

```
The oncut event occurs when the user cuts the content of an element. The
oncopy event occurs when the user copies the content of an element. The
onpaste event occurs when the user pastes some content in an element.
Syntax :
<element oncut="myScript">
<element oncopy="myScript">
<element onpaste="myScript">
```

Constraints:

File name should be index.html

Use oncut attribute for cut, oncopy attribute for copy and onpaste attribute for paste.

Refer the screenshots for the id to be used.

Use the event listeners to restrict cut,copy and paste in the webpage.

Function name	Description
warning()	This method is used to display the alert 'Cut/Copy/Paste is restricted.'

Note :

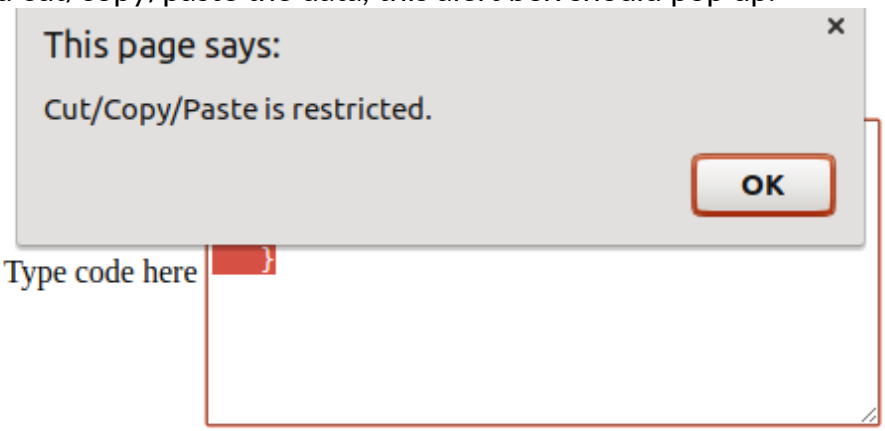
Content of the page should be present as shown in the screenshot.

Kindly refer the content which is given as a part of description.

Sample Screenshot 1 :

On entering the data in the textarea,

Sample Screenshot 2 :
When we try to select and cut/copy/paste the data, this alert box should pop up.




```
<!DOCTYPE html>
<html lang="en">
    <script>
        function warning() {
            alert('Cut/Copy/Paste is restricted.');
        }
    </script>
<body>
    <h2>Programming Contest</h2>
    <textarea id="editor" name="" rows="10" cols="30" oncut="warning()" oncopy="warning()" onpaste
="warning()"></textarea>
</body>
</html>
```

JS - Increase width and height

Nowadays, pictures play an important role in strengthening the communication and marketing efforts in all levels of publishing the organization. Recently Lee, the techno-assistant of "Pink Frag Event Organization" got a feedback regarding the images being uploaded in their official website couldnt be clearly understood,

as the resolution was not upto the level.

Hence, Lee has assigned you the task of making the images to be zoomed in i.e, the width and heigth of the images should be increased on every double click on the image using the javascript functions.

Hints : Width and height Property The width property sets or returns the width an element. The height property sets or returns the height of an element. The width and height property has effect only on block-level elements or on elements with absolute / fixed position. Syntax : object.style.width (for width) and object.style.height (for height) where object = document.getElementById("#id");

Content :

h1 - Increase width and height of div
div id="myDiv" width="400px" height="200px" ondblclick="addSize()"
h3 - Pink frag Event Organization
Our events team handles all your requirements from conceptualization to execution, leaving you free to focus completely on achieving your objectives.With proper planning and logistics management and the ability to activate requisite resources, we have undertaken a wide variety of projects.

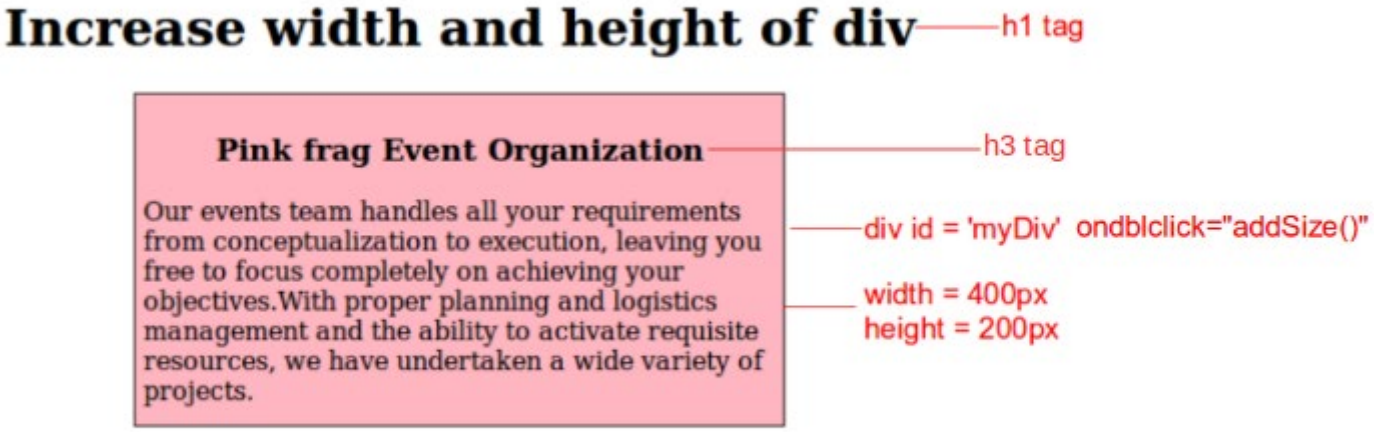
Constraints:

The file name should be index.html.
Create a div with id "myDiv" and the content as mentioned above.
Increase the width and the height of the div by "10px" on every double click.

Note :

Content of the page should be present as shown in the screenshot.
Kindly refer the content which is given as a part of description.

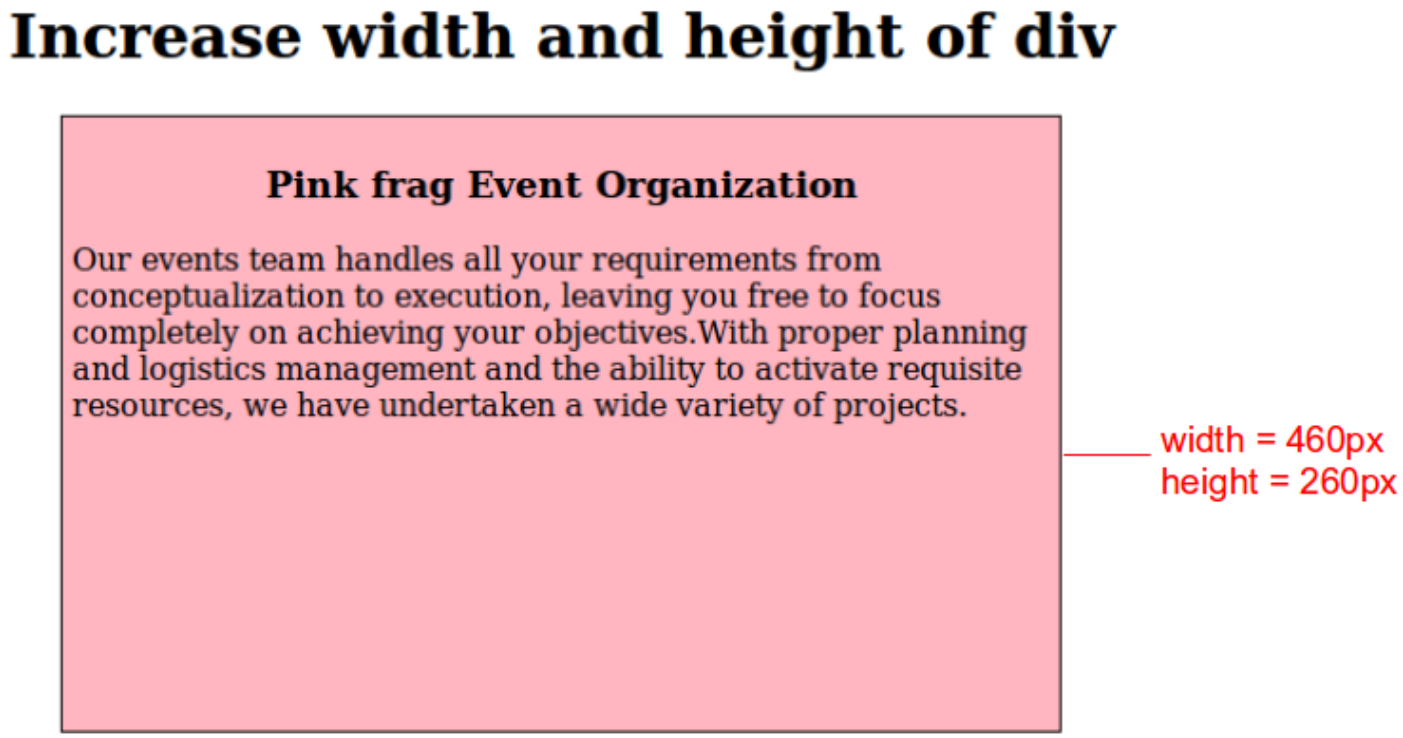
Sample Screenshot 1 :



Sample Screenshot 2 :
For the first doubleClick(),

Sample Screenshot 3 :

For the sixth doubleClick(),

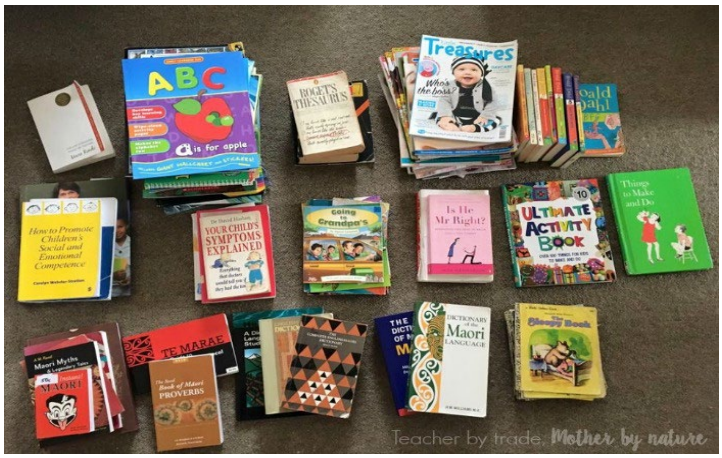



```
<!DOCTYPE html>
<html lang="en">

<head>
    <style>
        #myDiv {
            width: 400px;
            height: 200px;
            border: 1px solid black;
        }
    </style>
    <script>
        function addSize() {
            var widthElem = document.getElementById('myDiv').offsetWidth;
            var heightElem = document.getElementById('myDiv').offsetHeight;
            document.getElementById('myDiv').style.width = widthElem + 8 + 'px'
            document.getElementById('myDiv').style.height = heightElem + 8 + 'px'

        }
    </script>

</head>

<body>
    <h1>Increase width and height of div</h1>
    <div id="myDiv" ondblclick="addSize()">
        <h3>Pink frag Event Organization</h3>
        Our events team handles all your requirements from conceptuallization to execution,leaving
 you free to foocus completely on achieving your objectives. With proper planning and logistics ma
nagement and the ability to activate requisite resources, we have
        undertaken a wide variety of projects.
    </div>
```

```
</body>

</html>
```

ook Details using XML

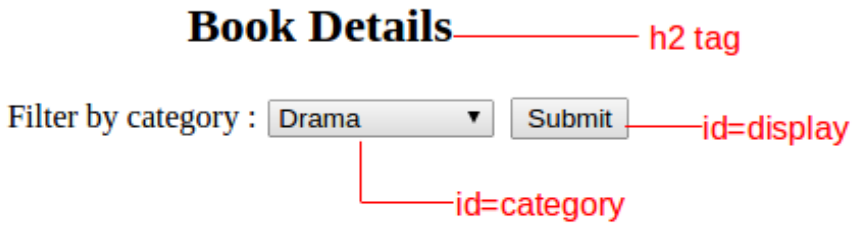
Gita used to read a lot of books. She has stored different books in her system. Now she wants to recollect the stored books by the category . She has confused while recollecting the books. Help her to recollect the books which are under same category using XML.



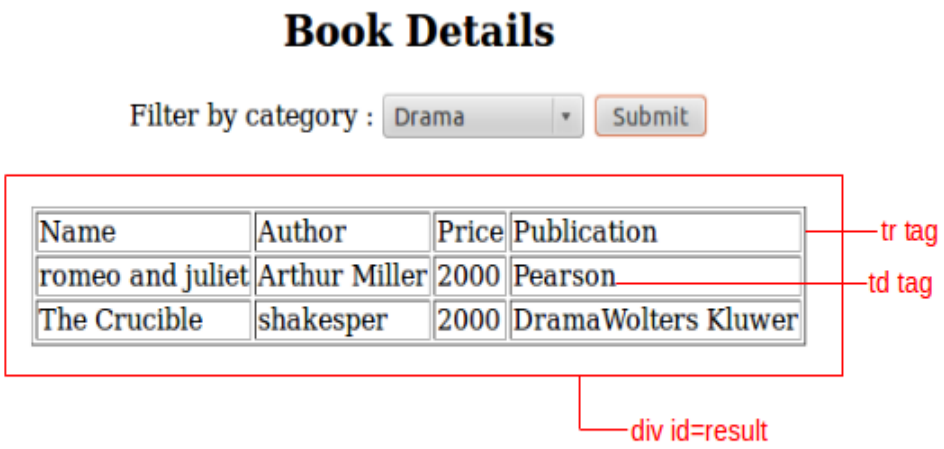
UI Constraints:

- The file name should be **index.html**
- For the xml file Click Here.
- Get the book details from the XML by selecting the category name
- By selecting the category display all the book details under that category

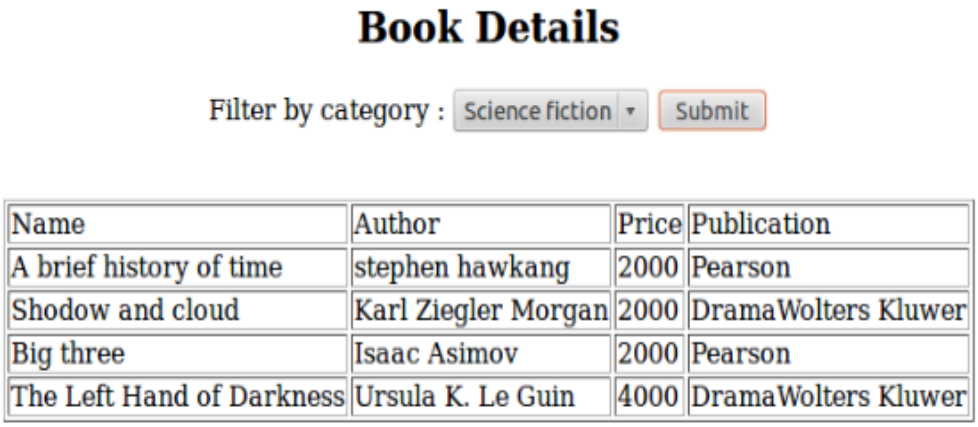
[Note:Strictly adhere to the specifications given in the screenshot. Use the same id names for all the corresponding fields specified.]
Sample Screenshot 1:



Sample Screenshot 2:



Sample Screenshot 3:




```
<!DOCTYPE html>
<html>
  <head>
    <style>
      td {
        border: 1px solid black;
        border-collapse: collapse;
      }
      table{
        border: 1px solid black;
      }
    </style>
    <script>
      function showTable(bookList) {
        var result = document.getElementById("result");
```

```
    result.innerHTML = "";

    var table = document.createElement("table");
    var tbody = document.createElement("tbody");

    table.appendChild(tbody);
    result.appendChild(table);

    tbody.innerHTML =
      "<tr><td>Name</td><td>Author</td><td>Price</td><td>Publication</td></tr>";

    for (var book of bookList) {
      var name = book.getElementsByTagName("name")[0].textContent;
      var author = book.getElementsByTagName("author")[0].textContent;
      var price = book.getElementsByTagName("price")[0].textContent;
      var publication =
        book.getElementsByTagName("publication")[0].textContent;

      tbody.innerHTML +=
        "<tr><td>" +
        name +
        "</td><td>" +
        author +
        "</td><td>" +
        price +
        "</td><td>" +
        publication +
        "</td></tr>";

    //   var tr = document.createElement("tr");
    //   tbody.appendChild(tr);
    }
  }
  function getXml() {
    var request = new XMLHttpRequest();
    request.open("GET", "bookDetails.xml");
    var category = document.getElementById("category").value;
    request.onreadystatechange = function () {
      if (request.readyState == 4 && request.status == 200) {
        var bookList = [];
        var doc = request.responseXML;
        var book = doc.getElementsByTagName("book");
        if (category == "Drama") {
          for (var i = 0; i < book.length; i++) {
            if (book[i].getAttribute("category") == category)
              bookList.push(book[i]);
          }
          showTable(bookList);
        } else if (category == "Science fiction") {
          for (var i = 0; i < book.length; i++) {
            if (book[i].getAttribute("category") == category)
              bookList.push(book[i]);
          }
          showTable(bookList);
        }
```

```
        }
      };
      request.send();
    }
    function handleEvent() {
      document.getElementById("display").onclick = getXml;
    }
    window.onload = handleEvent;
  </script>
 </head>
 <body>
  <h2>Book Details</h2>
  <label>Filter by category : </label>
  <select name="category" id="category">
    <option value="Drama">Drama</option>
    <option value="Science fiction">Science fiction</option>
  </select>
  <button id="display">Submit</button>
  <div id="result"></div>
 </body>
</html>
```

Supporting Documents

- playlist.xml

Ajax xml retrieval

Design a webpage which will get the xml details from a file and display it in a table as shown in screenshot using **AJAX.**

Problem specification:

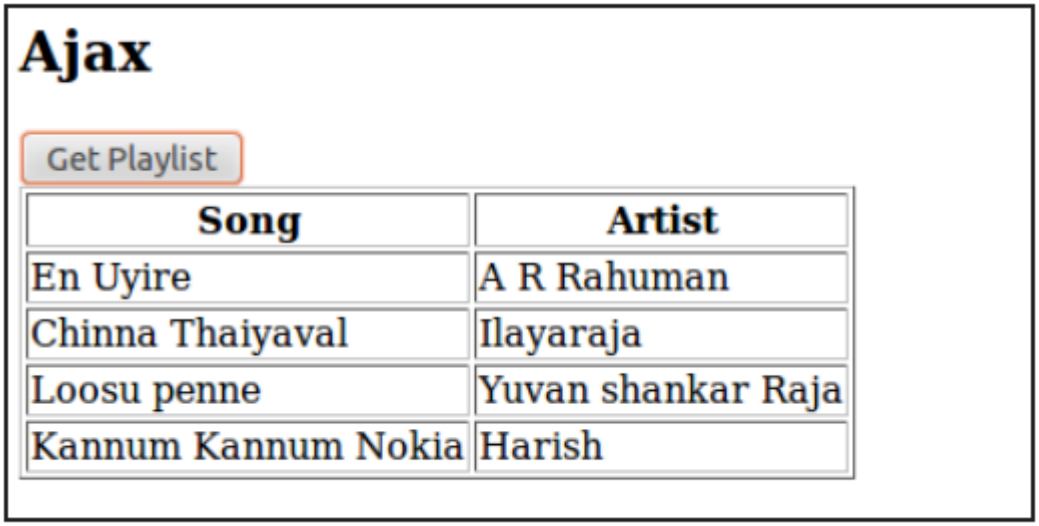
- The file name should be **index.html.**
- For the xml file Click Here.

UI Constraints :

- Have a heading text '**Ajax**' inside the '**h1**' tag.
- Have a button with id '**getPlaylist**' inside the html page.
- When the button is clicked, an ajax call is performed to retrive **playlist.xml**.
- The xml file name should be **playlist.xml.**
- Parse the xml elements and display it in tabular form as shown in screenshot.
- Have a div with id '**playlist**' . And display the result table inside the div with id '**playlist**'.
- While generating table , use th Tag to display the text "**Song**" and "**Artist**"

Sample Screenshot 1:

Sample Screenshot 2:




```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <title>Document</title>
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style>
      th,td {
        border: 1px solid black;
        border-collapse: collapse;
      }
      table {
        border: 1px solid black;
      }
    </style>
    <script>
      function showTable(playlist) {
        var result = document.getElementById("playlist");

        result.innerHTML = "";

        var table = document.createElement("table");
        var tbody = document.createElement("tbody");

        table.appendChild(tbody);
        result.appendChild(table);

        tbody.innerHTML =
          "<tr><th>Song</th><th>Artist</th></tr>";

        for (var song of playlist) {
          var name = song.getElementsByTagName("name")[0].textContent;
          var artist = song.getElementsByTagName("artist")[0].textContent;
          var description =
            "<td>" +
            name +
            "</td><td>" +
            artist +
```

```
            "</td>";

          var tr = document.createElement("tr");

          tr.innerHTML = description;
          tbody.appendChild(tr);
        }
      }
      function getXml() {
        var request = new XMLHttpRequest();
        request.open("GET", "playlist.xml", true);
        request.onreadystatechange = function () {
          if (request.readyState == 4 && request.status == 200) {
            var playlist = [];
            var doc = request.responseXML;
            var song = doc.getElementsByTagName("song");
            for (var i = 0; i < song.length; i++) {
              playlist.push(song[i]);
            }
            showTable(playlist);
          }
        };
        request.send();
      }
      function handleEvent() {
        document.getElementById("getPlaylist").addEventListener("click",getXml);
      }
      window.onload = handleEvent;
    </script>
  </head>
  <body>
    <h1>Ajax</h1>
    <button id="getPlaylist">Get Playlist</button>
    <div id="playlist"></div>
  </body>
</html>
```

Ajax – XML Retrieval and Json Conversion

Design a webpage which will get the xml details from a file and display it in a table. Then, display the Input fields to convert the particular booklist into a json format.

Problem specification:

- The file name should be **index.html.**
- Get the details from the XML data and display it in the table.
- Then displays the table data in the object notation format.

UI Constraints :

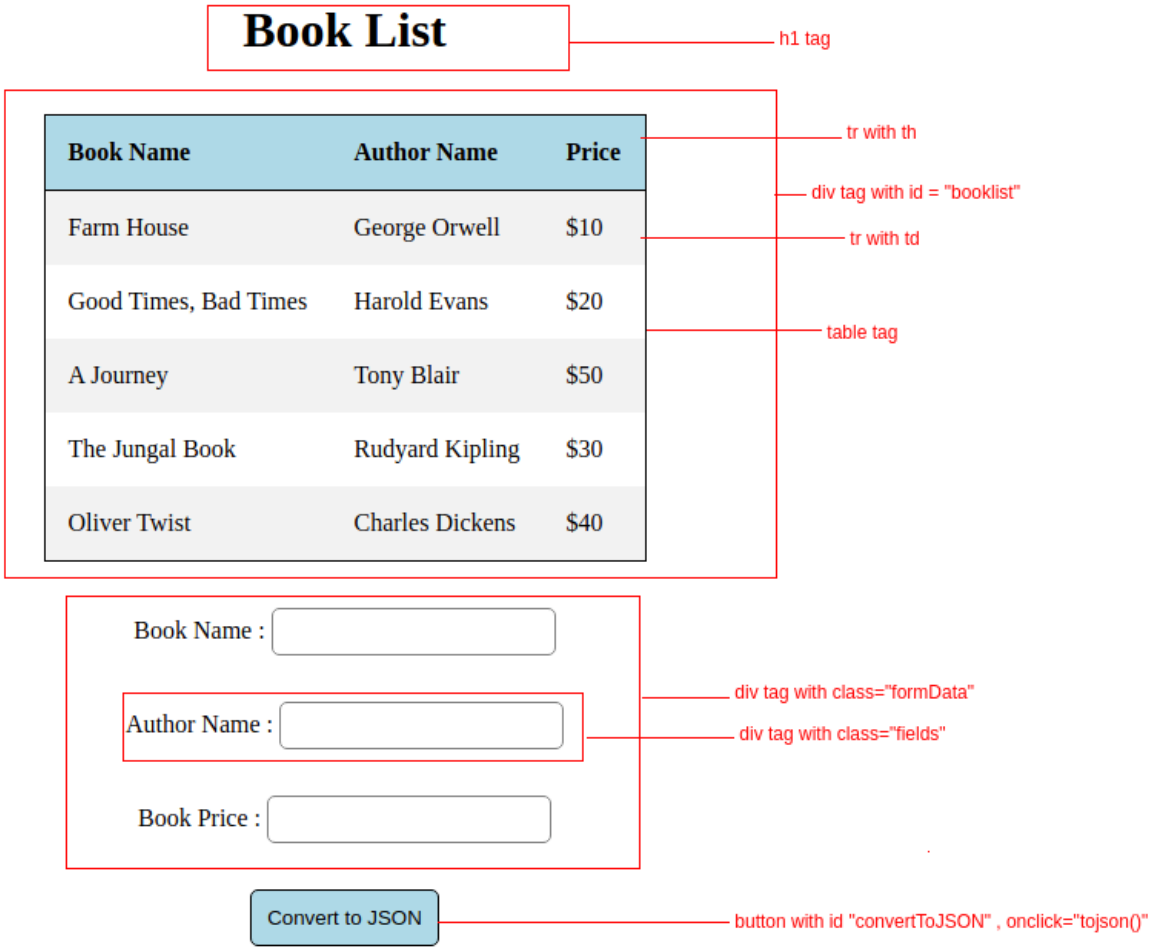
- Retrive the table data from xml during body onload and function name should be loadxml( ).
- Have a heading text 'Book List' inside the '**h1**' tag.
- The xml file name should be **booklist.xml.**
- Have a div tag with id “booklist”. And display the result table inside the div with id 'booklist'.
- Parse the xml elements and display it in tabular form as shown in screenshot.

- Have a button with id “convertToJSON” . While onclicking, the json format of entered book details should be displayed inside the div tag with id ‘jsonData’.
- Kindly give the id’s and class name as per the constraints and screenshot.

Template Code:

Click here to download the template code.

Sample Screenshot 1:



Sample Screenshot 2 :


```css
h1 {
  text-align: center;
  padding-top: 50px;
}

.btn {
  text-align: center;
}

#booklist {
  display: flex;
  justify-content: center;
}

table {
  border: 1px solid black;
  margin: 15px;
  border-collapse: collapse;
}

tr:first-child {
  background-color: lightblue;
```

```css
}

tr:nth-child(even) {
  background-color: #f2f2f2;
}

th {
  border-bottom: 1px solid black;
  text-align: left;
  padding: 15px !important;
}

td {
  padding: 15px;
}

#getBooklist {
  padding: 10px;
}

.formData {
  text-align: center;
}

.fields {
  padding: 15px;
}

#convertToJSON {
  margin-top: 15px;
  padding: 10px;
  border: 1px solid black;
  border-radius: 5px;
  background-color: lightblue;
}

#jsonData {
  margin-top: 15px;
}

input {
    border-radius: 5px;
    padding: 7px;
    border: 1px solid gray;
}
```

```html
<html>
<link rel="stylesheet" href="style.css">
<head>
   <script>
      function showTable(bookList){
         var result = document.getElementById("booklist");
         result.innerHTML = "";

         var table = document.createElement("table");
         var tbody = document.createElement("tbody");
```

```
        table.appendChild(tbody);
        result.appendChild(table);

        tbody.innerHTML = "<tr><th>Book Name</th><th>Author Name</th><th>Price</th></tr>";

        for (var book of bookList) {
           var name = book.getElementsByTagName("bookname")[0].textContent;
           var author = book.getElementsByTagName("authorname")[0].textContent;
           var price = book.getElementsByTagName("price")[0].textContent;

           tbody.innerHTML +=
           "<tr><td>" +
           name +
           "</td><td>" +
           author +
           "</td><td>" +
           price +
           "</td></tr>";

        }
     }
     function getXml(){
        var request = new XMLHttpRequest();
        request.open("GET", "booklist.xml", true);
        request.onreadystatechange = function () {
           if (request.readyState == 4 && request.status == 200) {
              var booklist = [];
              var doc = request.responseXML;
              var book = doc.getElementsByTagName("book");
              for (var i = 0; i < book.length; i++) {
                 booklist.push(book[i]);
              }
              showTable(booklist);
           }
         };
         request.send();
     }
     function tojson(){
        var name = document.getElementById("name").value;
        var author = document.getElementById("price").value;
        var price = document.getElementById("availableQuantity").value;

        var data = {
           "Book Name": name,
           "Author Name": author,
           "Price": price
        }

        document.getElementById("jsonData").innerHTML = JSON.stringify(data);
     }
  </script>
</head>
<body onload="getXml()">
  <h1>Book List</h1>
  <div id="booklist"></div>
```

```
    <div class="formData">
        <label>Book Name : </label>
        <input type="text" id="name" name="name" class="fields">
        <br>
        <br>
        <label>Author Name : </label>
        <input type="text" id="price" name="aname" class="fields">
        <br>
        <br>
        <label>Book Price : </label>
        <input type="text"  id="availableQuantity" name="price" class="fields">
        <br>
        <br>
    </div>
    <center>
        <button id="convertToJSON" onclick="tojson()">Convert to JSON</button>
    </center>
    <div id="jsonData"></div>
</body>

</html>
```