File IO:

Problem 6:
Create a class Employee having members as follows:

private int empNo
private String empName
private int empBasic

Parameterized constructor to initialize members.
Getter methods for all instance variables

- Create a class WriteEmployee.java having main method. Ask user to enter details of an employee and set them in an Employee object. Store details of this object in a file emp.txt.
- Read employee details from the file and display those details.

Problem 7:
Write a program to read the employees information, from "EmpRec.dat" file.

1001:Nishant:23000
1002:Devika:22000
1003:Joy:56000
1004:Praful:90000

Read the file, extract each record, convert to emp object and add to an array.

Day 7 : Collection Classes

Problem 1:

Create a class StudentOp.java having following members.

private ArrayList names – Arraylist of String type

public void setNames() – method to scan names of student and set in names arraylist
public void addName(String str)
public void searchName(String name) – method to search a student by name
public void searchName(int index) – method to print student name at an index
public void printNames( ) – method to print all names
public void removeName(String stuName) – method to delete a name

Create a class ArrayListDemo having main method. Create an object of StudentOps class and call all methods.

Problem 2:

Create a Emp.java that has empNo, empName, sal, desig

Create a class EmpOps.java.
Create a LinkedList of Employee objects.

Methods:
Public void addEmp(Emp emp)
Public void deleteEmp(int empno)
Public void displayAllEmps()
Public void incrSal() - Iterate thru list and increment salary of all employees by 10%
Public void displayEmpDetails(int id)
Public void displayEmpWithDesig(String desig)

Create TestEmpOps.java that has a main method. Test all methods of EmpOps.java

Problem 3:

Create a class Employee having members as follows:

private int empNo
private String empName
private int empBasic

Parameterized constructor to initialize members.
Getter methods for all instance variables

- Create a class WriteEmployee having main method. Ask user to enter details of an employee and set them in an Employee object. Store details of this object in a file emp.txt. Store it in the format empid : ename : sal
- Read employee details from the file and display those details.

Modify the above assignment : Read employee details, recreate the employee object, store into a List<Employee> . Iterate thru list and display all employee information.

Problem 2:

Create a set of 10 integers.
Perform the following operations:

- Display all elements in set
- Display the contents in reverse order
- Display the greatest element in the set that is less than the specified element.
- Display the greatest element in the set that is less than or equal to the specified element.
- Retrieve and remove the first and the last element of the set
- Obtain a subset of elements from set by proving the start and end values
- Which is the first element in the set?
- Return a portion of the set whose elements are less than or equal to given value.

Problem 3:

Create a login application that takes username and password. Now validate the user credentials against a database of user objects that are stored in a map.

Map<String, String>

Problem 3:

Create a class Student having following members:

private HashMap studNames – HashMap having rollno as key and name as value. Key and value are of type String
public void setNames( ) – method to set names in HashMap.
public void printNames( ) – method to print all names
public void getName(String key) – method to print value of a given key
public void printSize( ) – method to print size of HashMap
public void remove(String key) – method to remove a value of a given key

Create a class TestHashMap having main method. Create an object of Employee class and perform different operations on it.

Problem 4: Create a class Book having following members:
private int bookId;
private String bname;

private String author;
private double price;

Create a Interface BookDaoIntf.java
This interface will have the following methods:

- Void addBook(Book book) : adds a book object into the collection
- Void deleteBook(int bookid) : deletes book from collection having this id
- Void showAllBooks() : displays list of all books
- public Book getBookWithId(int bookid)
- public Book[] getBookWithAuthor(String author)
- public List<Book> getBooksContainingName(String str)
- public void applyDiscount(double discount) //for all books in set

✓ Create a class BookDao.java having a Treeset of book objects. It should implement Dao interface
✓ Create a class TestBookOp.java to test all methods of BookDao class

Problem 5:

Create a Account class – accId, acctName
Subclass the Account class to create 2 subclasses – SavingsAccount (minBal, bal), CurrAccount (bal, overdraftAmt)

Create a class AccountDao that does the following:

- Create 2 objects each of each subclass and store in a List
- Public void retrieveAllAccounts()
- Public void addAccount(Account account)
- Public void deleteAccount(int acctId)

JDBC:

1. Create a JDBC app that accepts emp id . Retrieve that emp info from db, convert retrieved info into emp obj and display
2. Create a method addEmp(Emp emp) which will insert emp obj into db

Problem-5:

Implement CRUD operation on Account objects.
Assume Account table to have : AcctID, AcctName, CreationDate, AcctType, AcctBal
From perspective of administrator, the application should be able to:

- Create and persist Account objects
- Read & display Account obj based on Acct ID
- Read & display Account obj based on Account Name
- Read & display all Account objs based on a specific type
- Delete an Account obj based on Acct ID. Upon deletion the book details must be stored in a file called AccountLog which will maintain the details such as AcctId, Acct Name and deletion_date. Optionally, remarks for why account has been deleted from banking system.

- Update bal of all Accounts whose creationdate is below given date by 10% as bonus/(customer loyalty)interest
- Display all Account details. Read into a list and iterate thru list to display records.

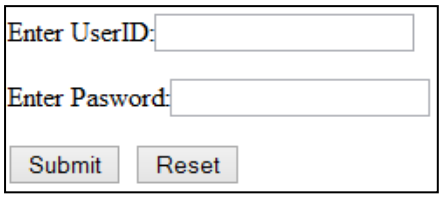
From perspective of a bank customer, application should be able to see menu:

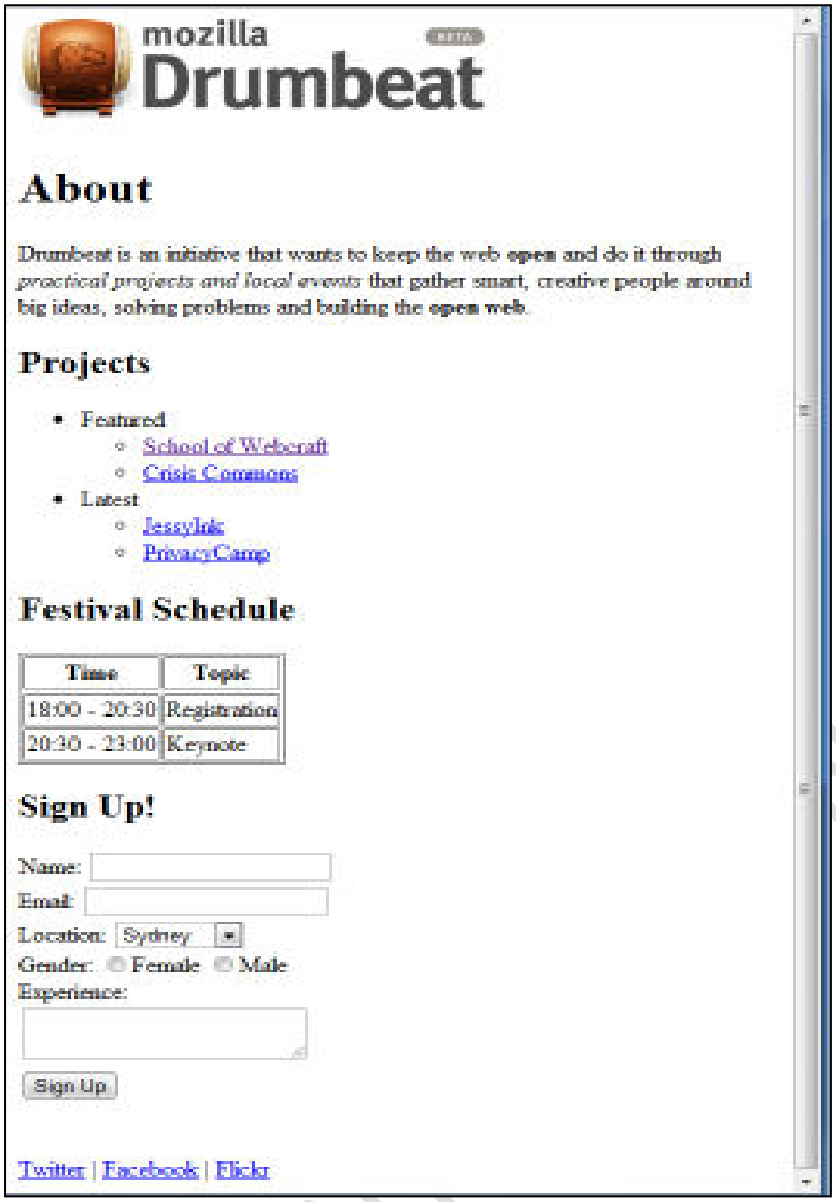
1. Display Menu to perform various transactions
    a. Balance enquiry

   b. Withdraw
   c. Deposit
   d. Exit
2. If customer opts for balance enquiry, then Account ID need to be entered by customer. Then display Account Holder Information and balance.
3. If customer opts for Withdraw, then Account ID and Amount has to be entered by customer. The Amount should to be deducted from balance and should display "Amount has been withdrawn successfully"
4. If customer opts for Deposit, then Account ID and Amount needs to be entered by customer and Amount need to be added into balance and should display "Amount has been Deposited successfully".
5. If customer opts for Exit, customer should be able to quit from application.

HTML:

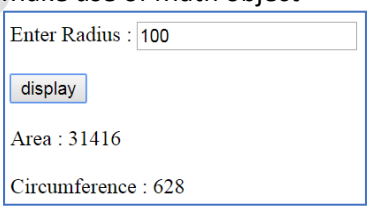
Create the following form in HTML:



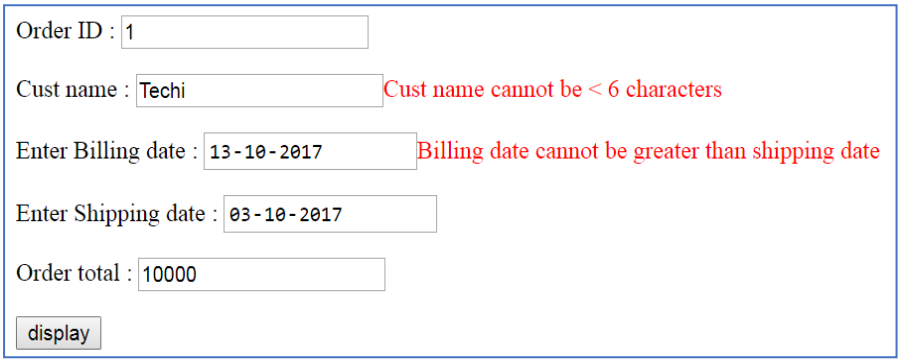


Javascript:

1. Write a program to accept three positive numbers and output the largest of them.
2. Accept an integer value and a message from user and print the message that many number of times.
3. Write Javascript function to find the sum and average of all passed parameters
4. Accept and store names of 6 employees into array and display in sorted order of names.
5. Create an array that holds 4 employee objects and displays each emp detail in a table. Emp details could be empid, empname and salary.
6. Create a HTML form that accepts radius and displays the area and circumference of a circle. Make use of Math object



7. Create a HTML page that accepts order details:

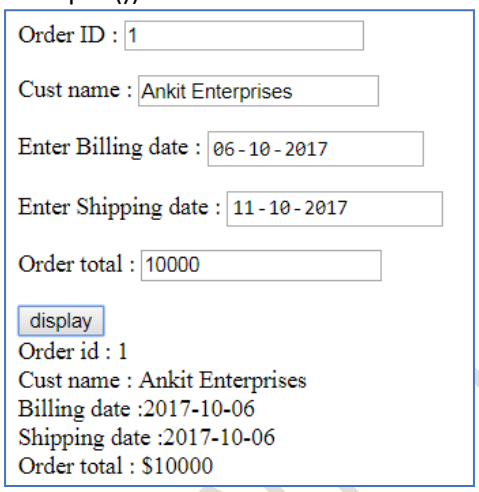


Following are validations required:

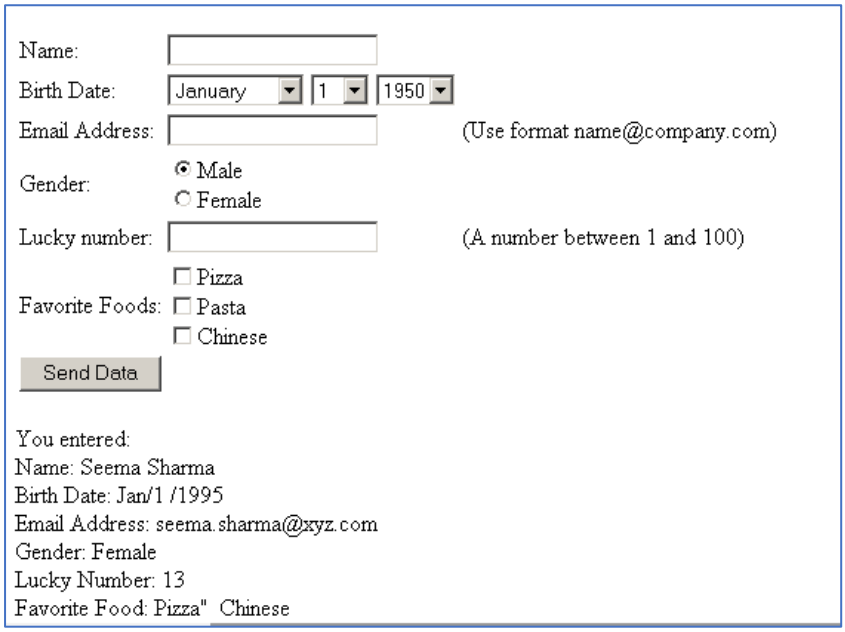
- All fields are mandatory
- Cust name must be greater than 5 characters.
- Billing date must be lesser than shipping date

If all goes well, display as follows;

~~On clicking the display button, all form information must be saved into an Order object. The details of this object must be displayed in a new window in a neat tabular fashion(hint : use window.open())~~



8. Create a HTML page that will display the following form and on clicking the button, the details will be listed as shown. Also create an object with all details from form, show them in new window thru the newly created object



Problem statement-3 : Use the HTML page created in above Problem.

The following requirements must be met:

- Name: Required, only alphabets allowed, maximum length is 10.
- Lucky number: Required, must be of 4 digits.
- All other fields are mandatory
- Display appropriate messages in case of invalid data.