Day 1 (24th July 2024)

1) Write a java program to print Fibonacci series up to 10 numbers and print sum of those numbers starting from: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34

2) Write a java program to accept number of seconds as integer and display total number of minutes, hours and days out of it.

3) Identify the classes, attributes, methods and the relationships between the classes for the below use case. You can show the relationship by using class diagrams.

   Vehicle manufacturing company wants to store details of their vehicles. There are 2 types of vehicles: TwoWheeler and FourWheeler. Both have some common properties like vehicle id, seating capacity. For TwoWheeler, the application should store an additional property handle type. For FourWheeler, the application should store an additional property steering type. Application also maintains details of engine used in the vehicles like engine capacity, fuel type.

Day2 (25th July 2024)

Write a java program which will accept elements for an array from user. Create a separate class which will have methods to perform following operations:

1. Sort the array using selection sort.
2. Accept an element and remove that element from the array. If element is not present, then corresponding message should be displayed.
3. Accept a position and remove element from that position in the array.
4. Accept an element and create an array of all elements which are larger than the element and display all the elements
5. Accept an element and position, then insert that element in that position in the array.

1) Create a class to calculate area of circle with one data member to store radius. Write following methods in the class:
   1. init – to accept value of radius from user.
   2. area – to calculate and return area of the circle.
   3. perimeter – to calculate and return perimeter of the circle.
   4. diameter – to calculate and return diameter of the circle.

2) Write a program to test the below methods of String class:
   a. lastIndexOf
   b. replace
   c. split
   d. isBlank
   e. stripTrailing
   f. substring

Day 3 (26th July 2024)

Note: All the classes should be part of packages. Create proper package hierarchy

1) A Job tracker application is used for tracking the activities of a team. Each activity is represented as a Job class. Create a class Job with the attributes job name, owner, effort required(in months), month of creation, year of creation, status (not started/ work in progress/completed). Write necessary methods to accept and display the information. Create the constructors based on the below rules.
   a. Job name and owner are mandatory fields and should be supplied at the time of creating a class
   b. Compiler should raise an error when we try to create Job object with out passing any parameters.
   c. Write a constructor which accepts all the attributes as parameters while creating the object. From this constructor call the constructor (mentioned at point a) to initialize mandatory fields
2) In the above class create a static variable **jobsCount.** Write necessary methods to get the values. Every time an object of Job is created, increment the value of jobsCount variable.
3) Create a jobId field. Make this variable as a readOnly method (make it private and write only getter method). Generate jobId value by using the below formula

jobId=”jobName”+”_”+jobsCount

eg. if the Job name is “Maintain Server” and the value of jobsCount variable is 31, then store Maintain Server_31 in jobId

4) Class PriorityJob contains all the attributes of Job class and other attributes to store the priority of the job (low, medium, high) and monitoredBy field to store the name of the person monitoring Job Class MultiOwnerJob contains all attributes of Job class and other name of the second owner.
   a. Write necessary classes, constructors and methods for storing and displaying additional information.

   Hint: use super keyword to call the methods/constructor of super class.

5) In the Job class write a method called showDetails() which returns a String by concatenating all the field values.

Day 4 (29th July 2024)

1) Override showDetails() method in the sub classes to include the details of additional fields. Use super key word to call the methods of super class if required.
2) Create an array of Job class and store Objects of Job, PriorityJob, and MultiOwnerJob classes in the array.
3) Using a single for loop try traverse the above array and call the method showDetails() on all the objects of the array. Understand the concept of runtime polymorphism.
4) Check is it possible to call all the methods of PriorityJob and PriorityJob while traversing the array. If not use typecasting to achieve the above task.
5) Complete the below tasks

a. Create Interfaces
b. Create inheritance by extending other interfaces
c. Creating class by extending another class and implementing more than 1 interface
d. Create a reference variable of an interface.
e. Create a class implementing above interface.
f. Store the object created in step e in the reference variable created in step d.
g. Call the methods by using interface reference

6) A pizza delivery outlet uses a software for maintaining information about their pizzas. Structure of the pizza class is given below

```
Class Pizza {
   String pizzaName // unique field
    String description;
   Int sizeInCms
   String majorIngredientOne
   String majorIngredientTwo
   String majorIngredientThree
   int    weight;
   float price;

  public void preparation(){  //this method displays the procedure for preparation }
  // identify other necessary methods and properties
}
```

Day 5 (30th July 2024)

Develop a layered application so that view layer is responsible for user interaction and is only responsible for accepting and displaying details from/to user. View layer will communicate with storage layer for all CRUD operations. Class structures, method information and responsibilities of each class are given below. Create a main class to test your application.

View Layer

CustomerView (Class)

addPizzaDetailsAndStore()
displayPizzaDetailsByName()
printPizzaNamesBySize()
Responsibility :
This class accepts the details from the user and calls the methods on Storage layer.
Call PizzaStorageFactory.getPizzaStorage() to get the Storage implementation object

Storage layer

PizzaStore (Interface)

void addNewPizza(Pizza e)
Pizza getPizzaByName(String pizzaname)
Pizza[] getPizzasCountBySize(int size)

PizzaStoreImpl (Class)

Responsibility:
This class implements PizzaStore

Create an array to store the Pizza objects and provide the proper implementation to the methods in the interface

PizzaStoreFactory (Class)

public static PizzaStore getPizzaStore()

Responsibility:
This method is used to get the object

Day 6(31st July 2024)

1) Test the below concepts by writing necessary classes and methods
   a. Single try .. catch .. finally block
   b. Multiple catch block
   c. Nested try catch blocks
   d. Usage of throw and throws keywords.
2) Create a checked exception **InvalidRangeException**. Override all the constructors from the super class
3) Create a RuntimeException **InvalidInputException**. Override all the constructors from the super class

4) Create the below two classes. DataCollector class is dependent on StatisticalUtility class to get the average of the data collected from user.

DataCollector

public int getResult(int data[], int min, int max, int v1, int v2) throws InvalidInputException.

Responsibility:
This method accepts list of integers, minimum value, maximum values . Pass these three values to the getAve get average. Return the result by using the below formula
Result=(average * v1)/v2;
In case of InvalidRangeException throw InvalidInputException by wrapping InvalidRangeException.

StatisticalUtility

public int getAverageValue(int data[], int min, int max) throws InvalidRangeException.

Responsibility:
This method accepts array of integers (data), and two integers (min and max). Method returns average of all the elements of the data array only if the average value is in between min and max values.
If the average value is outside the range of min and max then throw InvalidRangeException

Create the Main class for accepting the data from the user and call the methods on DataCollector object. In case of any exceptions catch the exceptions and display the appropriate message to the user.

5) In question No. 4 modify the interface definition as mentioned below.

Storage (Interface)

```
void addNewPizza(Pizza e) throws PizzaAlreadyExistsException
Pizza getPizzaByName(String pizzaname) throws NoPizzaFoundException
Pizza[] getPizzasCountBySize(int size) throws NoPizzaFoundException
```

- While adding the Pizza in the array if the Pizza with the same name already exists in the array then raise PizzaAlreadyExistsException
- In the search operation if the Pizza is not found then raise NoPizzaFoundException.

Modify other classes accordingly.

Day 7(1st August 2024) :

Note : use menu.txt for solving this assignment

1. Create an implementation class PizzaStoreFileImpl implementing PizzaStore interface   given day 5 assignment
2. Add an Pizza[] menu as an instance variable in PizzaStoreFileImpl class
3. Initialize the menu in constructor of PizzaStoreFileImpl class – size must be equal to no of lines in menu.txt
4. Implement the unimplemented methods PizzaStore in PizzaStoreFileImpl class as described below.

Method	Description
void addNewPizza(Pizza e) throws PizzaAlreadyExistsException	Read each line from menu.txt , create a Pizza Object and add to menu array
Pizza getPizzaByName(String pizzaname) throws NoPizzaFoundException	Read a pizza name from user and check if a pizza available in array and return Pizza obejct

Day 8 (2nd Aug 2024)

1) Create a java application to test the following
    e. Create threads by extending Thread class and Runnable interface
    f. Use join(),sleep() methods
    g. Synchronized method

2) Create a java application to test the below features of collection framework
    a. Store the objects and retrieve objects in ArrayList, Vector, LinkedList
    b. Write a function which accepts array of objects returns the List object with the same data.

3) In the previous question make the below modifications to the Pizza class.
    h. add **insertion date** field of type Date
    i. Override hashCode and equals method
    j. Implement comparable interface. While comparing two objects consider pizza name filed.
4) Create different implementation classes to Storage interface as mentioned below
    a. PizzaListStoreImpl : Use Array list for storing the pizza objects
    b. PizzaSortedStoreImpl : Use TreeSet for storing the Pizza objects
    c. PizzaMapStoreImpl : Use HashMap for storing the Pizza objects as values and pizza name as keys.
5) Modify getPizzaStore() as mentioned below. Method should accept integer as a parameter
    a. PizzaStore getPizzaStor(int code) method
    b. If code =1 then return PizzaStoreImpl object
    c. If code = 2 then return PizzaListStoreImpl
    d. If code=3 then return PizzaSortedStoreImpl
    e. For any other value return PizzaMapStoreImpl

Day 9 (5th August 2024) :

1) Create a Functional interface ResultFinder with following method
   Public String result(Student student);

Create a class student with following fields

```
Int rno
Int name
Int marks
```

And the following method

```
Public  boolean  getResult(ResultFinder resultfinder){
    Return resultFinder.result();
}
```

Create another class University

```
Class University {


    Public void printStudentResult (List<Student> students, ResultFinder rf) {

      For(Student s : students){

         If(rf.getResult(s)){
            System.out.println(s.getId()+" "+s.getName()+" "+s.getMarks+" "+"PASS");
            }
            Else
            {
            System.out.println(s.getId()+" "+s.getName()+" "+s.getMarks+" "+"Fail");

            }

            }

}
```

Create Main Class

```
Public FunctonalDemo{

Public static void main(String[] args){}

List<Student> studentList = Arrays.asList(

new Student(1,"raj",51),

new Student(2,"pradeep",41),

new Student(3,"Sheetal",55),

);


// print all PASSED Students using by calling the printStudentResult method

// print all FAILED Students using by calling the printStudentResult method


            //Note : Implement lambda expression for ResultFinder interface and pass that as argument to
printtStudentResult method s

}

}
```

Create a list of 10 students using above student class and find the list of employees whose names starts with "a"
Note : Use Stream API to solve above question

Day 10 : (6th August 2024)

1) Write the proper SQL instructions for the below actions
    k. Create a table Person (person id, person name, location, date of birth). Person id is a primary key
    l. Create a table Person Learnings (record id, name of course, duration, date of completion, person id) record id is a primary key and person id is foreign key .

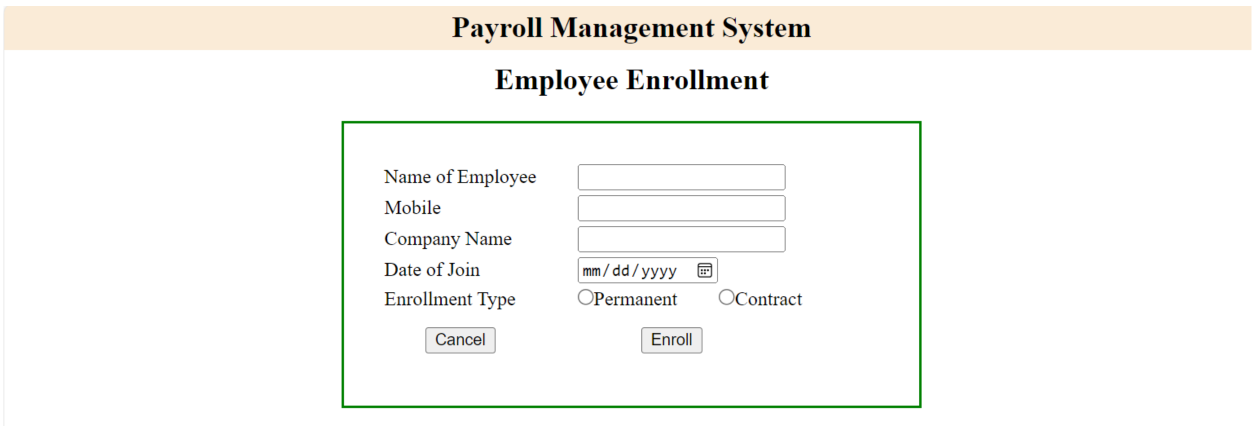
m. Insert, update and delete the rows in the above table.
n. Write SQL queries for
    i. List all persons
    ii. List all persons who are located in “India” or “China”
    iii. List all persons if their name starts with ‘A’
    iv. List all Person id, person name, course name and date of completion
    v. List all names of persons who had completed “Cloud” course

2) Write a class for PersonStore and write all the necessary methods for below operations.
    a. Public void addProduct(Person p) throws PersonAlreadyExistsException (Raise this exception the data base throws Record already exists exception)
    b. Public void updatePerson(Person p)
    c. Public List<Person> getAllPersons()
    d. Public List<Persont> getPersonsByCity(String location)
    e. Public void addTeam (Person p1, Person p2) throws PersonsNotAddedException (Raise this exception even if we are not able to add any one Person. Use transactions for this operation)
    Note : Implement all the above functionality with Lambda Expressions

Day 11 : No Assignment on this day(7th August 2024)

Day 12 : (8th August 2024)

1) Write the HTML file to create the below web page



Payroll Management System

Employee Enrollment

Name of Employee	
Mobile	
Company Name	
Date of Join	mm/dd/yyyy
Enrollment Type	○Permanent ○Contract
Cancel	Enroll

2) Validate all the fields as per following description.

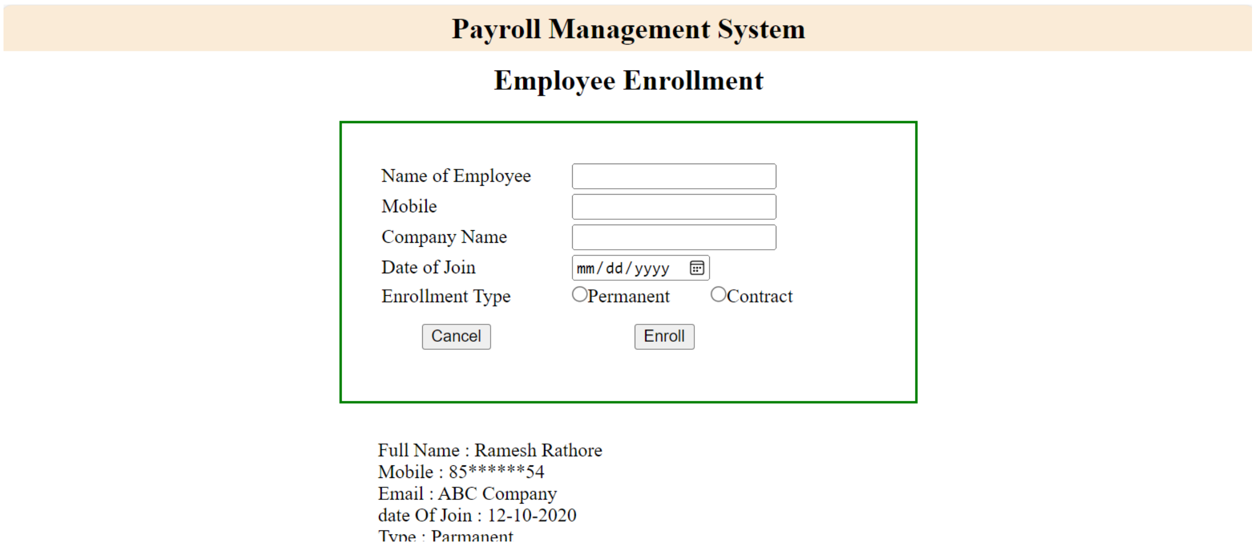
- All the fields should not be empty.
- Mobile number should be a 10 digit number.
- Mobile number should start with 6 / 7/ 9/ digit only

Day 13 (9th August 2024)

Set 2 :

Note : Use the Day 12 question solution for solving this assignment

1. Display all form data on the same web page as shown below if all data are validated to true .



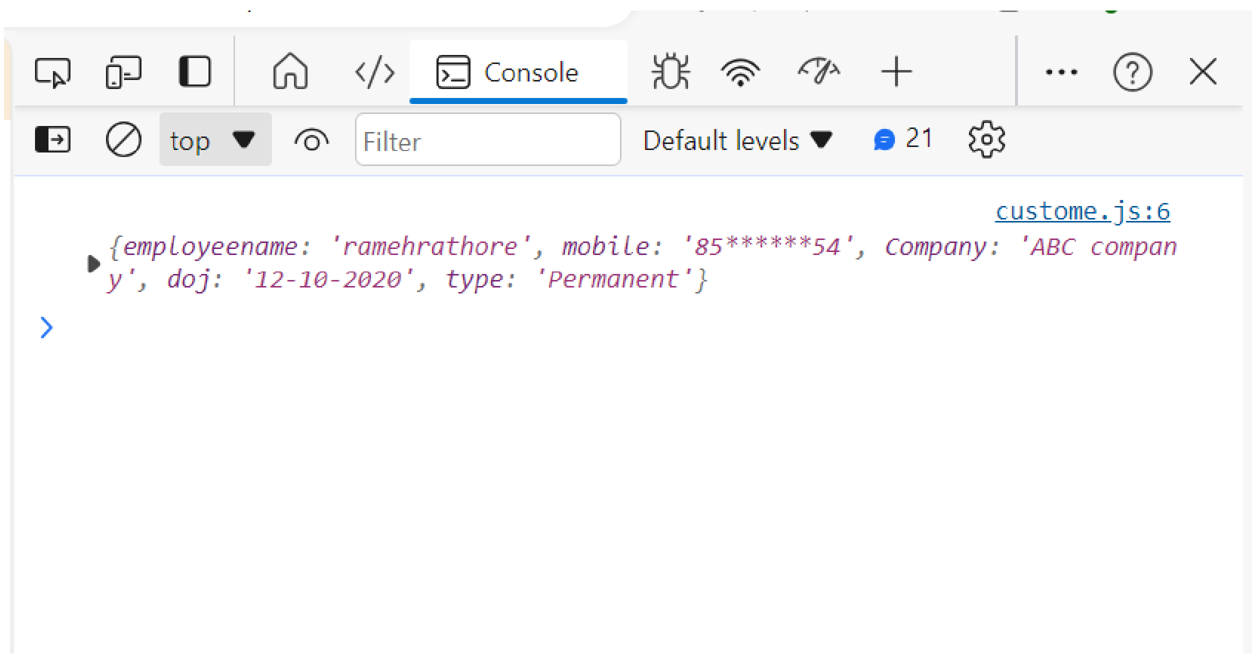
Payroll Management System

Employee Enrollment

Name of Employee	
Mobile	
Company Name	
Date of Join	mm/dd/yyyy
Enrollment Type	○Permanent ○Contract

```
Full Name : Ramesh Rathore
Mobile : 85******54
Email : ABC Company
date Of Join : 12-10-2020
Type : Parmanent
```

2. Create a JSON Object with the form data and display on browser console.



```
{employeename: 'ramehrathore', mobile: '85******54', Company: 'ABC company', doj: '12-10-2020', type: 'Permanent'}
```

Day 14: (12th August 2024)

Write and execute the Junit Test Cases for assignment 4,5,6