1. Create class Employee with fields empno, ename, job, showInfo() and create another class with name EmployeeContainer which contains:

addEmployee(Employee emp);

deleteEmployee(int empno);

viewEmployee(int empno);

viewEmployees();

addEmployee(): should add the employee to the Set in the container class, if the employee is already existing it should display " employee already exists".

deleteEmployee():It should search for empno in the Set, if empno is found the employee object has to be removed, otherwise it should display "Sorry! Employee is not found".

viewEmployee():It should search for empno in the Set, if empno is found, the employee details has to be displayed , otherwise it should display "Sorry! Employee is not found".

viewEmployees(): First make sure Set is not empty, if it is not empty you should display all the employees from Set, otherwise it should display "No elements are added to the Set".

2. A Sports club keeps separately the record of the players in two different games (football, cricket). There are some players who play in both the games. The Sports Club needs to keep track of the player playing in both the games. Write a method to accept the two lists containing the names of the players playing in the two games. The method should find out the names of player present in both the list and return the names as a Sorted ArrayList.

```
public class Player{

        String name,email,city,state;

        int age;

        ----------

 }

public List<String> getPlayers(List<Player> football, List<Player> cricket){

                //Write your logic
}
```

3. Create class PhonebookClient having user interface like

1. Add new phone book entry
2. Search name
3. Quit.

Create another

```
public class PhoneBook{

        Map<String,String> map=new HashMap<>();

        public  void addDetails(String phno,name){
                //Write your logic
        }

        public String getName(String phno){
                //Write your logic
        }
}
```

addDetails(): It should add the phno,name to map object

getName(): It should search the given phone number in the map and should return the person name, if not it should return “Sorry! No person found with the given number”.

Option 1: It allows add Phno and Name.

Option 2: It has to take Phno as input from the user and based on that it should return Name

Option 3: Will terminate the program

.

4. Create a class with name Question with:

String question, option1, option2, option3, option4, answer, Question (question, option1, option2, option3, option4, answer).

Create another class QuestionContainer with List<Question> list, int ccount, wcount;

```
QuestionContainer( ){
        list=new ArrayList<Question>();
        //Try to add 5 questions to List like
```

list.add("java is programming or platform","A. Programming","B. Platform", "C.Both","D.None of the above","C");

}

void beginTest( ){

//It should display questions one by one and it has to prompt the correct answer from the user. User input can be "A","B","C","D" and it should validate given answer with the correct answer. If answer is correct then count has to increase otherwise it should increase wcount. If wrong input is given by user next question shouldn't display but it should ask user to enter correct input.

}

showResult()

It should display the result of the test

Like

Total questions: value

Correct : value

Wrong : value

Result : Pass/Fail (>40% pass otherwise fail)

Create Another class ExamClient with main method, in which you create an instance of QuestionContainer; invoke beginTest( ) and showResult( ).

5. Create class MemberIdGenerator

private static int count=5005;

public String suffix="IND";

public String prifix="ASHO";

public static String getMemeberId( ){

// It should increment the counter value and return the id like "ASHO5006IND";

}

Create another class with name Member with String mid, name, city, country

Member(mid, name, city, country)

display(): to display the content of the member object.

Create another class MemeberContainer with Member member, List<Member>.

addMemeber (name, city, country): It should get the "mid" from the MemeberIdGenerator, create member object and initialize by invoking parameterized constructor. Add this member object to List object.

deleteMember(String id): It should search the member based on the given id, if found it should be deleted from the List ,otherwise it should display message "Member not found with given id";

searchById(String id): First make sure that the list is not empty, if it is empty it should display list is empty, otherwise It should search the member based on the given id, if the id is found it should display the member details, otherwise message should be "Sorry! No member found"

searchMember(int id,String name): First make sure that the list is not empty, if empty it should display list is empty, else it should search the member based on the given id or name. If the id or name is found it should display the member details, if not message should be "Sorry! No member found"

displaySortingOrder(String s) : if String s is Desc, it should display all the members in Descending order alphabetically based on name. If it is Aesc, it should display all the members in Ascending order alphabetically based on the name.

Hint: use the java.uti.Comparator interface API

Create class MemberClient with main method, and use the menu driven approach

1.Addmember

2.deleteMember

3.SearchById

4.searchByNameAndId

5.displaySortingOrder

6.exit

Create MemberContainer instance, and try to perform all the options.