**Home Assignment IV**

1)Create a Course class with three fields id, name and author. Use parameterised constructor to set the values and write the getters to retrieve the data. Override the toString( ) method too. Create another CourseController to return list of all objects as an array(the end point is /courses). Test the classes using Spring Boot

2)Create a Student class with three fields sno, sname. Use parameterised constructor to set the values and write the getters to retrieve the data. Override the toString( ) method too. Create another StudentController to return list of all objects as an array(the end point is /students). Test the classes using Spring Boot

3)Create a Library with three fields lno, lname and lauthor. Use parameterised constructor to set the values and write the getters to retrieve the data. Override the toString( ) method too. Create another LibraryController to return list of all objects as an array(the end point is /libraries). Test the classes using Spring Boot

4)Create a Bank with three fields bno, bname and bbranch. Use parameterised constructor to set the values and write the getters to retrieve the data. Override the toString( ) method too. Create another BankController to return list of all objects as an array(the end point is /banks). Test the classes using Spring Boot

5)Create a Hospital with two fields hname,haddress. Use parameterised constructor to set the values and write the getters to retrieve the data. Override the toString( ) method too. Create another HospitalController to return list of all objects as an array(the end point is /hospitals). Test the classes using Spring Boot

6)Create a Class School with three fields sname,saddress,scode. Use parameterised constructor to set the values and write the getters to retrieve the data. Override the toString( ) method too. Create another SchoolController to return list of all objects as an array(the end point is /schools). Test the classes using Spring Boot

7)Create a Class City with two fields cname and ccode. Use parameterised constructor to set the values and write the getters to retrieve the data. Override the toString( ) method too. Create another CityController to return list of all objects as an array(the end point is /citys). Test the classes using Spring Boot

8)Create a Class Bank with three fields bno, bname and bbranch. Use parameterised constructor to set the values and write the getters to retrieve the data. Override the toString( ) method too. Create another BankController to return list of all objects as an array(the end point is /bankls). Test the classes using Spring Boot

9)Create a Class Subject with four fields scode, sname,smarks and sfaculty. Use parameterised constructor to set the values and write the getters to retrieve the data. Override the toString( ) method too. Create another SubjectController to return list of all objects as an array(the end point is /subjects). Test the classes using Spring Boot

10)Create a Class Department with three fields dname, dhod and dcode.. Use parameterised constructor to set the values and write the getters to retrieve the data. Override the toString( ) method too. Create another DepartmentController to return list of all objects as an array(the end point is /departments). Test the classes using Spring Boot

11)Create a Class Book with three fields bno, bname and bprice. Use parameterised constructor to set the values and write the getters to retrieve the data. Override the toString( ) method too. Create another BookController to return list of all objects as an array(the end point is /books). Test the classes using Spring Boot

12)Create a Class Customer with four fields cno, cname,cmail and caddress. Use parameterised constructor to set the values and write the getters to retrieve the data. Override the toString( ) method too. Create another CustomerController to return list of all objects as an array(the end point is /customers). Test the classes using Spring Boot

13)Create a Class Order with three fields oid, oname,odate. Use parameterised constructor to set the values and write the getters to retrieve the data. Override the toString( ) method too. Create another OrderController to return list of all objects as an array(the end point is /orders). Test the classes using Spring Boot

14)Create a Class Costume with two fields ccolor, cname. Use parameterised constructor to set the values and write the getters to retrieve the data. Override the toString( ) method too. Create another CustomerController to return list of all objects as an array(the end point is /customers). Test the classes using Spring Boot

15)Create a Class Watch with three fields wid, wbrand and wprice. Use parameterised constructor to set the values and write the getters to retrieve the data. Override the toString( ) method too. Create another WatchController to return list of all objects as an array(the end point is /watches). Test the classes using Spring Boot

16)Create a Class Vehicle with three fields vno, vtype and vwheels. Use parameterised constructor to set the values and write the getters to retrieve the data. Override the toString( ) method too. Create another VehicleController to return list of all objects as an array(the end point is /vehicles). Test the classes using Spring Boot

17)Create a Class Mobile with three fields mid, mname and mprize. Use parameterised constructor to set the values and write the getters to retrieve the data. Override the toString( ) method too. Create another MobileController to return list of all objects as an array(the end point is / Mobiles). Test the classes using Spring Boot

18)Create a Class Employee with three fields eno, ename and department. Use parameterised constructor to set the values and write the getters to retrieve the data. Override the toString( ) method too. Create another EmployeeController to return list of all objects as an array(the end point is /employees). Test the classes using Spring Boot

19)Create a Class Pen with three fields pid, pbrand, pcolor. Use parameterised constructor to set the values and write the getters to retrieve the data. Override the toString( ) method too. Create another PenController to return list of all objects as an array(the end point is /pens). Test the classes using Spring Boot

20)Create a Class PanCard with three fields pno, pname and paddress. Use parameterised constructor to set the values and write the getters to retrieve the data. Override the toString( ) method too. Create another PanCardController to return list of all objects as an array(the end point is /pans). Test the classes using Spring Boot

21)Create a Class Aadhar with three fields ano, aname and address. Use parameterised constructor to set the values and write the getters to retrieve the data. Override the toString( ) method too. Create another AadharController to return list of all objects as an array(the end point is /aadhars). Test the classes using Spring Boot

22)Create a Class Bus with four fields bno, btype,bdestination and bseats. Use parameterised constructor to set the values and write the getters to retrieve the data. Override the toString( ) method too. Create another BusController to return list of all objects as an array(the end point is /buses). Test the classes using Spring Boot

23)Create a Class Plane with three fields pno, pname,psource,pdestination. Use parameterised constructor to set the values and write the getters to retrieve the data. Override the toString( ) method too. Create another PlaneController to return list of all objects as an array(the end point is /planes). Test the classes using Spring Boot

24)Create a Class College with three fields ccode, cname and caddress. Use parameterised constructor to set the values and write the getters to retrieve the data. Override the toString( ) method too. Create another CollegeController to return list of all objects as an array(the end point is /colleges). Test the classes using Spring Boot

25)Create a Class Faculty with three fields fid, fname and fsalary. Use parameterised constructor to set the values and write the getters to retrieve the data. Override the toString( ) method too. Create another FacultyController to return list of all objects as an array(the end point is /faculties). Test the classes using Spring Boot

26)Create a Class Hostel with three fields hno, hname and hrent. Use parameterised constructor to set the values and write the getters to retrieve the data. Override the toString( ) method too. Create another Hostel Controller to return list of all objects as an array(the end point is / hostels). Test the classes using Spring Boot

27)Create a Class Shoe with four fields sid, sbrand, scost and scolor. Use parameterised constructor to set the values and write the getters to retrieve the data. Override the toString( ) method too. Create another ShoeController to return list of all objects as an array(the end point is /shoes). Test the classes using Spring Boot

28)Create a Class Train with four fields tno, tsource, tdestination and tname. Use parameterised constructor to set the values and write the getters to retrieve the data. Override the toString( ) method too. Create another TrainController to return list of all objects as an array(the end point is /trains). Test the classes using Spring Boot

29)Create a Class Laptop with four fields lid, lram, lcost and lcolor. Use parameterised constructor to set the values and write the getters to retrieve the data. Override the toString( ) method too. Create another LaptopController to return list of all objects as an array(the end point is /laptops). Test the classes using Spring Boot

30)Create a Class Speaker with three fields sid, scost, srange. Use parameterised constructor to set the values and write the getters to retrieve the data. Override the toString( ) method too. Create another SpeakerController to return list of all objects as an array(the end point is /speakers). Test the classes using Spring Boot

31)Create a Class Movie with three fields mname, mgenre, mlanguage. Use parameterised constructor to set the values and write the getters to retrieve the data. Override the toString( ) method too. Create another MovieController to return list of all objects as an array(the end point is /movies). Test the classes using Spring Boot

32)Create a Class Project with three fields pname, pdomain, pstatus. Use parameterised constructor to set the values and write the getters to retrieve the data. Override the toString( ) method too. Create another ProjectController to return list of all objects as an array(the end point is /projects). Test the classes using Spring Boot

33)Create a Class Product with three fields pid, pname, pprice. Use parameterised constructor to set the values and write the getters to retrieve the data. Override the toString( ) method too. Create another ProductController to return list of all objects as an array(the end point is /products). Test the classes using Spring Boot

34)Create a Class Event with four fields eid, ename, edate, elocation. Use parameterised constructor to set the values and write the getters to retrieve the data. Override the toString( ) method too. Create another EventController to return list of all objects as an array(the end point is /events). Test the classes using Spring Boot

35)Create a Class Inventory with four fields ino, iname, icategory, icost. Use parameterised constructor to set the values and write the getters to retrieve the data. Override the toString( ) method too. Create another InventoryController to return list of all objects as an array(the end point is /inventories). Test the classes using Spring Boot