**Emandi User Update**

Emandi Corporation planned to improve their registration system using hibernate.

Write a program to update the existing user in the database using hibernate and display the details of the user in the ascending order of user id.

**[Strictly adhere to the object-oriented programming specifications given in the problem statement. Template code is provided to ease the input-output process.]**  
**Use either Class.hbm.xml or annotations for mapping**

Consider the class **User**with the following attributes

|  |  |
| --- | --- |
| **Datatype** | **Variable** |
| Integer | authenticate |
| Integer | userid |
| String | username |
| String | password |
| String | name |
| String | email |

The methods for **getters**, **setters**and **constructors**are given in the template code.

Consider the class **UserDAO**and define the following methods

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| public void updateUser(Integer id, String pass) | This method is used to update the user password if the id is present in the database. Else print “**No such id exists**” and stop the execution. |
| public List<User> listDetails() | This method is used to return the list of user details from the database in  the ascending order of user id. |

Consider the driver class **Main**. In the main method, read inputs and test above classes.

**Table Properties:**

create table "user"(  
   userid NUMBER(10) NOT NULL,  
   authenticate NUMBER(10) NOT NULL ,  
   username VARCHAR2(20) default NULL,  
   password VARCHAR2(20) default NULL,  
   name VARCHAR2(20) default NULL,  
   email VARCHAR2(20) default NULL,  
   PRIMARY KEY (userid)  
);

For eclipse IDE users, use the below link to download the SpringHibernate jars to configure with your local spring project.  
[Spring Hibernate Jars](https://hcl.e-box.co.in/uploads/2696/Resource/JARS/springHibernateJar.zip)

**Download the oracle jar file in the below link.**  
[Oracle jar](https://hcl.e-box.co.in/uploads/File/ojdbc6.jar.zip)

**Download the template code in the below link**.  
[Code Template](https://hcl.e-box.co.in/problem/showTemplateCodeSet/5715?op=download&langid=6)

**HibernateUtil.java**

Use the below code to retrieve the connection details from oracle properties to establish a connection  
  
            ResourceBundle rb = ResourceBundle.getBundle("oracle");  
            Configuration cfg = new Configuration();  
            cfg.configure("hibernate.cfg.xml");  
            cfg.setProperty("hibernate.connection.url", rb.getString("db.url"));  
            cfg.setProperty("hibernate.connection.username", rb.getString("db.username"));  
            cfg.setProperty("hibernate.connection.password", rb.getString("db.password"));

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

**Inorder to disable the log information in console,set the logger property -OFF in main using the below code:**

Logger log = Logger.getLogger("org.hibernate");  
log.setLevel(Level.OFF);  
System.setProperty("org.apache.commons.logging.Log",  
"org.apache.commons.logging.impl.NoOpLog");

**Note:**

* All the input and output statements must be included in the main java class.
* Create Static details of users in the database.
* Enter the user id and the password to be changed in the database.
* Display the user details remaining in the database in the ascending order of user id.
* User id must be auto-generated.

**Finally in Main class use System.exit(0) for successfull termination.**

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

**Sample Input and Output 1:**

Name: ZaraAli  
UserId: 1  
UserName: Zara  
Password: Ali  
Email-ID: zara123@gmail.com  
Authenticate: 1  
Name: DaisyDas  
UserId: 2  
UserName: Daisy  
Password: Das  
Email-ID: daisy456@gmail.com  
Authenticate: 2  
Name: JohnPaul  
UserId: 3  
UserName: John  
Password: Paul  
Email-ID: john789@gmail.com  
Authenticate: 3  
Name: TharaVarthan  
UserId: 4  
UserName: Thara  
Password: Varthan  
Email-ID: thara123@gmail.com  
Authenticate: 4  
Name: ShanMathi  
UserId: 5  
UserName: Shan  
Password: Mathi  
Email-ID: shan456@gmail.com  
Authenticate: 5  
Name: HaveenaHavi  
UserId: 6  
UserName: Haveena  
Password: Havi  
Email-ID: havi789@gmail.com  
Authenticate: 6  
Enter the id and password to be updated  
**5**  
**shanmathi555555**  
Password for Id 5 is updated  
Name: ZaraAli  
UserId: 1  
UserName: Zara  
Password: asdf  
Email-ID: zara123@gmail.com  
Authenticate: 1  
Name: DaisyDas  
UserId: 2  
UserName: Daisy  
Password: Das  
Email-ID: daisy456@gmail.com  
Authenticate: 2  
Name: JohnPaul  
UserId: 3  
UserName: John  
Password: Paul  
Email-ID: john789@gmail.com  
Authenticate: 3  
Name: TharaVarthan  
UserId: 4  
UserName: Thara  
Password: Varthan  
Email-ID: thara123@gmail.com  
Authenticate: 4  
Name: ShanMathi  
UserId: 5  
UserName: Shan  
Password: shanmathi555555  
Email-ID: shan456@gmail.com  
Authenticate: 5  
Name: HaveenaHavi  
UserId: 6  
UserName: Haveena  
Password: Havi  
Email-ID: havi789@gmail.com  
Authenticate: 6

**Sample Input and Output 2:**

Name: ZaraAli  
UserId: 1  
UserName: Zara  
Password: Ali  
Email-ID: zara123@gmail.com  
Authenticate: 1  
Name: DaisyDas  
UserId: 2  
UserName: Daisy  
Password: Das  
Email-ID: daisy456@gmail.com  
Authenticate: 2  
Name: JohnPaul  
UserId: 3  
UserName: John  
Password: Paul  
Email-ID: john789@gmail.com  
Authenticate: 3  
Name: TharaVarthan  
UserId: 4  
UserName: Thara  
Password: Varthan  
Email-ID: thara123@gmail.com  
Authenticate: 4  
Name: ShanMathi  
UserId: 5  
UserName: Shan  
Password: shanmathi555555  
Email-ID: shan456@gmail.com  
Authenticate: 5  
Name: HaveenaHavi  
UserId: 6  
UserName: Haveena  
Password: Havi  
Email-ID: havi789@gmail.com  
Authenticate: 6  
Enter the id and password to be updated  
**8  
james0755**  
No such id exists

**Delete Items**

**[Strictly adhere to the object-oriented programming specifications given in the problem statement. Template code is provided to ease the input-output process.]**  
**Use either Class.hbm.xml or annotations for mapping  
  
Important note -** Please start working on the template code which is provided. Template code will provide the base template or format in which the end solution is expected from you. Template code is not an executable code and will help you add code on top of it to ensure you make your submissions in the right format. If the submission is NOT as per the template code format the submissions will not be accepted.  
  
Mr.Ganesh wants to prepare an item list. Sometimes he may remove some items from the list because of the budget limit. Write a program for Mr.Ganesh to remove any item from the list and display the list in the ascending order of item id on the console.  
   
**Problem specification**  
Create a class **Item** with following private attributes.

|  |  |
| --- | --- |
| **Variable Name** | **Data type** |
| id | Integer |
| name | String |
| category | String |
| quantity | Integer |
| price | Float |

Include a default constructor, getters and setters for above class.  
   
Create class **ItemDAO** with the following methods.

|  |  |
| --- | --- |
| **Method name** | **Method Description** |
| public void delete(Item item) | This method is used to delete an existing item record in the database. Use **delete** method of **Session** class to delete the record. |
| public Item find(int id) | This method is used to retrieve item record by its id. Use **get** method of **Session** class to find the record. |
| public List<Item> list() | This method is used to retrieve all item records from the database. Use **list** method of **Criteria** class to retrieve the records. Return the item list in the ascending order of id. |

Create an XML mapping file named**item.hbm.xml** to map the persistence class - **Item.java** to the database table.  
  
Create a hibernate configuration XML file named **hibernate.cfg.xml** to configure database connection details and specify the **item.hbm.xml** file as a mapping resource. But, do not specify connection URL, username, and password properties.  
  
Create a class **HibernateUtil** to build an instance of**SessionFactory** and store it in static attribute **sessionFactory**.

|  |  |
| --- | --- |
| **Method name** | **Method Description** |
| public static SessionFactory getSessionFactory() | This method is used to return **sessionFactory** attribute. |

Use the code below to retrieve the connection details from **oracle.properties** to configure connection URL, username, and password properties into **hibernate.cfg.xml**.  
  
            ResourceBundle rb = ResourceBundle.getBundle("oracle");  
            Configuration cfg = new Configuration();  
            cfg.configure("hibernate.cfg.xml");  
            cfg.setProperty("hibernate.connection.url", rb.getString("db.url"));  
            cfg.setProperty("hibernate.connection.username", rb.getString("db.username"));  
            cfg.setProperty("hibernate.connection.password", rb.getString("db.password"));  
   
**oracle.properties**  
Use the following code to configure the connection.  
            
“db.url = jdbc:oracle:thin:@localhost:1521:xe  
db.username = root  
db.password = student”  
  
**Inorder to disable the log information in console,set the logger property -OFF in HibernateUtil class using the below code:**  
Logger log = Logger.getLogger("org.hibernate");  
log.setLevel(Level.OFF);  
System.setProperty("org.apache.commons.logging.Log",  
"org.apache.commons.logging.impl.NoOpLog");  
  
**Use the hibernate jar files from the below link. If you added any other version of hibernate jar files, your submission will not get accepted.**  
[e-box.co.in/uploads/SpringHibernateJars.zip](https://hcl.e-box.co.in/uploads/2696/Resource/JARS/springHibernateJar.zip)

**Download the oracle jar file in the below link.**  
[Oracle jar](https://hcl.e-box.co.in/uploads/File/ojdbc6.jar.zip)

**use the below link to download the template**  
[e-box.co.in/uploads/template.zip](https://hcl.e-box.co.in/uploads/File/templates/19390-temp.zip)

**Table Details:**  
create table item(id number(19) primary key,name varchar2(255),category varchar2(255),quantity number(19),price binary\_double);  
//sequence for sequential increment of id  
create sequence item\_seq start with 1 increment by 1;  
  
DBMS create scripts are available in the **queries.sql** file as part of code template. The table and column names are case sensitive.  
   
**Sample Output Format:**  
To display the item details on the console, use  
**System.out.printf("%-15s %-15s %-15s %-15s %s\n","Id","Name","Category","Number","Price");**

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

**Sample Input and Output 1:**

Menu  
1.Remove item  
2.Display items  
3.Exit  
Enter choice  
**2**  
Id              Name            Category        Quantity        Price  
1               chair           furniture       10              1000.0  
2               bat             sports          5               5000.0  
3               mouse           electronics     5               4000.0  
4               shoes           fashion         15              600.0  
Menu  
1.Remove item  
2.Display items  
3.Exit  
Enter choice  
**1**  
Enter item id  
**4**  
Item was Deleted  
Menu  
1.Remove item  
2.Display items  
3.Exit  
Enter choice  
**2**  
Id              Name            Category        Quantity        Price  
1               chair           furniture       10              1000.0  
2               bat             sports          5               5000.0  
3               mouse           electronics     5               4000.0  
Menu  
1.Remove item  
2.Display items  
3.Exit  
Enter choice  
**3**  
Bye

**Sample Input and Output 2:**

Menu  
1.Remove item  
2.Display items  
3.Exit  
Enter choice  
2  
Id              Name            Category        Quantity        Price  
1               chair           furniture       10              1000.0  
2               bat             sports          5               5000.0  
3               mouse           electronics     5               4000.0  
Menu  
1.Remove item  
2.Display items  
3.Exit  
Enter choice  
1  
Enter item id  
4  
Item not found  
Menu  
1.Remove item  
2.Display items  
3.Exit  
Enter choice  
3  
Bye

**PROBLEM**

ShopKit is a small e-commerce company in Hyderabad, India. As there are some of the products grouped together under packages, they are planning to design a spring-based console application for calculating the cost of each package. This task has been assigned to the Technical Manager, Miss Vidya.

Write the program to calculate the cost of each package and display the details by using Collection Injection with List.

**Problem Specifications**

Consider a class**Product** with the following private attributes.

|  |  |
| --- | --- |
| **Attribute** | **Data Type** |
| name | String |
| description | String |
| cost | Double |
| type | String |
| brand | String |

Include appropriate constructors, getters and setters.  
Include a toString() method that returns the details of the Product in the format specified.

Create a class **StandardPackage**with the following private attributes.

|  |  |
| --- | --- |
| **Attribute** | **Data Type** |
| name | String |
| status | String |
| cost | Double |
| productList | List<Product> |

Include appropriate constructors, getters and setters.

The **StandardPackage** class also contains the following methods.

|  |  |
| --- | --- |
| **Method** | **Description** |
| public void calculateCost() | This method calculates and sets the value of cost as the sum of costs of individual products in the package. |
| public void display() | This method displays the Package and Products in the Package. |

In the **applicationContext.xml** file, provide the Products into the Bean:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **id** | **name** | **description** | **cost** | **type** | **brand** |
| product1 | iPhone XR | Apple iPhone XR (64GB) Red | 44900 | Mobiles and Accessories | Apple |
| product2 | iPhone XR Back case | iPhone XR Transparent Back case | 699 | Mobiles and Accessories | Apple |
| product3 | Lenovo Laptop | Lenovo vintage Black | 45000 | Laptops | Lenovo |
| product4 | External Hard Drive | Expansion Portable 1.5TB HDD | 4099 | Data Storage | Seagate |
| product5 | Accessory Kit | Kit with a mouse, Joystick and headset | 7999 | Laptop Accessories | Sony |
| product6 | Skybag FlyAround | Navy Blue SkyBag-15Lts | 1500 | BackPacks | SkyBags |
| product7 | Men's Jacket | Coffee Brown Men's Jacket | 1499 | Winter wear | Max |
| product8 | Men's Sweatshirt | Men's Sweatshirt Black | 629 | Winter wear | Max |
| product9 | Liberty shoes | Boy's Lace Up Shoe | 599 | Shoes | Liberty |

All the Product Bean details are added to the xml file and given in the template code.

In the **applicationContext.xml** file, provide the StandardPackages into the Bean:

|  |  |  |  |
| --- | --- | --- | --- |
| **id** | **name** | **status** | **productList** |
| package1 | iPhone Pack | Available | List[product1, product2] |
| package2 | Laptop Set | Available | List[product3, product4, product5] |
| package3 | Style World | Available | List[product6, product7, product8, product9] |

Inject all products to productList of StandardPackage using List based Collection Injection.

In the main() create the instance of **ApplicationContext** class to access the bean objects from the applicationContext.xml file and call the method calculateCost() to find the total cost of StandardPackage and then display the product details by calling the display() method.

**Note:**  
For eclipse IDE users,  
Use the below link to download the SpringHibernate jars to configure with your local spring project.

[e-box.co.in/uploads/SpringHibernateJars.zip](https://hcl.e-box.co.in/uploads/2696/Resource/JARS/springHibernateJar.zip)

click the below link to download the template

[e-box.co.in/uploads/template.zip](https://hcl.e-box.co.in/uploads/File/templates/16005-temp.zip)

**Important note -** Please start working on the template code which is provided. The template code will provide the base template or format in which the end solution is expected from you. Template code is not an executable code and will help you add code on top of it to ensure you make your submissions in the right format. If the submission is NOT as per the template code format the submissions will not be accepted.

**Output Format:**

Display StandardPackage Details using the below format.  
System.out.format("%-25s %-25s %-15s \n","Name","Status","Cost");

Display Product Details using the below format.  
System.out.format("%-30s %-40s %-15s %-30s %-15s\n","Name","Description","Cost","Type","Brand");

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

Package Details:  
  
Package-1  
Name                      Status                    Cost              
iPhone Pack               Available                 45599.0           
Product Details:  
Name                           Description                              Cost            Type                           Brand            
iPhone XR                      Apple iPhone XR (64GB) Red               44900.00        Mobiles and Accessories        Apple            
iPhone XR Back case            iPhone XR Transparent Back case          699.00          Mobiles and Accessories        Apple            
  
Package-2  
Name                      Status                    Cost              
Laptop Set                Available                 57098.0           
Product Details:  
Name                           Description                              Cost            Type                           Brand            
Lenovo Laptop                  Lenovo vintage Black                     45000.00        Laptops                        Lenovo           
External Hard Drive            Expansion Portable 1.5TB HDD             4099.00         Data Storage                   Seagate          
Accessory Kit                  Kit with a mouse, Joystick and headset   7999.00         Laptop Accessories             Sony             
  
Package-3  
Name                      Status                    Cost              
Style World               Available                 4227.0            
Product Details:  
Name                           Description                              Cost            Type                           Brand            
Skybag FlyAround               Navy Blue SkyBag-15Lts                   1500.00         BackPacks                      SkyBags          
Men's Jacket                   Coffee Brown Men's Jacket                1499.00         Winter wear                    Max              
Men's Sweatshirt               Men's Sweatshirt Black                   629.00          Winter wear                    Max              
Liberty shoes                  Boy's Lace Up Shoe                       599.00          Shoes                          Liberty        

|  |
| --- |
| public class Product {  String name;  String description;  Double cost;  String type;  String brand;  public Product(String name, String description, Double cost, String type, String brand) {  super();  this.name = name;  this.description = description;  this.cost = cost;  this.type = type;  this.brand = brand;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public String getDescription() {  return description;  }  public void setDescription(String description) {  this.description = description;  }  public Double getCost() {  return cost;  }  public void setCost(Double cost) {  this.cost = cost;  }  public String getType() {  return type;  }  public void setType(String type) {  this.type = type;  }  public String getBrand() {  return brand;  }  public void setBrand(String brand) {  this.brand = brand;  }  public String toString() {  //Fill your code here  }    } |
| import java.util.List;  public class StandardPackage {  String name;  String status;  Double cost;  List<Product> productList;    public StandardPackage(String name, String status, List<Product> productList) {  super();  this.name = name;  this.status = status;  this.productList = productList;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public List<Product> getProductList() {  return productList;  }  public void setProductList(List<Product> productList) {  this.productList = productList;  }  public String getStatus() {  return status;  }  public void setStatus(String status) {  this.status = status;  }  public Double getCost() {  return cost;  }  public void setCost(Double cost) {  this.cost = cost;  }  public void calculateCost() {  //Fill your code here  }  public void display() {  //Fill your code here  }  } |
| <?xml version="1.0" encoding="UTF-8"?>  <beans  xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:p="http://www.springframework.org/schema/p"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">  <bean id="product1" class="Product">  <constructor-arg value="iPhone XR"></constructor-arg>  <constructor-arg value="Apple iPhone XR (64GB) Red"></constructor-arg>  <constructor-arg value="44900"></constructor-arg>  <constructor-arg value="Mobiles and Accessories"></constructor-arg>  <constructor-arg value="Apple"></constructor-arg>  </bean>  <bean id="product2" class="Product">  <constructor-arg value="iPhone XR Back case"></constructor-arg>  <constructor-arg value="iPhone XR Transparent Back case"></constructor-arg>  <constructor-arg value="699"></constructor-arg>  <constructor-arg value="Mobiles and Accessories"></constructor-arg>  <constructor-arg value="Apple"></constructor-arg>  </bean>  <bean id="product3" class="Product">    <constructor-arg value="Lenovo Laptop"></constructor-arg>  <constructor-arg value="Lenovo vintage Black"></constructor-arg>  <constructor-arg value="45000"></constructor-arg>  <constructor-arg value="Laptops"></constructor-arg>  <constructor-arg value="Lenovo"></constructor-arg>  </bean>    <bean id="product4" class="Product">  <constructor-arg value="External Hard Drive"></constructor-arg>  <constructor-arg value="Expansion Portable 1.5TB HDD"></constructor-arg>  <constructor-arg value="4099"></constructor-arg>  <constructor-arg value="Data Storage"></constructor-arg>  <constructor-arg value="Seagate"></constructor-arg>  </bean>  <bean id="product5" class="Product">  <constructor-arg value="Accessory Kit"></constructor-arg>  <constructor-arg value="Kit with a mouse, Joystick and headset"></constructor-arg>  <constructor-arg value="7999"></constructor-arg>  <constructor-arg value="Laptop Accessories"></constructor-arg>  <constructor-arg value="Sony"></constructor-arg>  </bean>  <bean id="product6" class="Product">  <constructor-arg value="Skybag FlyAround"></constructor-arg>  <constructor-arg value="Navy Blue SkyBag-15Lts"></constructor-arg>  <constructor-arg value="1500"></constructor-arg>  <constructor-arg value="BackPacks"></constructor-arg>  <constructor-arg value="SkyBags"></constructor-arg>  </bean>  <bean id="product7" class="Product">  <constructor-arg value="Men's Jacket"></constructor-arg>  <constructor-arg value="Coffee Brown Men's Jacket"></constructor-arg>  <constructor-arg value="1499"></constructor-arg>  <constructor-arg value="Winter wear"></constructor-arg>  <constructor-arg value="Max"></constructor-arg>  </bean>  <bean id="product8" class="Product">  <constructor-arg value="Men's Sweatshirt"></constructor-arg>  <constructor-arg value="Men's Sweatshirt Black"></constructor-arg>  <constructor-arg value="629"></constructor-arg>  <constructor-arg value="Winter wear"></constructor-arg>  <constructor-arg value="Max"></constructor-arg>  </bean>  <bean id="product9" class="Product">  <constructor-arg value="Liberty shoes"></constructor-arg>  <constructor-arg value="Boy's Lace Up Shoe"></constructor-arg>  <constructor-arg value="599"></constructor-arg>  <constructor-arg value="Shoes"></constructor-arg>  <constructor-arg value="Liberty"></constructor-arg>  </bean>  <bean id="product10" class="Product">  <constructor-arg value="The Theory of Everything"></constructor-arg>  <constructor-arg value="The Theory of Everything-Stephen Hawking"></constructor-arg>  <constructor-arg value="199"></constructor-arg>  <constructor-arg value="Biographies"></constructor-arg>  <constructor-arg value="PaperBack"></constructor-arg>  </bean>  <bean id="product11" class="Product">  <constructor-arg value="Three Thousand Stitches"></constructor-arg>  <constructor-arg value="Three Thousand Stitches-Sudha Murthy"></constructor-arg>  <constructor-arg value="190"></constructor-arg>  <constructor-arg value="Biographies"></constructor-arg>  <constructor-arg value="PaperBack"></constructor-arg>  </bean>  <bean id="product12" class="Product">  <constructor-arg value="Wings of Fire"></constructor-arg>  <constructor-arg value="Wings of Fire-A.P.J Abdul Kalam"></constructor-arg>  <constructor-arg value="258"></constructor-arg>  <constructor-arg value="Biographies"></constructor-arg>  <constructor-arg value="PaperBack"></constructor-arg>  </bean>  <!-- Fill your code here -->    <bean id="package1" class="StandardPackage">  <constructor-arg value="iPhone Pack"></constructor-arg>  <constructor-arg value="Available"></constructor-arg>  <constructor-arg>  <list>  <ref bean="product1"/>  <ref bean="product2"/>  </list>  </constructor-arg>  </bean>  <bean id="package2" class="StandardPackage">  <constructor-arg value="Laptop Set"></constructor-arg>  <constructor-arg value="Available"></constructor-arg>  <constructor-arg>  <list>  <ref bean="product3"/>  <ref bean="product4"/>  <ref bean="product5"/>  </list>  </constructor-arg>  </bean>  <bean id="package3" class="StandardPackage">  <constructor-arg value="Style World"></constructor-arg>  <constructor-arg value="Available"></constructor-arg>  <constructor-arg>  <list>  <ref bean="product6"/>  <ref bean="product7"/>  <ref bean="product8"/>  <ref bean="product9"/>  </list>  </constructor-arg>  </bean>  <bean id="package4" class="StandardPackage">  <constructor-arg value="Books"></constructor-arg>  <constructor-arg value="Fast sellers"></constructor-arg>  <constructor-arg>  <list>  <ref bean="product10"/>  <ref bean="product11"/>  <ref bean="product12"/>  </list>  </constructor-arg>  </bean>  </beans> |
| import java.util.logging.Level;  import java.util.logging.Logger;  public class Main {  public static void main(String[] args) {  Logger log = Logger.getLogger("org.hibernate");  log.setLevel(Level.OFF);  System.setProperty("org.apache.commons.logging.Log", "org.apache.commons.logging.impl.NoOpLog");  //Fill your code here    }  } |
|  |

**PROBLEM**

Cene Universe is an online troubleshooting company. Mr. Naveen is a troubleshooter working in the company. Since there are many movies running simultaneously on different screens of different theatres, they are planning to design a spring-based console application for displaying the screen details along with the details of the theatre that it belongs to.

Create a **Screen**class with the following attributes/variables.

|  |  |
| --- | --- |
| **Data Type** | **Variable** |
| String | name |
| String | status |
| Integer | capacity |
| Theatre | theatre |

Include appropriate getters and setters for the class.  
Include a toString() method that returns the details of the Screen in the format specified.

Create a **Theatre**class with the following attributes/variables.

|  |  |
| --- | --- |
| **Data Type** | **Variable** |
| String | name |
| String | location |
| String | city |

Include appropriate getters and setters for the class.  
Include a toString() method that returns the details of the Theatre in the format specified.

In the applicationContext.xml file, provide the Screens into the Bean:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **id** | **name** | **status** | **capacity** | **theatre** |
| screen1 | DRC Screen 1 | Available | 220 | theatre1 |
| screen2 | DRC Screen 2 | Available | 180 | theatre1 |
| screen3 | INOX Screen 1 | Unavailable | 240 | theatre2 |
| screen4 | INOX Screen 2 | Available | 240 | theatre2 |
| screen5 | INOX Screen 3 | Available | 240 | theatre2 |
| screen6 | PVR Screen 1 | Unavailable | 200 | theatre3 |
| screen7 | PVR Screen 2 | Available | 180 | theatre3 |
| screen8 | PVR Screen 3 | Unavailable | 220 | theatre3 |
| screen9 | PVR Screen 4 | Available | 220 | theatre3 |
| screen10 | PVR Screen 5 | Available | 240 | theatre3 |

All the Screen Bean details are added to the XML file and given in the template code.  
In the **applicationContext.xml** file, provide the Theatres into the Bean:

|  |  |  |  |
| --- | --- | --- | --- |
| **id** | **name** | **location** | **city** |
| theatre1 | DRC Cinemas | Vijayanagar | Mysore |
| theatre2 | INOX | Siddarth Nagar | Mysore |
| theatre3 | PVR | Mysore City | Mysore |

Inject all screens to Screen class and dependent Theatre using Constructor Dependency Injection.  
In the main() create the instance of **ApplicationContext** class to access the bean objects from the applicationContext.xml file display the screen details.

**Note:**  
For eclipse IDE users,  
Use the below link to download the SpringHibernate jars to configure with your local spring project.

[e-box.co.in/uploads/SpringHibernateJars.zip](https://hcl.e-box.co.in/uploads/2696/Resource/JARS/springHibernateJar.zip)

click the below link to download the template

[e-box.co.in/uploads/template.zip](https://app.e-box.co.in/uploads/hcl/springboot/16005-temp.zip)

**Important note -** Please start working on the template code which is provided. The template code will provide the base template or format in which the end solution is expected from you. Template code is not an executable code and will help you add code on top of it to ensure you make your submissions in the right format. If the submission is NOT as per the template code format the submissions will not be accepted.

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

**Output Format:**

Display Screen and Theatre Details using the below format.  
System.out.format("%-30s %-30s %-15s %-30s %-30s %s\n","Name","Status","Capacity","Theatre","Location","City");

**Sample Input and Output 1**

Screen Details:  
Name                           Status                         Capacity        Theatre                        Location                       City  
DRC Screen 1                   Available                      220             DRC Cinemas                    Vijayanagar                    Mysore  
DRC Screen 2                   Available                      180             DRC Cinemas                    Vijayanagar                    Mysore  
INOX Screen 1                  Unavailable                    240             INOX                           Siddarth Nagar                 Mysore  
INOX Screen 2                  Available                      240             INOX                           Siddarth Nagar                 Mysore  
INOX Screen 3                  Available                      240             INOX                           Siddarth Nagar                 Mysore  
PVR Screen 1                   Unavailable                    200             PVR                            Mysore City                    Mysore  
PVR Screen 2                   Available                      180             PVR                            Mysore City                    Mysore  
PVR Screen 3                   Unavailable                    220             PVR                            Mysore City                    Mysore  
PVR Screen 4                   Available                      220             PVR                            Mysore City                    Mysore  
PVR Screen 5                   Available                      240             PVR                            Mysore City                    Mysore

Top of Form

Bottom of Form

**Additional Sample TestCases**

**Sample Input and Output 1 :**

Screen Details:

Name Status Capacity Theatre Location City

DRC Screen 1 Available 220 DRC Cinemas Vijayanagar Mysore

DRC Screen 2 Available 180 DRC Cinemas Vijayanagar Mysore

INOX Screen 1 Unavailable 240 INOX Siddarth Nagar Mysore

INOX Screen 2 Available 240 INOX Siddarth Nagar Mysore

INOX Screen 3 Available 240 INOX Siddarth Nagar Mysore

PVR Screen 1 Unavailable 200 PVR Mysore City Mysore

PVR Screen 2 Available 180 PVR Mysore City Mysore

PVR Screen 3 Unavailable 220 PVR Mysore City Mysore

PVR Screen 4 Available 220 PVR Mysore City Mysore

PVR Screen 5 Available 240 PVR Mysore City Mysore

|  |
| --- |
| public class Screen {  String name;  String status;  Integer capacity;  Theatre theatre;    public Screen() {  super();  }  public Screen(String name, String status, Integer capacity, Theatre theatre) {  super();  this.name = name;  this.status = status;  this.capacity = capacity;  this.theatre = theatre;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public String getStatus() {  return status;  }  public void setStatus(String status) {  this.status = status;  }  public Integer getCapacity() {  return capacity;  }  public void setCapacity(Integer capacity) {  this.capacity = capacity;  }  public Theatre getTheatre() {  return theatre;  }  public void setTheatre(Theatre theatre) {  this.theatre = theatre;  }  //Fill your code here  } |
| public class Theatre {  String name;  String location;  String city;    public Theatre() {  super();  }  public Theatre(String name, String location, String city) {  super();  this.name = name;  this.location = location;  this.city = city;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public String getLocation() {  return location;  }  public void setLocation(String location) {  this.location = location;  }  public String getCity() {  return city;  }  public void setCity(String city) {  this.city = city;  }  //Fill your code here  } |
| import java.util.logging.Level;  import java.util.logging.Logger;  public class Main {  public static void main(String[] args) {  Logger log = Logger.getLogger("org.hibernate");  log.setLevel(Level.OFF);  System.setProperty("org.apache.commons.logging.Log", "org.apache.commons.logging.impl.NoOpLog");  //Fill your code here    }  } |
| <?xml version="1.0" encoding="UTF-8"?>  <beans  xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:p="http://www.springframework.org/schema/p"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">  <bean id="screen1" class="Screen">  <constructor-arg value="DRC Screen 1"></constructor-arg>  <constructor-arg value="Available"></constructor-arg>  <constructor-arg value="220"></constructor-arg>  <!-- Fill your code here -->  </bean>  <bean id="screen2" class="Screen">  <constructor-arg value="DRC Screen 2"></constructor-arg>  <constructor-arg value="Available"></constructor-arg>  <constructor-arg value="180"></constructor-arg>  <!-- Fill your code here -->  </bean>  <bean id="screen3" class="Screen">  <constructor-arg value="INOX Screen 1"></constructor-arg>  <constructor-arg value="Unavailable"></constructor-arg>  <constructor-arg value="240"></constructor-arg>  <!-- Fill your code here -->  </bean>    <bean id="screen4" class="Screen">  <constructor-arg value="INOX Screen 2"></constructor-arg>  <constructor-arg value="Available"></constructor-arg>  <constructor-arg value="240"></constructor-arg>  <!-- Fill your code here -->  </bean>  <bean id="screen5" class="Screen">  <constructor-arg value="INOX Screen 3"></constructor-arg>  <constructor-arg value="Available"></constructor-arg>  <constructor-arg value="240"></constructor-arg>  <!-- Fill your code here -->  </bean>  <bean id="screen6" class="Screen">  <constructor-arg value="PVR Screen 1"></constructor-arg>  <constructor-arg value="Unavailable"></constructor-arg>  <constructor-arg value="200"></constructor-arg>  <!-- Fill your code here -->  </bean>  <bean id="screen7" class="Screen">  <constructor-arg value="PVR Screen 2"></constructor-arg>  <constructor-arg value="Available"></constructor-arg>  <constructor-arg value="180"></constructor-arg>  <!-- Fill your code here -->  </bean>  <bean id="screen8" class="Screen">  <constructor-arg value="PVR Screen 3"></constructor-arg>  <constructor-arg value="Unavailable"></constructor-arg>  <constructor-arg value="220"></constructor-arg>  <!-- Fill your code here -->  </bean>  <bean id="screen9" class="Screen">  <constructor-arg value="PVR Screen 4"></constructor-arg>  <constructor-arg value="Available"></constructor-arg>  <constructor-arg value="220"></constructor-arg>  <!-- Fill your code here -->  </bean>  <bean id="screen10" class="Screen">  <constructor-arg value="PVR Screen 5"></constructor-arg>  <constructor-arg value="Available"></constructor-arg>  <constructor-arg value="240"></constructor-arg>  <!-- Fill your code here -->  </bean>  <bean id="theatre1" class="Theatre">  <constructor-arg value="DRC Cinemas"></constructor-arg>  <constructor-arg value="Vijayanagar"></constructor-arg>  <constructor-arg value="Mysore"></constructor-arg>  </bean>  <bean id="theatre2" class="Theatre">  <constructor-arg value="INOX"></constructor-arg>  <constructor-arg value="Siddarth Nagar"></constructor-arg>  <constructor-arg value="Mysore"></constructor-arg>  </bean>  <bean id="theatre3" class="Theatre">  <constructor-arg value="PVR"></constructor-arg>  <constructor-arg value="Mysore City"></constructor-arg>  <constructor-arg value="Mysore"></constructor-arg>  </bean>  </beans> |

**PROBLEM**

ShopKit is a small book store in Hyderabad, India. As there are some of the books grouped together under packages, they are planning to design a spring-based console application for calculating the cost of each package.

Write the application to calculate the cost of each package and display the details by using Collection Injection with List.

**Problem Specifications**

Consider the class**Book** with the following private attributes.

|  |  |
| --- | --- |
| **Attribute** | **Data Type** |
| name | String |
| description | String |
| cost | Double |
| category | String |
| author | String |

The methods for **getters**, **setters**and **constructors**are given in the template code.

Override a **toString()** method that returns the details of the Book in the format specified.

Consider a class **StandardPackage**with the following private attributes.

|  |  |
| --- | --- |
| **Attribute** | **Data Type** |
| name | String |
| status | String |
| cost | Double |
| bookList | List<Book> |

The methods for **getters**, **setters**and **constructors**are given in the template code.

In the **StandardPackage** class, define the following methods.

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| public void calculateCost() | This method calculates and sets the value of cost as the sum of costs of individual books in the package. |
| public void display() | This method displays the Package and Books in the Package. |

In the **applicationContext.xml** file, provide the Books into the Bean:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **id** | **name** | **description** | **cost** | **category** | **author** |
| book1 | War and Peace | Fiction novel based on war times romance | 432 | Fiction | Leo Tolstoy |
| book2 | Brave New World | Science Fiction Novel | 289 | SciFi | Aldous Huxley |
| book3 | Gone Girl | Suspense Thriller Novel | 805 | Thriller | Gillian Flynn |
| book4 | Murder on the Orient Express | Murder Mystery and Detective Novel | 988 | Murder Mystery | Agatha Christie |
| book5 | The Shining | Best selling psychological horror fiction novel | 1199 | Horror Thriller | Stephen King |
| book6 | The Notebook | Best selling romance fiction novel | 749 | Romance | Nicholas Sparks |
| book7 | The Wedding Date | Contemporary romance novel | 349 | Romcom | Jasmine Guillory |
| book8 | Bossypants | Humour autobiography novel | 357 | Comedy | Tina Fey |
| book9 | The Stupidest Angel | Humour Horror Fantasy Christmas Story | 285 | Horror Comedy | Christopher Moore |

All the Book Bean details are added to the XML file and given in the template code.

In the **applicationContext.xml** file, provide the StandardPackages into the Bean:

|  |  |  |  |
| --- | --- | --- | --- |
| **id** | **name** | **status** | **bookList** |
| package1 | Fiction Novel Pack | Available | List[book1, book2] |
| package2 | Mystery Thriller Pack | Available | List[book3, book4, book5] |
| package3 | Comedy Romance Pack | Available | List[book6, book7, book8, book9] |

Inject all books to bookList of StandardPackage using List based Collection Injection.

In the main() create the instance of **ApplicationContext**class to access the bean objects from the applicationContext.xml file and call the method calculateCost() to find the total cost of StandardPackage and then display the book details by calling the display() method.

**Note:**

For eclipse IDE users, use the below link to download the SpringHibernate jars to configure with your local spring project.  
[Spring Hibernate Jars](https://hcl.e-box.co.in/uploads/2696/Resource/JARS/springHibernateJar.zip)

**Download the template code in the below link**.  
[Code Template](https://hcl.e-box.co.in/problem/showTemplateCodeSet/19526?op=download&langid=6)

**Important note -** Please start working on the template code which is provided. The template code will provide the base template or format in which the end solution is expected from you. Template code is not an executable code and will help you add code on top of it to ensure you make your submissions in the right format. If the submission is NOT as per the template code format the submissions will not be accepted.

**Output Format:**

Display StandardPackage Details using the below format.  
System.out.format(**"%-25s %-25s %-15s \n"**,"Name","Status","Cost");

Display Product Details using the below format.  
System.out.format(**"%-30s %-40s %-15s %-30s %-15s\n"**,"Name","Description","Cost","Category","Author");

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

**Sample Input and Output:**

Package Details:  
  
Package-1  
Name                      Status                    Cost              
Fiction Novel Pack        Available                 721.0             
Product Details:  
Name                           Description                              Cost            Category                       Author           
War and Peace                  Fiction novel based on war times romance 432.00          Fiction                        Leo Tolstoy      
Brave New World                Science Fiction Novel                    289.00          SciFi                          Aldous Huxley    
  
Package-2  
Name                      Status                    Cost              
Mystery Thriller Pack     Available                 2992.0            
Product Details:  
Name                           Description                              Cost            Category                       Author           
Gone Girl                      Suspense Thriller Novel                  805.00          Thriller                       Gillian Flynn    
Murder on the Orient Express   Murder Mystery and Detective Novel       988.00          Murder Mystery                 Agatha Christie  
The Shining                    Best selling psychological horror fiction novel 1199.00         Horror Thriller                Stephen King     
  
Package-3  
Name                      Status                    Cost              
Comedy Romance Pack       Available                 1740.0            
Product Details:  
Name                           Description                              Cost            Category                       Author           
The Notebook                   Best selling romance fiction novel       749.00          Romance                        Nicholas Sparks  
The Wedding Date               Contemporary romance novel               349.00          Romcom                         Jasmine Guillory  
Bossypants                     Humour autobiography novel               357.00          Comedy                         Tina Fey         
The Stupidest Angel            Humour Horror Fantasy Christmas Story    285.00          Horror Comedy                  Christopher Moore

Top of Form

Bottom of Form

|  |
| --- |
| public class Book {  private String name;  private String description;  private Double cost;  private String category;  private String author;  public Book(String name, String description, Double cost, String category, String author) {  super();  this.name = name;  this.description = description;  this.cost = cost;  this.category = category;  this.author = author;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public String getDescription() {  return description;  }  public void setDescription(String description) {  this.description = description;  }  public Double getCost() {  return cost;  }  public void setCost(Double cost) {  this.cost = cost;  }  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;  }    //Fill your code here      } |
| import java.util.List;  public class StandardPackage {  private String name;  private String status;  private Double cost;  private List<Book> bookList;    public StandardPackage(String name, String status, List<Book> bookList) {  super();  this.name = name;  this.status = status;  this.bookList = bookList;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public List<Book> getBookList() {  return bookList;  }  public void setBookList(List<Book> bookList) {  this.bookList = bookList;  }  public String getStatus() {  return status;  }  public void setStatus(String status) {  this.status = status;  }  public Double getCost() {  return cost;  }  public void setCost(Double cost) {  this.cost = cost;  }    public void calculateCost() {  //Fill your code here  }    public void display() {  //Fill your code here  }  } |
| import java.util.logging.Level;  import java.util.logging.Logger;  public class Main {  public static void main(String[] args) {  Logger log = Logger.getLogger("org.hibernate");  log.setLevel(Level.OFF);  System.setProperty("org.apache.commons.logging.Log", "org.apache.commons.logging.impl.NoOpLog");  //Fill your code here    }  } |
| <?xml version="1.0" encoding="UTF-8"?>  <beans  xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:p="http://www.springframework.org/schema/p"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">  <bean id="book1" class="Book">  <constructor-arg value="War and Peace"></constructor-arg>  <constructor-arg value="Fiction novel based on war times romance"></constructor-arg>  <constructor-arg value="432"></constructor-arg>  <constructor-arg value="Fiction"></constructor-arg>  <constructor-arg value="Leo Tolstoy"></constructor-arg>  </bean>  <bean id="book2" class="Book">  <constructor-arg value="Brave New World"></constructor-arg>  <constructor-arg value="Science Fiction Novel"></constructor-arg>  <constructor-arg value="289"></constructor-arg>  <constructor-arg value="SciFi"></constructor-arg>  <constructor-arg value="Aldous Huxley"></constructor-arg>  </bean>  <bean id="book3" class="Book">    <constructor-arg value="Gone Girl"></constructor-arg>  <constructor-arg value="Suspense Thriller Novel"></constructor-arg>  <constructor-arg value="805"></constructor-arg>  <constructor-arg value="Thriller"></constructor-arg>  <constructor-arg value="Gillian Flynn"></constructor-arg>  </bean>    <bean id="book4" class="Book">  <constructor-arg value="Murder on the Orient Express"></constructor-arg>  <constructor-arg value="Murder Mystery and Detective Novel"></constructor-arg>  <constructor-arg value="988"></constructor-arg>  <constructor-arg value="Murder Mystery"></constructor-arg>  <constructor-arg value="Agatha Christie"></constructor-arg>  </bean>  <bean id="book5" class="Book">  <constructor-arg value="The Shining"></constructor-arg>  <constructor-arg value="Best selling psychological horror fiction novel"></constructor-arg>  <constructor-arg value="1199"></constructor-arg>  <constructor-arg value="Horror Thriller"></constructor-arg>  <constructor-arg value="Stephen King"></constructor-arg>  </bean>  <bean id="book6" class="Book">  <constructor-arg value="The Notebook"></constructor-arg>  <constructor-arg value="Best selling romance fiction novel"></constructor-arg>  <constructor-arg value="749"></constructor-arg>  <constructor-arg value="Romance"></constructor-arg>  <constructor-arg value="Nicholas Sparks"></constructor-arg>  </bean>  <bean id="book7" class="Book">  <constructor-arg value="The Wedding Date"></constructor-arg>  <constructor-arg value="Contemporary romance novel"></constructor-arg>  <constructor-arg value="349"></constructor-arg>  <constructor-arg value="Romcom"></constructor-arg>  <constructor-arg value="Jasmine Guillory"></constructor-arg>  </bean>  <bean id="book8" class="Book">  <constructor-arg value="Bossypants"></constructor-arg>  <constructor-arg value="Humour autobiography novel"></constructor-arg>  <constructor-arg value="357"></constructor-arg>  <constructor-arg value="Comedy"></constructor-arg>  <constructor-arg value="Tina Fey"></constructor-arg>  </bean>  <bean id="book9" class="Book">  <constructor-arg value="The Stupidest Angel"></constructor-arg>  <constructor-arg value="Humour Horror Fantasy Christmas Story"></constructor-arg>  <constructor-arg value="285"></constructor-arg>  <constructor-arg value="Horror Comedy"></constructor-arg>  <constructor-arg value="Christopher Moore"></constructor-arg>  </bean>  <bean id="package1" class="StandardPackage">  <constructor-arg value="Fiction Novel Pack"></constructor-arg>  <constructor-arg value="Available"></constructor-arg>  <constructor-arg>  <list>  <ref bean="book1"/>  <ref bean="book2"/>  </list>  </constructor-arg>  </bean>  <bean id="package2" class="StandardPackage">  <constructor-arg value="Mystery Thriller Pack"></constructor-arg>  <constructor-arg value="Available"></constructor-arg>  <constructor-arg>  <list>  <ref bean="book3"/>  <ref bean="book4"/>  <ref bean="book5"/>  </list>  </constructor-arg>  </bean>  <bean id="package3" class="StandardPackage">  <constructor-arg value="Comedy Romance Pack"></constructor-arg>  <constructor-arg value="Available"></constructor-arg>  <constructor-arg>  <list>  <ref bean="book6"/>  <ref bean="book7"/>  <ref bean="book8"/>  <ref bean="book9"/>  </list>  </constructor-arg>  </bean>  </beans> |

**PROBLEM**

ShopKit is a small e-commerce company in Hyderabad, India. As there are some of the products grouped together under packages, they are planning to design a spring-based console application for displaying the product details along with the package name that it belongs to. This task has been assigned to the Technical Manager, Miss Vidya.

Write the program to display the product details with the package name by using Constructor Dependency Injection.

**Problem Specifications**

Create a class**Product** with the following private attributes.

|  |  |
| --- | --- |
| **Attribute** | **Data Type** |
| name | String |
| description | String |
| cost | Double |
| type | String |
| brand | String |
| package | StandardPackage |

Include appropriate constructors, getters, and setters.  
Include a toString() method that returns the details of the Product in the format specified.

Create a class **StandardPackage**with the following private attributes.

|  |  |
| --- | --- |
| **Attribute** | **Data Type** |
| name | String |
| status | String |

Include appropriate constructors, getters, and setters.  
Include a toString() method that returns the details of the Product in the format specified.

In the **applicationContext.xml** file, provide the Products into the Bean:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **id** | **name** | **description** | **cost** | **type** | **brand** | **package** |
| product1 | iPhone XR | Apple iPhone XR (64GB) Red | 44900 | Mobiles and Accessories | Apple | package1 |
| product2 | iPhone XR Back case | iPhone XR Transparent Back case | 699 | Mobiles and Accessories | Apple | package1 |
| product3 | Lenovo Laptop | Lenovo vintage Black | 45000 | Laptops | Lenovo | package2 |
| product4 | External Hard Drive | Expansion Portable 1.5TB HDD | 4099 | Data Storage | Seagate | package2 |
| product5 | Accessory Kit | Kit with a mouse, Joystick and headset | 7999 | Laptop Accessories | Sony | package2 |
| product6 | Skybag FlyAround | Navy Blue SkyBag-15Lts | 1500 | BackPacks | SkyBags | package3 |
| product7 | Men's Jacket | Coffee Brown Men's Jacket | 1499 | Winter wear | Max | package3 |
| product8 | Men's Sweatshirt | Men's Sweatshirt Black | 629 | Winter wear | Max | package3 |
| product9 | Liberty shoes | Boy's Lace Up Shoe | 599 | Shoes | Liberty | package3 |

All the Product Bean details are added to the XML file and given in the template code.

In the **applicationContext.xml** file, provide the StandardPackages into the Bean:

|  |  |  |
| --- | --- | --- |
| **id** | **name** | **status** |
| package1 | iPhone Pack | Available |
| package2 | Laptop Set | Available |
| package3 | Style World | Available |

Inject all products to Product class and dependent StandardPackage using Constructor Dependency Injection.

In the main() create the instance of **ApplicationContext** class to access the bean objects from the applicationContext.xml file display the product details.

**Note:**  
For eclipse IDE users,  
Use the below link to download the SpringHibernate jars to configure with your local spring project.

[e-box.co.in/uploads/SpringHibernateJars.zip](https://hcl.e-box.co.in/uploads/2696/Resource/JARS/springHibernateJar.zip)

click the below link to download the template

[e-box.co.in/uploads/template.zip](https://app.e-box.co.in/uploads/hcl/springboot/16005-temp.zip)

**Important note -** Please start working on the template code which is provided. The template code will provide the base template or format in which the end solution is expected from you. Template code is not an executable code and will help you add code on top of it to ensure you make your submissions in the right format. If the submission is NOT as per the template code format the submissions will not be accepted.

**Output Format:**

Display Product Details using the below format.  
System.out.format("%-30s %-40s %-15s %-30s %-15s %s\n","Name","Description","Cost","Type","Brand",”Package”);

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

Product Details:  
Name                           Description                              Cost            Type                           Brand           Package  
iPhone XR                      Apple iPhone XR (64GB) Red               44900.00        Mobiles and Accessories        Apple           iPhone Pack  
iPhone XR Back case            iPhone XR Transparent Back case          699.00          Mobiles and Accessories        Apple           iPhone Pack  
Lenovo Laptop                  Lenovo vintage Black                     45000.00        Laptops                        Lenovo          Laptop Set  
External Hard Drive            Expansion Portable 1.5TB HDD             4099.00         Data Storage                   Seagate         Laptop Set  
Accessory Kit                  Kit with a mouse, Joystick and headset   7999.00         Laptop Accessories             Sony            Laptop Set  
Skybag FlyAround               Navy Blue SkyBag-15Lts                   1500.00         BackPacks                      SkyBags         Style World  
Men's Jacket                   Coffee Brown Men's Jacket                1499.00         Winter wear                    Max             Style World  
Men's Sweatshirt               Men's Sweatshirt Black                   629.00          Winter wear                    Max             Style World  
Liberty shoes                  Boy's Lace Up Shoe                       599.00          Shoes                          Liberty         Style World

Top of Form

Bottom of Form

**Additional Sample TestCases**

**Sample Input and Output 1 :**

Product Details:

Name Description Cost Type Brand Package

iPhone XR Apple iPhone XR (64GB) Red 44900.00 Mobiles and Accessories Apple iPhone Pack

iPhone XR Back case iPhone XR Transparent Back case 699.00 Mobiles and Accessories Apple iPhone Pack

Lenovo Laptop Lenovo vintage Black 45000.00 Laptops Lenovo Laptop Set

External Hard Drive Expansion Portable 1.5TB HDD 4099.00 Data Storage Seagate Laptop Set

Accessory Kit Kit with a mouse, Joystick and headset 7999.00 Laptop Accessories Sony Laptop Set

Skybag FlyAround Navy Blue SkyBag-15Lts 1500.00 BackPacks SkyBags Style World

Men's Jacket Coffee Brown Men's Jacket 1499.00 Winter wear Max Style World

Men's Sweatshirt Men's Sweatshirt Black 629.00 Winter wear Max Style World

Liberty shoes Boy's Lace Up Shoe 599.00 Shoes Liberty Style World

|  |
| --- |
| public class Product {  String name;  String description;  Double cost;  String type;  String brand;  StandardPackage stdPackage;  public Product(String name, String description, Double cost, String type, String brand, StandardPackage stdPackage) {  super();  this.name = name;  this.description = description;  this.cost = cost;  this.type = type;  this.brand = brand;  this.stdPackage = stdPackage;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public String getDescription() {  return description;  }  public void setDescription(String description) {  this.description = description;  }  public Double getCost() {  return cost;  }  public void setCost(Double cost) {  this.cost = cost;  }  public String getType() {  return type;  }  public void setType(String type) {  this.type = type;  }  public String getBrand() {  return brand;  }  public void setBrand(String brand) {  this.brand = brand;  }  public StandardPackage getStdPackage() {  return stdPackage;  }  public void setStdPackage(StandardPackage stdPackage) {  this.stdPackage = stdPackage;  }    //Fill your code here    } |
| import java.util.List;  public class StandardPackage {  String name;  String status;    public StandardPackage(String name, String status) {  super();  this.name = name;  this.status = status;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public String getStatus() {  return status;  }  public void setStatus(String status) {  this.status = status;  }    //Fill your code here  } |
| <?xml version="1.0" encoding="UTF-8"?>  <beans  xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:p="http://www.springframework.org/schema/p"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">  <bean id="product1" class="Product">  <constructor-arg value="iPhone XR"></constructor-arg>  <constructor-arg value="Apple iPhone XR (64GB) Red"></constructor-arg>  <constructor-arg value="44900"></constructor-arg>  <constructor-arg value="Mobiles and Accessories"></constructor-arg>  <constructor-arg value="Apple"></constructor-arg>  <!-- Fill your code here -->  </bean>  <bean id="product2" class="Product">  <constructor-arg value="iPhone XR Back case"></constructor-arg>  <constructor-arg value="iPhone XR Transparent Back case"></constructor-arg>  <constructor-arg value="699"></constructor-arg>  <constructor-arg value="Mobiles and Accessories"></constructor-arg>  <constructor-arg value="Apple"></constructor-arg>  <!-- Fill your code here -->  </bean>  <bean id="product3" class="Product">  <constructor-arg value="Lenovo Laptop"></constructor-arg>  <constructor-arg value="Lenovo vintage Black"></constructor-arg>  <constructor-arg value="45000"></constructor-arg>  <constructor-arg value="Laptops"></constructor-arg>  <constructor-arg value="Lenovo"></constructor-arg>  <!-- Fill your code here -->  </bean>  <bean id="product4" class="Product">  <constructor-arg value="External Hard Drive"></constructor-arg>  <constructor-arg value="Expansion Portable 1.5TB HDD"></constructor-arg>  <constructor-arg value="4099"></constructor-arg>  <constructor-arg value="Data Storage"></constructor-arg>  <constructor-arg value="Seagate"></constructor-arg>  <!-- Fill your code here -->  </bean>  <bean id="product5" class="Product">  <constructor-arg value="Accessory Kit"></constructor-arg>  <constructor-arg value="Kit with a mouse, Joystick and headset"></constructor-arg>  <constructor-arg value="7999"></constructor-arg>  <constructor-arg value="Laptop Accessories"></constructor-arg>  <constructor-arg value="Sony"></constructor-arg>  <!-- Fill your code here -->  </bean>  <bean id="product6" class="Product">  <constructor-arg value="Skybag FlyAround"></constructor-arg>  <constructor-arg value="Navy Blue SkyBag-15Lts"></constructor-arg>  <constructor-arg value="1500"></constructor-arg>  <constructor-arg value="BackPacks"></constructor-arg>  <constructor-arg value="SkyBags"></constructor-arg>  <!-- Fill your code here -->  </bean>  <bean id="product7" class="Product">  <constructor-arg value="Men's Jacket"></constructor-arg>  <constructor-arg value="Coffee Brown Men's Jacket"></constructor-arg>  <constructor-arg value="1499"></constructor-arg>  <constructor-arg value="Winter wear"></constructor-arg>  <constructor-arg value="Max"></constructor-arg>  <!-- Fill your code here -->  </bean>  <bean id="product8" class="Product">  <constructor-arg value="Men's Sweatshirt"></constructor-arg>  <constructor-arg value="Men's Sweatshirt Black"></constructor-arg>  <constructor-arg value="629"></constructor-arg>  <constructor-arg value="Winter wear"></constructor-arg>  <constructor-arg value="Max"></constructor-arg>  <!-- Fill your code here -->  </bean>  <bean id="product9" class="Product">  <constructor-arg value="Liberty shoes"></constructor-arg>  <constructor-arg value="Boy's Lace Up Shoe"></constructor-arg>  <constructor-arg value="599"></constructor-arg>  <constructor-arg value="Shoes"></constructor-arg>  <constructor-arg value="Liberty"></constructor-arg>  <!-- Fill your code here -->  </bean>    <bean id="package1" class="StandardPackage">  <constructor-arg value="iPhone Pack"></constructor-arg>  <constructor-arg value="Available"></constructor-arg>  </bean>  <bean id="package2" class="StandardPackage">  <constructor-arg value="Laptop Set"></constructor-arg>  <constructor-arg value="Available"></constructor-arg>  </bean>  <bean id="package3" class="StandardPackage">  <constructor-arg value="Style World"></constructor-arg>  <constructor-arg value="Available"></constructor-arg>  </bean>  </beans> |
| import java.util.logging.Level;  import java.util.logging.Logger;  public class Main {  public static void main(String[] args) {  Logger log = Logger.getLogger("org.hibernate");  log.setLevel(Level.OFF);  System.setProperty("org.apache.commons.logging.Log", "org.apache.commons.logging.impl.NoOpLog");  //Fill your code here    }  } |

click the below link to download the template  
[http://app.e-box.co.in/uploads/template.zip.](https://app.e-box.co.in/uploads/hcl/springboot/16011-temp.zip)

**Important note** - Please start working on the template code which is provided. The template code will provide the base template or format in which the end solution is expected from you. Template code is not an executable code and will help you add code on top of it to ensure you make your submissions in the right format. If the submission is NOT as per the template code format the submissions will not be accepted.

**All text in bold corresponds to the input and the rest corresponds to output.**  
  
**Sample Input and Output 1**

Enter the status to be searched  
**available**

Theatre Name    Location        City            Screen Name           
DRC Cinemas     Vijayanagar     Mysore          DRC Screen 1          
DRC Cinemas     Vijayanagar     Mysore          DRC Screen 2          
INOX            Siddarth Nagar  Mysore          INOX Screen 2         
INOX            Siddarth Nagar  Mysore          INOX Screen 2         
PVR             Mysore City     Mysore          PVR Screen 2          
PVR             Mysore City     Mysore          PVR Screen 4          
PVR             Mysore City     Mysore          PVR Screen 5          
Cenepolis       Mysore City     Mysore          Cenepolis Screen 4    
Cenepolis       Mysore City     Mysore          Cenepolis Screen 5

**Sample Input and Output 2**  
Enter the status to be searched  
**not available**  
No Theatres available with the specified status

|  |
| --- |
| package com.spring;  public class Theatre {  Integer id;  String name;  String location;  String city;    public Theatre() {  super();  }  public Theatre(Integer id, String name, String location, String city) {  super();  this.id = id;  this.name = name;  this.location = location;  this.city = city;  }  public Integer getId() {  return id;  }  public void setId(Integer id) {  this.id = id;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public String getLocation() {  return location;  }  public void setLocation(String location) {  this.location = location;  }  public String getCity() {  return city;  }  public void setCity(String city) {  this.city = city;  }    } |
| package com.spring;  public class Screen {  Integer id;  String name;  Theatre theatre;  String status;  Integer capacity;    public Screen() {  super();  }  public Screen(Integer id, String name, Theatre theatre, String status, Integer capacity) {  super();  this.id = id;  this.name = name;  this.theatre = theatre;  this.status = status;  this.capacity = capacity;  }  public Integer getId() {  return id;  }  public void setId(Integer id) {  this.id = id;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public Theatre getTheatre() {  return theatre;  }  public void setTheatre(Theatre theatre) {  this.theatre = theatre;  }  public String getStatus() {  return status;  }  public void setStatus(String status) {  this.status = status;  }  public Integer getCapacity() {  return capacity;  }  public void setCapacity(Integer capacity) {  this.capacity = capacity;  }  @Override  public String toString() {  //Fill your code here  }  } |
| package com.spring;  import java.util.ArrayList;  import java.util.List;  //Fill your code here  public class TheatreDAO {    public List<Screen> getTheatreDetails(){  List<Screen> s=new ArrayList<Screen>();  Theatre t1=new Theatre(1,"DRC Cinemas","Vijayanagar","Mysore");  s.add(new Screen(1, "DRC Screen 1", t1,"Available",220));  s.add(new Screen(2, "DRC Screen 2",t1, "Available",180));  Theatre t2=new Theatre(2,"INOX","Siddarth Nagar","Mysore");  s.add(new Screen(3, "INOX Screen 1", t2, "Unavailable",240));  s.add(new Screen(4, "INOX Screen 2", t2, "Available",240));  s.add(new Screen(5, "INOX Screen 2",t2, "Available",240));  Theatre t3=new Theatre(3,"PVR","Mysore City","Mysore");  s.add(new Screen(6, "PVR Screen 1",t3,"Unavailable",180));  s.add(new Screen(7, "PVR Screen 2",t3,"Available",200));  s.add(new Screen(8, "PVR Screen 3",t3,"Unavailable",200));  s.add(new Screen(9, "PVR Screen 4",t3,"Available",240));  s.add(new Screen(10, "PVR Screen 5",t3,"Available",180));  Theatre t4=new Theatre(4,"Cenepolis","Mysore City","Mysore");  s.add(new Screen(11, "Cenepolis Screen 1",t4,"Unavailable",200));  s.add(new Screen(12, "Cenepolis Screen 2",t4,"Unavailable",180));  s.add(new Screen(13, "Cenepolis Screen 3",t4,"Unavailable",220));  s.add(new Screen(14, "Cenepolis Screen 4",t4,"Available",220));  s.add(new Screen(15, "Cenepolis Screen 5",t4,"Available",240));  return s;  }  public List<Screen> searchTheatreByStatus(String status){  //Fill your code here  }  } |
| import com.spring.\*;  import java.util.logging.Level;  import java.util.logging.Logger;  public class Main {  public static void main(String[] args) {  Logger log = Logger.getLogger("org.hibernate");  log.setLevel(Level.OFF);  System.setProperty("org.apache.commons.logging.Log", "org.apache.commons.logging.impl.NoOpLog");  //Fill your code here  }  } |

SpringJDBC

**Leave Management**

Write the spring application using jdbc template to manage the leave details stored in the employee database.

**[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.]**

Consider a class named **LeaveDetail**with the following private attributes/variables.0

|  |  |
| --- | --- |
| **Data type** | **Variable** |
| Integer | employeeId |
| String | name |
| String | department |
| Integer | leavesTaken |
| Integer | availableLeaves |

The appropriate **getters, setters, default**and **parameterized constructors**for the above class are given in the template code

Consider a class **LeaveDetailDAO**class with the following attributes/variables

|  |  |
| --- | --- |
| **Data Type** | **Attributes** |
| DataSource | dataSource |
| JdbcTemplate | jdbcTemplate |

The **LeaveDetailDAO**class has the following methods

|  |  |
| --- | --- |
| **Method** | **Description** |
| public List<LeaveDetail> getAllLeaveDetails() | This method returns all the LeaveDetail objects available in the database based on the ascending order of employeeId. |
| public boolean createLeaveDetail(LeaveDetail leaveDetail) | This method adds new leaveDetail record into the database using the given leaveDetail object if the employeeId of the leaveDetail is not available and returns true. If the employeeId of the given leaveDetail object is already available in the database, it returns false. |

Consider an **applicationContext.xml**  file should contains the following bean,

**bean id = “dataSource”, class = “org.springframework.jdbc.datasource.DriverManagerDataSource”  
properties :**  
driverClassName = oracle.jdbc.OracleDriver  
${db.url} = url  
${db.username} = username  
${db.password} = password  
  
**bean class="org.springframework.jdbc.core.JdbcTemplate" id="jdbcTemplate"  
properties :**  
name="dataSource" ref="dataSource  
  
**bean id="leaveDetailDAO" class = "com.spring.dao.LeaveDetailDAO**  
**properties :**  
name = "dataSource" ref = "dataSource  
  
**Use,**  
<context:annotation-config/>  
<context:component-scan base-package="com.spring"/>  
<context:property-placeholder location="classpath:oracle.properties"/>

**Table properties :**

create table leave\_detail(  
employee\_id number primary key not null,  
name varchar2(255) not null,  
department varchar2(255) not null,  
leaves\_taken number not null,  
available\_leaves number not null);

**Note :**

* In the main method , the leaveDetailDAO bean is obtained, the details of all leaves are displayed, leave details to add into the database are read and the list after insertion is displayed.
* Use **System.out.printf("%-15s %-15s %-15s %-15s %s\n", "Employee Id", "Name", "Department", "Leaves Taken In Days", "Available Leaves In Days");** to print the output.

**Download the spring jars from the below link:**  
[Spring-Hibernate jars](https://hcl.e-box.co.in/uploads/SpringHibernateJars.zip)

**Download the template code from the below link:**  
[Code template](https://hcl.e-box.co.in/problem/showTemplateCodeSet/27134?op=download&langid=6)

**Download the oracle jar file in the below link.**  
[Oracle jar](https://hcl.e-box.co.in/uploads/File/ojdbc6.jar.zip)

**Input and Output Format**     

Refer sample input and output for formatting specifications.       
**All text in bold corresponds to the input and the rest corresponds to output.**

**Sample Input and Output 1 :** 

Before Insertion :  
Employee Id     Name            Department      Leaves Taken In Days Available Leaves In Days  
1010            Michael         Administration  3               17  
1015            Brett           Development     2               18  
1123            Aliya           Operations      9               15  
Enter the leave details to add in CSV format(employeeId,name,department,leavesTaken,availableLeaves) :  
**1019,Smith,Content,10,12**  
After Insertion :  
Employee Id     Name            Department      Leaves Taken In Days Available Leaves In Days  
1010            Michael         Administration  3               17  
1015            Brett           Development     2               18  
1019            Smith           Content         10              12  
1123            Aliya           Operations      9               15  
  
**Sample Input and Output 2 :**   
  
Before Insertion :  
Employee Id     Name            Department      Leaves Taken In Days Available Leaves In Days  
1010            Michael         Administration  3               17  
1015            Brett           Development     2               18  
1019            Smith           Content         10              12  
1123            Aliya           Operations      9               15  
Enter the leave details to add in CSV format(employeeId,name,department,leavesTaken,availableLeaves) :  
**1019,Harry,Content,10,12**  
Employee id already exists

Top of Form

Bottom of Form

|  |
| --- |
| begin  execute immediate 'drop table leave\_detail';  exception  when others then null;  end;  /  create table leave\_detail(  employee\_id number primary key not null,  name varchar2(255) not null,  department varchar2(255) not null,  leaves\_taken number not null,  available\_leaves number not null  );  insert into leave\_detail(employee\_id,name,department,leaves\_taken,available\_leaves)values(1015,'Brett','Development',2,18);  insert into leave\_detail(employee\_id,name,department,leaves\_taken,available\_leaves)values(1123,'Aliya','Operations',9,15);  insert into leave\_detail(employee\_id,name,department,leaves\_taken,available\_leaves)values(1010,'Michael','Administration',3,17); |
| db.url = jdbc:oracle:thin:@localhost:1521:xe  db.username = root  db.password = student |
| package com.spring.entity;  public class LeaveDetail {  private Integer employeeId;  private String name;  private String department;  private Integer leavesTaken;  private Integer availableLeaves;  public LeaveDetail() {  super();  // TODO Auto-generated constructor stub  }  public LeaveDetail(Integer employeeId, String name, String department, Integer leavesTaken,  Integer availableLeaves) {  super();  this.employeeId = employeeId;  this.name = name;  this.department = department;  this.leavesTaken = leavesTaken;  this.availableLeaves = availableLeaves;  }  public Integer getEmployeeId() {  return employeeId;  }  public void setEmployeeId(Integer employeeId) {  this.employeeId = employeeId;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public String getDepartment() {  return department;  }  public void setDepartment(String department) {  this.department = department;  }  public Integer getLeavesTaken() {  return leavesTaken;  }  public void setLeavesTaken(Integer leavesTaken) {  this.leavesTaken = leavesTaken;  }  public Integer getAvailableLeaves() {  return availableLeaves;  }  public void setAvailableLeaves(Integer availableLeaves) {  this.availableLeaves = availableLeaves;  }    } |
| package com.spring.dao;  import com.spring.entity.\*;  import java.sql.ResultSet;  import java.sql.Types;  import java.sql.SQLException;  import java.util.List;  import javax.sql.DataSource;  import org.springframework.dao.DuplicateKeyException;  import org.springframework.dao.EmptyResultDataAccessException;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.jdbc.core.JdbcTemplate;  import org.springframework.jdbc.core.RowMapper;  import org.springframework.stereotype.Component;  @Component("leaveDetailDAO")  public class LeaveDetailDAO {    private DataSource datasource;  @Autowired  private JdbcTemplate jdbcTemplate;    public DataSource getDataSource() {  return datasource;  }  public void setDataSource(DataSource dataSource) {  this.jdbcTemplate= new JdbcTemplate(dataSource);  }    public List<LeaveDetail> getAllLeaveDetails() {  //fill your code  return null;  }  public boolean createLeaveDetail(LeaveDetail leaveDetail) {  //fill your code  return false;  }    } |
| import com.spring.\*;  import java.util.logging.Level;  import java.util.logging.Logger;  import org.springframework.context.ApplicationContext;  import org.springframework.context.support.ClassPathXmlApplicationContext;  public class Main {  public static void main(String[] args) {  Logger log = Logger.getLogger("org.hibernate");  log.setLevel(Level.OFF);  System.setProperty("org.apache.commons.logging.Log", "org.apache.commons.logging.impl.NoOpLog");    }  } |
| <beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:context="http://www.springframework.org/schema/context"  xsi:schemaLocation="  http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans-3.1.xsd  http://www.springframework.org/schema/context  http://www.springframework.org/schema/context/spring-context-3.1.xsd">  <!-- fill your code -->  </beans> |

**Training Management**

Write the spring application using jdbc template to manage the training details stored in the database.

**[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.]**

Consider a class named **Training**with the following private attributes/variables.

|  |  |
| --- | --- |
| **Data type** | **Variable** |
| String | trainingName |
| Date | startDate |
| Date | endDate |
| String | trainerName |
| String | venue |

The appropriate **getters, setters, default**and **parameterized constructors**for the above class are given in the template code

Consider a class **TrainingDAO**class with the following attributes/variables

|  |  |
| --- | --- |
| **Data Type** | **Attributes** |
| DataSource | dataSource |
| JdbcTemplate | jdbcTemplate |

The **TrainingDAO**class has the following methods

|  |  |
| --- | --- |
| **Method** | **Description** |
| public List<Training> getAllTrainings() | This method returns all the training objects available in the database based on the ascending order of trainingName. |
| public boolean deleteTraining(String trainingName) | This method deletes the training record from the database for the given trainingName if the name of the training is available and returns true. If the trainingName of the given training object is not available in the database, it returns false. |

Consider an **applicationContext.xml**  file should contains the following bean,

**bean id = “dataSource”, class = “org.springframework.jdbc.datasource.DriverManagerDataSource”  
properties :**  
driverClassName = oracle.jdbc.OracleDriver  
${db.url} = url  
${db.username} = username  
${db.password} = password  
  
**bean class="org.springframework.jdbc.core.JdbcTemplate" id="jdbcTemplate"  
properties :**  
name="dataSource" ref="dataSource  
  
**bean id="trainingDAO" class = "com.spring.dao.TrainingDAO  
properties :**  
name = "dataSource" ref = "dataSource  
  
**Use,**  
<context:annotation-config/>  
<context:component-scan base-package="com.spring"/>  
<context:property-placeholder location="classpath:oracle.properties"/>

**Table properties :**

create table training(  
training\_name varchar2(255) not null,  
start\_date date not null,  
end\_date date not null,  
trainer\_name varchar2(255) not null,  
venue varchar2(255) not null  
);

**Note :**

* In the main method , the trainingDAO bean is obtained, the details of all trainings are displayed, training name to delete from the database are read and the list after deletion is displayed.
* Use **System.out.printf("%-15s %-15s %-15s %-15s %s\n", "Training Name", "Start Date", "End Date", "Trainer Name", "Venue");** to print the output.
* The date format used should be **“dd-MM-yyyy”.**

**Download the spring jars from the below link:**  
[Spring-Hibernate jars](https://hcl.e-box.co.in/uploads/SpringHibernateJars.zip)

**Download the template code from the below link:**  
[Code template](https://hcl.e-box.co.in/problem/showTemplateCodeSet/27127?op=download&langid=6)

**Download the oracle jar file in the below link.**  
[Oracle jar](https://hcl.e-box.co.in/uploads/File/ojdbc6.jar.zip)

**Input and Output Format**     

Refer sample input and output for formatting specifications.       
**All text in bold corresponds to the input and the rest corresponds to output.**

**Sample Input and Output 1 :** 

Before Deletion :  
Training Name   Start Date      End Date        Trainer Name    Venue  
Android         06-03-2015      13-12-2015      Brett           Melbourne  
C-Sharp         03-08-2017      02-01-2018      Ariana          Los Angeles  
Full-Stack      16-01-2011      30-03-2012      Robert          London  
JavaScript      15-10-2017      15-12-2017      Michael         Texas  
Python          10-02-2015      20-05-2015      Aliya           Sydney  
Enter the training name to delete from database :  
**Python**  
After Deletion :  
Training Name   Start Date      End Date        Trainer Name    Venue  
Android         06-03-2015      13-12-2015      Brett           Melbourne  
C-Sharp         03-08-2017      02-01-2018      Ariana          Los Angeles  
Full-Stack      16-01-2011      30-03-2012      Robert          London  
JavaScript      15-10-2017      15-12-2017      Michael         Texas  
  
**Sample Input and Output 2 :**   
  
Before Deletion :  
Training Name   Start Date      End Date        Trainer Name    Venue  
Android         06-03-2015      13-12-2015      Brett           Melbourne  
C-Sharp         03-08-2017      02-01-2018      Ariana          Los Angeles  
Full-Stack      16-01-2011      30-03-2012      Robert          London  
JavaScript      15-10-2017      15-12-2017      Michael         Texas  
Enter the training name to delete from database :  
**fullstack**  
No such training exists

Top of Form

Bottom of Form

|  |
| --- |
| begin  execute immediate 'drop table training';  exception  when others then null;  end;  /  create table training(  training\_name varchar2(255) not null,  start\_date date not null,  end\_date date not null,  trainer\_name varchar2(255) not null,  venue varchar2(255) not null  );  insert into training(training\_name,start\_date,end\_date,trainer\_name,venue)values('Android',TO\_DATE('2015-03-06','yyyy-mm-dd'),TO\_DATE('2015-12-13','yyyy-mm-dd'),'Brett','Melbourne');  insert into training(training\_name,start\_date,end\_date,trainer\_name,venue)values('Python',TO\_DATE('2015-02-10','yyyy-mm-dd'),TO\_DATE('2015-05-20','yyyy-mm-dd'),'Aliya','Sydney');  insert into training(training\_name,start\_date,end\_date,trainer\_name,venue)values('C-Sharp',TO\_DATE('2017-08-03','yyyy-mm-dd'),TO\_DATE('2018-01-02','yyyy-mm-dd'),'Ariana','Los Angeles');  insert into training(training\_name,start\_date,end\_date,trainer\_name,venue)values('Full-Stack',TO\_DATE('2011-01-16','yyyy-mm-dd'),TO\_DATE('2012-03-30','yyyy-mm-dd'),'Robert','London');  insert into training(training\_name,start\_date,end\_date,trainer\_name,venue)values('JavaScript',TO\_DATE('2017-10-15','yyyy-mm-dd'),TO\_DATE('2017-12-15','yyyy-mm-dd'),'Michael','Texas'); |
| package com.spring.entity;  import java.util.Date;  public class Training {  private String trainingName;  private Date startDate;  private Date endDate;  private String trainerName;  private String venue;  public Training() {  super();  // TODO Auto-generated constructor stub  }  public Training(String trainingName, Date startDate, Date endDate, String trainerName, String venue) {  super();  this.trainingName = trainingName;  this.startDate = startDate;  this.endDate = endDate;  this.trainerName = trainerName;  this.venue = venue;  }  public String getTrainingName() {  return trainingName;  }  public void setTrainingName(String trainingName) {  this.trainingName = trainingName;  }  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;  }  public String getTrainerName() {  return trainerName;  }  public void setTrainerName(String trainerName) {  this.trainerName = trainerName;  }  public String getVenue() {  return venue;  }  public void setVenue(String venue) {  this.venue = venue;  }    } |
| package com.spring.dao;  import com.spring.entity.\*;  import java.sql.ResultSet;  import java.sql.Types;  import java.sql.SQLException;  import java.util.List;  import javax.sql.DataSource;  import org.springframework.dao.DuplicateKeyException;  import org.springframework.dao.EmptyResultDataAccessException;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.jdbc.core.JdbcTemplate;  import org.springframework.jdbc.core.RowMapper;  import org.springframework.stereotype.Component;  @Component("trainingDAO")  public class TrainingDAO {    private DataSource datasource;  @Autowired  private JdbcTemplate jdbcTemplate;    public DataSource getDataSource() {  return datasource;  }  public void setDataSource(DataSource dataSource) {  this.jdbcTemplate= new JdbcTemplate(dataSource);  }    public List<Training> getAllTrainings() {  //fill your code  return null;  }  public boolean deleteTraining(String trainingName) {  //fill your code  return false;  }    } |
| import com.spring.\*;  import java.util.logging.Level;  import java.util.logging.Logger;  import org.springframework.context.ApplicationContext;  import org.springframework.context.support.ClassPathXmlApplicationContext;  public class Main {  public static void main(String[] args) {  Logger log = Logger.getLogger("org.hibernate");  log.setLevel(Level.OFF);  System.setProperty("org.apache.commons.logging.Log", "org.apache.commons.logging.impl.NoOpLog");    }  } |
| <beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:context="http://www.springframework.org/schema/context"  xsi:schemaLocation="  http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans-3.1.xsd  http://www.springframework.org/schema/context  http://www.springframework.org/schema/context/spring-context-3.1.xsd">  <!-- fill your code -->  </beans> |

**Policy Management**

Write the spring application using jdbc template to manage the policy details stored in the database.

**[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.]**

Consider a class named **Policy**with the following private attributes/variables.

|  |  |
| --- | --- |
| **Data type** | **Variable** |
| String | policyId |
| String | customer |
| Double | policyValue |
| String | policyPlan |
| String | status |
| Double | coveragePercentage |

The appropriate **getters, setters, default**and **parameterized constructors**for the above class are given in the template code

Consider a class **PolicyDAO**class with the following attributes/variables

|  |  |
| --- | --- |
| **Data Type** | **Attributes** |
| DataSource | dataSource |
| JdbcTemplate | jdbcTemplate |

The **PolicyDAO**class has the following methods

|  |  |
| --- | --- |
| **Method** | **Description** |
| public List<Policy> getAllPolicies() | This method returns all the Policy objects available in the database based on the ascending order of policyId. |
| public boolean deletePolicy(String policyId) | This method deletes the policy data from the database for the given policyId if it is available and returns true. If the given policyId is not available, then it returns false. |

Consider an **applicationContext.xml**  file should contains the following bean,

**bean id = “dataSource”, class = “org.springframework.jdbc.datasource.DriverManagerDataSource”  
properties :**  
driverClassName = oracle.jdbc.OracleDriver  
${db.url} = url  
${db.username} = username  
${db.password} = password  
  
**bean class="org.springframework.jdbc.core.JdbcTemplate" id="jdbcTemplate"  
properties :**  
name="dataSource" ref="dataSource  
  
**bean id="policyDAO" class = "com.spring.dao.PolicyDAO  
properties :**  
name = "dataSource" ref = "dataSource  
  
**Use,**  
<context:annotation-config/>  
<context:component-scan base-package="com.spring"/>  
<context:property-placeholder location="classpath:oracle.properties"/>

**Table properties :**

create table policy(  
    policy\_id varchar2(30),  
    customer varchar2(40),  
    policy\_value binary\_double,  
    policy\_plan varchar2(40),  
    status varchar2(20),  
    coverage\_percentage binary\_double);

**Note :**

* In the main method , the policyDAO bean is obtained, the details of all policies are displayed, policyId to delete the particular policy from the database is read and the list after deletion is displayed.
* Use **System.out.printf("%-10s %-15s %-15s %-15s %-15s %s\n", "Policy Id", "Customer", "Policy Value", "Policy Plan", "Status", "Coverage Percentage");** to print the output.
* All double values should be formatted upto 2 decimal places.
* If the given policyId is not available then print “**Invalid policy id**”

**Download the spring jars from the below link:**  
[Spring-Hibernate jars](https://hcl.e-box.co.in/uploads/SpringHibernateJars.zip)

**Download the template code from the below link:**  
[Code template](https://hcl.e-box.co.in/problem/showTemplateCodeSet/27124?op=download&langid=6)

**Download the oracle jar file in the below link.**  
[Oracle jar](https://hcl.e-box.co.in/uploads/File/ojdbc6.jar.zip)

**Input and Output Format**     

Refer sample input and output for formatting specifications.       
**All text in bold corresponds to the input and the rest corresponds to output.**

**Sample Input and Output 1 :**

Available Policies :  
Policy Id  Customer        Policy Value    Policy Plan     Status          Coverage Percentage  
PCICAAND6  Andrew          50000.00        Vehicle         Closed          72.00  
PCICABEN7  Robin           100000.00       Apartment       Closed          75.60  
PCICABEN8  Benita          100000.00       Villa           Closed          73.00  
PCICAJOH1  John            80000.00        Home            Active          80.00  
PCICANAN3  Jack            45000.00        Health          Active          92.00  
PCICANAN4  Nancy           60000.00        Health          Active          93.40  
PCICASAR5  Sara            70000.00        Vehicle         Active          74.50  
Enter the policy id to delete :  
**PCICASAR5**  
Policy deleted successfully  
Available Policies :  
PCICAAND6  Andrew          50000.00        Vehicle         Closed          72.00  
PCICABEN7  Robin           100000.00       Apartment       Closed          75.60  
PCICABEN8  Benita          100000.00       Villa           Closed          73.00  
PCICAJOH1  John            80000.00        Home            Active          80.00  
PCICANAN3  Jack            45000.00        Health          Active          92.00  
PCICANAN4  Nancy           60000.00        Health          Active          93.40

**Sample Input and Output 2 :**

Available Policies :  
Policy Id  Customer        Policy Value    Policy Plan     Status          Coverage Percentage  
PCICAAND6  Andrew          50000.00        Vehicle         Closed          72.00  
PCICABEN7  Robin           100000.00       Apartment       Closed          75.60  
PCICABEN8  Benita          100000.00       Villa           Closed          73.00  
PCICAJOH1  John            80000.00        Home            Active          80.00  
PCICANAN3  Jack            45000.00        Health          Active          92.00  
PCICANAN4  Nancy           60000.00        Health          Active          93.40  
Enter the policy id to delete :  
**PCICASAT5**  
Invalid policy id

Top of Form

Bottom of Form

**Delete User**

Harry is a famous business man in the city. He wants to delete the unwanted owners from the list as he buys their shares in the company. So he decides to develop the spring application to delete the owners.

Write the spring application using jdbc template to display the owner list and to delete the owner for that particular id.

**[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.]**

Consider an **Owner** Class with the following attributes/variables

|  |  |
| --- | --- |
| **Data Type** | **Attributes** |
| String | name |
| String | password |
| String | emailId |
| String | mobileNumber |

Include appropriate getters and setters.

Consider an **OwnerDAO** class with the following attributes/variables

|  |  |
| --- | --- |
| **Data Type** | **Attributes** |
| DataSource | dataSource |
| JdbcTemplate | jdbcTemplate |

The **ownerDAO** class has the following methods

|  |  |
| --- | --- |
| **Method** | **Description** |
| public List<Owners> getAllOwners() | This method returns all the Owner objects. |
| public String getOwnerById(int id) | This method returns the owner name for the given id. |
| public void deleteOwner(int id) | This method is used to delete the owner from the table using the given id. |

Consider an **applicationContext.xml**  file should contains the following bean,

**bean id = “dataSource”, class = “org.springframework.jdbc.datasource.DriverManagerDataSource”  
properties :**  
driverClassName = oracle.jdbc.OracleDriver  
${db.url} = url  
${db.username} = username  
${db.password} = password  
  
**bean class="org.springframework.jdbc.core.JdbcTemplate" id="jdbcTemplate"  
properties :**  
name="dataSource" ref="dataSource  
  
**bean id="ownerDAO" class = "com.spring.dao.OwnerDAO  
properties :**  
name = "dataSource" ref = "dataSource  
  
**Use,**  
<context:annotation-config/>  
<context:component-scan base-package="com.spring"/>  
<context:property-placeholder location="classpath:oracle.properties"/>

**Table properties :**  
create table owner(id number(19) primary key,name varchar2(255),password varchar2(255),email\_id varchar2(255),mobile\_number varchar2(255));

**Download the jars from the below link**  
[Spring jars](https://hcl.e-box.co.in/uploads/SpringHibernateJars.zip)

**Download the code template from the below link**  
[Code Template](http://hcl.e-box.co.in/problem/showTemplateCodeSet/27081?op=download&langid=6)

**Download the oracle jar file in the below link.**  
[Oracle jar](https://hcl.e-box.co.in/uploads/File/ojdbc6.jar.zip)

**HINT :**

In the main method , the OwnerDAO bean is obtained and the details of all owners are displayed. Then the id of the particular user is entered and getOwnerById() method is called on it for getting the owner name. If the owner is present, then deleteOwner() method is called with the id to delete the owner. If no owner found for that given id then print **“No such owner!”.**

Use **System.out.printf("%-15s %-15s %-15s %s\n","Id","Name","Email","Mobile");** to print the output.

**Input and Output Format**       
Refer sample input and output for formatting specifications.       
**All text in bold corresponds to the input and the rest corresponds to output.**       
**Sample Input and Output  1:**

... Owner List ...  
Id              Name            Email          Mobile  
1               Robert          robert@gmail.com 7894561230  
2               William         william@gmail.com 7894567890  
3               Jessy           jessy@gmail.com 968520147  
4               Peter           peter@gmail.com 8523697410  
5               Stuart          stuart@gmail.com 9874563201  
Enter the id of the person you want to delete  
1  
owner deleted successfully  
... Updated owner List ...  
Id              Name            Email          Mobile  
2               William         william@gmail.com 7894567890  
3               Jessy           jessy@gmail.com 968520147  
4               Peter           peter@gmail.com 8523697410  
5               Stuart          stuart@gmail.com 9874563201

**Sample Input and Output  2:**

... Owner List ...  
Id              Name            Email          Mobile  
2               William         william@gmail.com 7894567890  
3               Jessy           jessy@gmail.com 968520147  
4               Peter           peter@gmail.com 8523697410  
5               Stuart          stuart@gmail.com 9874563201  
Enter the id of the person you want to delete  
**9**  
No such owner!

**Library Management**

Write the spring application using jdbc template to manage the book details stored in the library database.

**[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.]**

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

|  |  |
| --- | --- |
| **Data type** | **Variable** |
| Integer | id |
| String | title |
| String | category |
| String | author |
| Double | price |

The appropriate **getters, setters, default**and **parameterized constructors**for the above class are given in the template code

Consider a class **BookDAO**class with the following attributes/variables

|  |  |
| --- | --- |
| **Data Type** | **Attributes** |
| DataSource | dataSource |
| JdbcTemplate | jdbcTemplate |

The **BookDAO**class has the following methods

|  |  |
| --- | --- |
| **Method** | **Description** |
| public List<Book> getAllBooks() | This method returns all the Book objects available in the database based on the ascending order of title. |
| public boolean createBook(Book book) | This method adds new book record into the database using the given book object if the id of the book is not available and returns true. If the id of the given book object is already available in the database, it returns false. |

Consider an **applicationContext.xml**  file should contains the following bean,

**bean id = “dataSource”, class = “org.springframework.jdbc.datasource.DriverManagerDataSource”  
properties :**  
driverClassName = oracle.jdbc.OracleDriver  
${db.url} = url  
${db.username} = username  
${db.password} = password  
  
**bean class="org.springframework.jdbc.core.JdbcTemplate" id="jdbcTemplate"  
properties :**  
name="dataSource" ref="dataSource  
  
**bean id="bookDAO" class = "com.spring.dao.BookDAO  
properties :**  
name = "dataSource" ref = "dataSource  
  
**Use,**  
<context:annotation-config/>  
<context:component-scan base-package="com.spring"/>  
<context:property-placeholder location="classpath:oracle.properties"/>

**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));

**Note :**

* In the main method , the bookDAO bean is obtained, the details of all books are displayed, book details to add into the database are read and the list after insertion is displayed.
* Use **System.out.printf("%-5s %-15s %-15s %-15s %s\n", "Id", "Title", "Category", "Author", "Price");** to print the output.
* All double values should be formatted upto 2 decimal places.

**Download the spring jars from the below link:**  
[Spring-Hibernate jars](https://hcl.e-box.co.in/uploads/SpringHibernateJars.zip)

**Download the template code from the below link:**  
[Code template](https://hcl.e-box.co.in/problem/showTemplateCodeSet/27126?op=download&langid=6)

**Download the oracle jar file in the below link.**  
[Oracle jar](https://hcl.e-box.co.in/uploads/File/ojdbc6.jar.zip)

**Input and Output Format**     

Refer sample input and output for formatting specifications.       
**All text in bold corresponds to the input and the rest corresponds to output.**

**Sample Input and Output 1 :** 

Before Insertion :  
Id    Title           Category        Author          Price  
1     Shout           Poetry          Laurie          500.00  
2     Testaments      Fiction         Margaret        1000.00  
Enter the book details to add in CSV format(id,title,category,author,price) :  
**3,Twilight,Fantasy,JK Rowling,850**  
After Insertion :  
1     Shout           Poetry          Laurie          500.00  
2     Testaments      Fiction         Margaret        1000.00  
3     Twilight        Fantasy         JK Rowling      850.00  
  
**Sample Input and Output 2 :**   
  
Before Insertion :  
Id    Title           Category        Author          Price  
1     Shout           Poetry          Laurie          500.00  
2     Testaments      Fiction         Margaret        1000.00  
3     Twilight        Fantasy         JK Rowling      850.00  
Enter the book details to add in CSV format(id,title,category,author,price) :  
**3,Twilight,Fantasy,JK Rowling,850**  
Book id already exists

Top of Form

Bottom of Form

**Create and Categorize Books**

Write a Spring Boot application to create a book object and categorize the books based on the publisher.

**Important note** **-**Please start working on the template code which is provided. The template code will provide the base template or format in which the end solution is expected from you. Template code is not an executable code and will help you add code on top of it to ensure you make your submissions in the right format. If the submission is NOT as per the template code format the submissions will not be accepted.

**Problem Specifications**

Create a class **Book** with the following private attributes.

|  |  |
| --- | --- |
| **Attribute Name** | **Data Type** |
| id | Long |
| name | String |
| subject | String |
| author | String |
| publisher | String |

Generate appropriate getters, setters, default, and parameterized constructors.  
  
**Controller Specification**

Create a class for the controller named **BookController** with the following methods.

|  |  |  |
| --- | --- | --- |
| **Method Name** | **URL(Request Mapping)** | **Method description** |
| public Book create(Book book) | /book/create | Map this method to the URL ‘**/book/create**’ using @PostMapping  annotation. This method will be called, when the URL ‘**/book/create**’  was hit on postman with the JSON  parameters (id, name, subject, author, publisher). This method  calls the saveBook method of BookDAO and returns the value which is  returned by the saveBook method of bookDAO class. |
| public Map<String, List<Book>> categorizeBooks() | /book/categorize | Map this method to the URL ‘**/book/categorize**’ using @GetMapping  annotation. This method will be called, when the URL ‘**/book/categorize**’  was hit on browser / postman. This method calls the  categorizeBooks method of BookDAO to obtain the categorized book  details and returns the categorized book map. |

Use @**RestController**annotation to make the above class act as a RESTful controller.  
Create a class App that acts as the Main class. Create the main method for the application in this class. Use **@SpringBootApplication**annotation in this class.

**DAO Specification**

Create a class **BookDAO**with the below method.

|  |  |
| --- | --- |
| **Method Name** | **Method Description** |
| public Book saveBook(Book book) | This method gets an object of the type Book as an argument and adds the book object to the  book list and returns the same book object. |
| public Map<String, List<Book>> categorizeBooks() | This method uses the static book list in BookDAO and categorizes the book based on publisher.  In a HashMap, store book publisher as key and list of books as value and return the categorized  book map. |

Use the static book list which is provided in the template to store and categorize books.

Use @**Component** annotation for the above class.

**XML Configuration**  
The pom.xml file is available as a part of the code template, all dependencies for solving this problem are already added in this file.

**Screenshot 1**

**URL:**Base URL+”/book/categorize”  
**Method**: GET

**Screenshot 2**

**URL:**Base URL+”/book/create”  
**Method**: POST

|  |
| --- |
| package com.springboot.domain;  public class Book {  private Long id;  private String name;  private String subject;  private String publisher;  private String author;    public Book() {    }  public Book(Long id, String name, String subject, String publisher, String author) {  this.id = id;  this.name = name;  this.subject = subject;  this.publisher = publisher;  this.author = author;  }  public Long getId() {  return id;  }  public void setId(Long id) {  this.id = id;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public String getSubject() {  return subject;  }  public void setSubject(String subject) {  this.subject = subject;  }  public String getPublisher() {  return publisher;  }  public void getPublisher(String publisher) {  this.publisher = publisher;  }  public String getAuthor() {  return author;  }  public void setAuthor(String author) {  this.author = author;  }  } |
| package com.springboot.dao;  import java.util.ArrayList;  import java.util.HashMap;  import java.util.List;  import java.util.Map;  import org.springframework.stereotype.Component;  import com.springboot.domain.Book;  //Fill your code here  public class BookDAO {    public static List<Book> bookList = new ArrayList<Book>();  static {  bookList.add(new Book(1L, "The Silent Patient","Psychological thriller","Celadon Books", "Alex Michaelides"));  bookList.add(new Book(2L, "The Plot","Domestic Fiction","Celadon Books", "Jean Hanff Korelitz"));  bookList.add(new Book(3L, "In Cold Blood","Non-fiction","Random House", "Truman Capote"));  bookList.add(new Book(4L, "The Road to Reality","LAWS OF THE UNIVERSE","Random House", "ROGER PENROSE"));  bookList.add(new Book(5L, "Big Little Lies","Non-fiction","Penguin Publishing","Liane Moriarty"));  }  public Map<String, List<Book>> categorizeBooks() {  //Fill your code here  }    public Book saveBook(Book book) {  //Fill your code here  }  } |
| package com.springboot;  import org.springframework.boot.SpringApplication;  import org.springframework.boot.autoconfigure.SpringBootApplication;  @SpringBootApplication  public class App {  public static void main(String[] args) {  //Fill your code here  }  } |
| package com.springboot.controller;  import java.util.List;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.web.bind.annotation.GetMapping;  import org.springframework.web.bind.annotation.PostMapping;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.RequestBody;  import org.springframework.web.bind.annotation.RestController;  import com.springboot.domain.Book;  import com.springboot.dao.BookDAO;  import java.util.Map;  //Fill your code here  public class BookController {  //Fill your code here  public Book create(@RequestBody Book book)  {  //Fill your code here  }  //Fill your code here  public Map<String, List<Book>> categorizeBooks()  {  //Fill your code here  }  } |

**List and Display Event Details**

**List and Display Event Details**

Create a Spring Boot application to list all the event details and get a particular event detail as JSON using rest controller.  
  
**Important note** **-**Please start working on the template code which is provided. The template code will provide the base template or format in which the end solution is expected from you. Template code is not an executable code and will help you add code on top of it to ensure you make your submissions in the right format. If the submission is NOT as per the template code format the submissions will not be accepted.

**Problem Specifications**  
  
Create a class **Event** with the following private attributes.

|  |  |
| --- | --- |
| **Attribute Name** | **Data Type** |
| id | Long |
| name | String |
| type | String |
| hall | String |
| ticketPrice | Double |

Generate appropriate getters, setters, and parameterized constructors.

**Controller Specification**

Create a class for controller named **EventController** with the following methods.  
Map this class to the URL ‘**/event**’ using @RequestMapping annotation.

|  |  |  |
| --- | --- | --- |
| **Method Name** | **URL(Request Mapping)** | **Method description** |
| List<Event> getEvents() | /event/list | Map this method to the URL ‘**/list**’ using @GetMapping annotation. This method will be called, When the URL ‘**/event/list**’ was hit on browser / postman.This method calls the list method of EventDAO class to get the list of event objects and returns the list of event objects. |
| public Event showEvent() | /event/show/{id} | Map this method to the URL ‘**/show/{id}**’ using @GetMapping annotation. Get the event id from the URL using @PathVariable annotation. This method will be called, When the URL ‘**/event/show/{id}**’ was hit on browser / postman. This method calls the list method of EventDAO class to get the list of events and find the event object with the id provided to this method and returns the event object for the given id. |

Use @**RestController**annotation to make the above class act as a RESTful controller.  
Create a class App which acts as a Main class. Create the main method for the application in this class. Use **@SpringBootApplication**annotation in this class.  
  
**DAO Specification**

Create a class **EventDAO**with the below method.

|  |  |
| --- | --- |
| **Method Name** | **Method Description** |
| List<Event> list() | This method will return a list of pre-populated event objects. It is already available as a part of the code template. |

Use @**Component** annotation for the above class.

**XML Configuration**  
The pom.xml file is available as a part of code template, all dependencies for solving this problem are already added in this file.

**Event Details**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Id** | **Name** | **Type** | **Hall** | **Ticket Price** |
| 1 | La Centrale Food Festival | Exhibition | Rein Fire Ranch | 750.0 |
| 2 | Grand Central Market | Exhibition | The Heritage Room | 1500.0 |
| 3 | Great Northern Food Festival | Exhibition | Reflections | 200.0 |
| 4 | Fareground Festival | Stage show | Pier Sixty | 250.0 |

**Sample Screenshot 1:**  
  
**URL:**Base URL+”/event/list”  
**Method**: GET

**Sample Screenshot 2:**  
  
**URL:**Base URL+”/event/show/3”  
**Method**: GET

|  |
| --- |
| package com.springboot.domain;  public class Event {    private Long id;  private String name;  private String type;  private String hall;  private Double ticketPrice;      public Event() {  super();  }      public Event(Long id, String name, String type, String hall, Double ticketPrice) {  super();  this.id = id;  this.name = name;  this.type = type;  this.hall = hall;  this.ticketPrice = ticketPrice;  }  public Long getId() {  return id;  }  public void setId(Long id) {  this.id = id;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public String getType() {  return type;  }  public void setType(String type) {  this.type = type;  }  public String getHall() {  return hall;  }  public void setHall(String hall) {  this.hall = hall;  }  public Double getTicketPrice() {  return ticketPrice;  }  public void setTicketPrice(Double ticketPrice) {  this.ticketPrice = ticketPrice;  }      } |
| package com.springboot.controller;  import java.util.ArrayList;  import java.util.List;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.web.bind.annotation.GetMapping;  import org.springframework.web.bind.annotation.PathVariable;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.RestController;  import com.springboot.dao.EventDAO;  import com.springboot.domain.Event;  //fill your code here  public class EventController {    //fill your code here  public List<Event> getEvents() {    //fill your code here  }    public Event showEvent(@PathVariable Long id) {  //fill your code here    }    } |
| package com.springboot.dao;  import java.util.ArrayList;  import java.util.List;  import org.springframework.stereotype.Component;  import com.springboot.domain.Event;  //fill your code here  public class EventDAO {  public List<Event> list() {  ArrayList<Event> eventList = new ArrayList<Event>();  eventList.add(new Event(1L,"La Centrale Food Festival","Exhibition","Rein Fire Ranch",750.0));  eventList.add(new Event(2L,"Grand Central Market","Exhibition","The Heritage Room",1500.0));  eventList.add(new Event(3L,"Great Northern Food Festival","Exhibition","Reflections",200.0));  eventList.add(new Event(4L,"Fareground Festival","Stage show","Pier Sixty",250.0));  return eventList;  }  } |

**Team List using @PostMapping**

Create a Spring Boot Application to create the team and list all the teams.

**Important note** **-**Please start working on the template code which is provided. The template code will provide the base template or format in which the end solution is expected from you. Template code is not an executable code and will help you add code on top of it to ensure you make your submissions in the right format. If the submission is NOT as per the template code format the submissions will not be accepted.

Create a class **Team** with the following private attributes.

|  |  |
| --- | --- |
| **Attribute Name** | **Data Type** |
| id | Long |
| name | String |
| captain | String |
| Size | Integer |
| Color | String |

Generate appropriate getters, setters, and parameterized constructors for the above class.

**Controller Specification**

Create a class for a controller named **TeamController** with the following methods

|  |  |  |
| --- | --- | --- |
| **Method Name** | **URL** | **Method Description** |
| List<Team> list() | /list | Map this method to the URL ‘**/list**’ using @GetMapping annotation. This method will be called when the URL ‘**/team/list**’ was hit on browser/postman.This method calls the list method of TeamDAO and returns the list of team values which is returned by the TeamDAO as a JSON response. |
| Boolean create(@RequestBody Team team) | /create | Map this method to the URL ‘**/create**’ using @PostMapping annotation. This method will be called when the URL ‘**/team/create**’ was hit on browser/postman. This method gets an object of type Team as an argument through the request body and calls the save method of TeamDAO and returns the team object which is returned by the save method of TeamDAO class as a JSON response. |

Use @**RestController**annotation to make the above class act as a RESTful controller and use **@RequestMapping**(“/team”) for the above controller.  
Create a class App that acts as the Main class. Create the main method for the application in this class. Use **@SpringBootApplication**annotation in this class.

**DAO Specification**

Create a class **TeamDAO**with the below methods

|  |  |
| --- | --- |
| **Method Name** | **Method Description** |
| List<Team> list() | This method is used to return the populated list of team values. The populated team list is already available as a part of  the template code. |
| Boolean save(Team team) | This method gets an object of the type Team as an argument and adds the team object to the team list and returns the boolean value. |

Use @**Component** annotation for the above class.

**XML Configuration**  
The pom.xml file is available as a part of the code template, all dependencies for solving this problem are already added in this file.

**Screenshot 1 – Listing Teams**

**URL:**Base URL+”/team/list”  
**Method:** GET

**Screenshot 2 – Create a Team**

**URL:**Base URL+”/team/create”  
**Method:** POST

|  |
| --- |
| package com.springboot.domain;  public class Team {    private Long id;  private String name;  private String captain;  private Integer size;  private String color;  public Team(){  }  public Team(Long id, String name, String captain,  Integer size, String color) {  super();  this.id = id;  this.name = name;  this.captain = captain;  this.size = size;  this.color = color;  }    public Long getId() {  return id;  }  public void setId(Long id) {  this.id = id;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public String getCaptain() {  return captain;  }  public void setCaptain(String captain) {  this.captain = captain;  }  public Integer getSize() {  return size;  }  public void setSize(Integer size) {  this.size = size;  }  public String getColor() {  return color;  }  public void setColor(String color) {  this.color = color;  }    } |
| package com.springboot.dao;  import java.util.\*;  import org.springframework.boot.autoconfigure.SpringBootApplication;  import org.springframework.stereotype.Component;  import com.springboot.domain.Team;  @Component  public class TeamDAO {  static ArrayList<Team> teamList = new ArrayList<Team>();  static {  teamList.add(new Team(1L,"Los Angeles Lakers","Frank Vogel",11,"Yellow"));  teamList.add(new Team(2L,"Golden State Warriors","Steve Kerr",13,"Blue"));  teamList.add(new Team(3L,"Chicago Bulls","Billy Donovan",12,"Red"));  teamList.add(new Team(4L,"Brooklyn Nets","Steve Nash",13,"Black"));  }    public Boolean save(Team team) {  //Fill your code here  }  public List<Team> list() {  //Fill your code here  }    } |
| package com.springboot.controller;  //Fill your code here  public class TeamController {  //Fill your code here  } |
| package com.springboot;  @SpringBootApplication  public class App {  public static void main(String[] args) {  //Fill your code here  }  } |