```
package com.cts.library.bo;


import java.util.Map;
import java.util.Set;
import com.cts.library.exception.InvalidBookNameException;
import com.cts.library.exception.InvalidStudentIdException;
import com.cts.library.model.BookInfo;
import com.cts.library.model.StudentInfo;


public class LibraryBo {
	private StudentInfo stInfo;
	private BookInfo bkInfo;


	public StudentInfo getStInfo() {
		return stInfo;
	}


	public void setStInfo(StudentInfo stInfo) {
		this.stInfo = stInfo;
	}


	public BookInfo getBkInfo() {
		return bkInfo;
	}


	public void setBkInfo(BookInfo bkInfo) {
		this.bkInfo = bkInfo;
	}
	/**
 * Method to validate student id  and throws exception if validation fails
 *
 * @exception InvalidStudentIdException
 */

public void validateStudentId(int stid) throws InvalidStudentIdException
{//Insert code here..
        Map<Integer, String> stdetails = getStInfo().getStDetails();
        Set<Integer> stids = stdetails.keySet();
        boolean flag=false;
        for (Integer id : stids) {
                if (id==stid) {
                        flag=true;
                }
        }
          if(flag==false)
                 throw new InvalidStudentIdException("The given Student id does not exist!!!");
}

/**
 * Method to validate book name  and   throws exception if validation fails
 *
 * @exception InvalidBookNameException
 */
public void validateBookName(String bookname) throws InvalidBookNameException
{//Insert code here..
        Map<String, Integer> bkdetails = getBkInfo().getBookDetails();
        Set<String> bknames = bkdetails.keySet();
        boolean flag=false;
```

```
                for(String name:bknames)

                {

                        if(name.equalsIgnoreCase(bookname))

                        {

                                flag=true;

                        }

                }

                if(flag==false)

                         throw new InvalidBookNameException("The given Book Name does not
exist!!!");


        }


        /**

         * Method checks no. Of copies of the given Bookname and return updated noOfCopies

         */


        public int checkNoOfCopies(String bookname)

         {//Insert code here..

                Map<String, Integer> bkdetails = getBkInfo().getBookDetails();

                Integer value = bkdetails.get(bookname);

                int copies=value;

                if(copies>0)

                {

                        copies=copies-1;

                        bkdetails.put(bookname,copies);

                }

                 return value;

                 //return 0;
            }

        public int updateNoOfCopies(String bookname) {

                //Insert code here...

                Map<String, Integer> bkdetails = getBkInfo().getBookDetails();

                Integer value = bkdetails.get(bookname);


                        value=value+1;

                        bkdetails.put(bookname, value);

                        return value;

                //return -1;


        }


}
```

================

```
package com.cts.library.exception;

public class InvalidBookNameException extends Exception {
	public InvalidBookNameException(String msg) {
		//Insert code here..
		super(msg);
	}
}
```

========

```
package com.cts.library.exception;

public class InvalidStudentIdException extends Exception{


	public InvalidStudentIdException(String msg)
	{
```

```
            //Insert code here..
            super(msg);

        }
}




=============

package com.cts.library.main;

import java.util.Scanner;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.cts.library.service.LibraryService;
import com.cts.library.skeletonvalidator.SkeletonValidator;

public class LibraryMgmtApplication {
        public static void main(String[] args) {
                String choice = "yes";
                SkeletonValidator validator = new SkeletonValidator();
                ApplicationContext ctx = new
ClassPathXmlApplicationContext("beans.xml");
                // Insert code here..
                LibraryService ls= (LibraryService) ctx.getBean("libService");
                do {
                        System.out.println("Choose the option:- 1 . Issue Book   2.
Return  Book");
                        Scanner sc = new Scanner(System.in);
                        int ch = sc.nextInt();
                        if (ch == 1) {
                                // Insert code here...
                                System.out.println("Enter student id");
                    int stid=sc.nextInt();
                    if(ls.validateStudentId(stid)) {

                    System.out.println("Enter Book Name");
                    String bookname=sc.next();
                    if(ls.validateBookName(bookname)){

                        ls.issueBook(stid, bookname);
                    }
                    }

                        } else if (ch == 2) {
                                // Insert code here...
                                System.out.println("Enter student id");
                    int stid=sc.nextInt();
                    if(ls.validateStudentId(stid)) {
```

```
                System.out.println("Enter Book Name");
                String bookname=sc.next();
                if(ls.validateBookName(bookname)) {

                System.out.println("Enter Issue Date");
                String idte=sc.next();
                System.out.println("Enter Return Date");
                String rdte=sc.next();

                DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy-MM-dd");
                LocalDate issueDate = LocalDate.parse(idte,dtf);
                LocalDate returnDate = LocalDate.parse(rdte,dtf);

                     System.out.println(ls.returnBook(stid, bookname, issueDate, 
returnDate));
                }
         }
                    } else {
                          System.out.println("Invalid Choice");
                    }
                    System.out.println("Would you like to continue?yes/no");
                    choice = sc.next();

             } while (choice.equalsIgnoreCase("yes"));
      }
}
```

```
==============

package com.cts.library.model;

import java.util.Map;

public class BookInfo {
      Map<String,Integer> bookDetails;

      public Map<String, Integer> getBookDetails() {
             return bookDetails;
      }

      public void setBookDetails(Map<String, Integer> bookDetails) {
             this.bookDetails = bookDetails;
      }
}


===============

package com.cts.library.model;
```

```
import java.util.Map;

public class StudentInfo {
	Map<Integer,String> stDetails;

	public Map<Integer, String> getStDetails() {
		return stDetails;
	}

	public void setStDetails(Map<Integer, String> stDetails) {
		this.stDetails = stDetails;
	}
}


==================

package com.cts.library.service;

import java.time.LocalDate;
import java.time.Period;
import java.time.format.DateTimeFormatter;
import java.util.Map;
import java.util.Set;

import com.cts.library.bo.LibraryBo;
import com.cts.library.exception.InvalidBookNameException;
import com.cts.library.exception.InvalidStudentIdException;
public class LibraryService {
	private LibraryBo libBo;

	public LibraryBo getLibBo() {
		return libBo;
	}

	public void setLibBo(LibraryBo libBo) {
		this.libBo = libBo;
	}

	/**
	 * Method to call validateStudentId and handles exception and returns false if
	 * validation fails
	 *
	 * @exception InvalidStudentIdException
	 */

	public boolean validateStudentId(int stid)
	{// Insert code here..
		try {
		         libBo.validateStudentId(stid);
		}
		catch(InvalidStudentIdException e)
		{
			System.out.println(e.getMessage());
```

```
        }
        Map<Integer, String> stdetails = libBo.getStInfo().getStDetails();
        Set<Integer> stids = stdetails.keySet();
        for (Integer id : stids) {
              if (id==stid) {
                      return true;
              }
        }
        return false;
  }

  /**
   * Method to call validateBookName and handles exception and returns false if
   * validation fails
   *
   * @exception InvalidBookNameException
   */

  public boolean validateBookName(String bookname)
  {
        // Insert code here..
        try {
               libBo.validateBookName(bookname);
              }
              catch(InvalidBookNameException e)
              {
                    System.out.println(e.getMessage());
              }
        Map<String, Integer> bkdetails = libBo.getBkInfo().getBookDetails();
        Set<String> bknames = bkdetails.keySet();
        for(String name:bknames)
        {
              if(name.equalsIgnoreCase(bookname))
              {
                    return true;
              }
        }
        return false;
  }

  /**
   * Method to check NoOfCopies and return appropriate message if the book is
   * available
   *
   */
  public String issueBook(int stid, String bookname) {// Insert code here;
        int copyes=libBo.checkNoOfCopies(bookname);
        if(copyes<=0)
        {

              try {
        throw new InvalidBookNameException("Sorry!!!The Book is not available");
              }
              catch(InvalidBookNameException e)
              {
```

```
                        System.out.println(e.getMessage());
                    }
                }
                else
                {
                    if(copyes>0)
                    {
                        //successfully issued
                  LocalDate issue_date = LocalDate.now();
                  LocalDate return_date = issue_date.plusDays(7);
                  System.out.println("The Book Issued Successfully.Return date
is"+return_date);
                    }
                }
                return null;
        }

        /**
         * Method to calculate fine on the basis of issue date and return date and
         * return appropriate message
         *
         */

        public String returnBook(int stid, String bookname, LocalDate issueDate,
LocalDate returnDate) {
                // Insert code here;
                libBo.updateNoOfCopies(bookname);

                int fine = calculateFine(issueDate, returnDate);

                String ID=issueDate.format(DateTimeFormatter.ofPattern("dd/MM/yyyy"));
                String RD=returnDate.format(DateTimeFormatter.ofPattern("dd/MM/yyyy"));

                String Fine=String.valueOf(fine);
                String SID=String.valueOf(stid);

                return "Student Id:"+SID+"\nBook
Name:"+bookname+"\nIssueDate:"+ID+"\nReturn Date:"+RD+"\nFine:"+Fine;

        }

        /**
         * Method to calculate fine on the basis of issue date and return date and
         * return the fine
         */

        public int calculateFine(LocalDate issueDate, LocalDate returnDate) {
                // Insert code here;
                LocalDate issueDatePlus7 = issueDate.plusDays(7);
         if(returnDate.compareTo(issueDatePlus7)>0){
                Period period = Period.between(returnDate, issueDatePlus7);
                int days=period.getDays();
                return days*50;
         }
         else{
```

```
			return 0;
		}
		}
}
```

============

```
package com.cts.library.skeletonvalidator;


import java.lang.reflect.Method;

import java.util.logging.Level;

import java.util.logging.Logger;


public class SkeletonValidator {

	public SkeletonValidator() {

		validateClassName("com.cts.library.service.LibraryService");

		validateClassName("com.cts.library.bo.LibraryBo");

		validateClassName("com.cts.library.model.BookInfo");

		validateClassName("com.cts.library.model.StudentInfo");




		validateMethodSignature(

	"validateStudentId:boolean,validateBookName:boolean,issueBook:java.lang.String,"

				+ "returnBook:java.lang.String,calculateFine:int",

				"com.cts.library.service.LibraryService");

		validateMethodSignature(

	"validateStudentId:void,validateBookName:void,checkNoOfCopies:int,updateNoOfCopies:int",

				"com.cts.library.bo.LibraryBo");
	}
	private static final Logger LOG = Logger.getLogger("SkeletonValidator");
	protected final boolean validateClassName(String className) {


		boolean iscorrect = false;
		try {
			Class.forName(className);
			iscorrect = true;
			LOG.info("Class Name " + className + " is correct");


		} catch (ClassNotFoundException e) {
			LOG.log(Level.SEVERE, "You have changed either the " + "class name/package.
Use the correct package "
					+ "and class name as provided in the skeleton");


		} catch (Exception e) {
			LOG.log(Level.SEVERE,
					"There is an error in validating the " + "Class Name. Please
manually verify that the "
							+ "Class name is same as skeleton before
uploading");
		}
		return iscorrect;
	}


	protected final void validateMethodSignature(String methodWithExcptn, String className) {
		Class cls = null;
		try {
```

```
                    String[] actualmethods = methodWithExcptn.split(",");
                    boolean errorFlag = false;
                    String[] methodSignature;
                    String methodName = null;
                    String returnType = null;

                    for (String singleMethod : actualmethods) {
                        boolean foundMethod = false;
                        methodSignature = singleMethod.split(":");

                        methodName = methodSignature[0];
                        returnType = methodSignature[1];
                        cls = Class.forName(className);
                        Method[] methods = cls.getMethods();
                        for (Method findMethod : methods) {
                            if (methodName.equals(findMethod.getName())) {
                                foundMethod = true;
                                if
(!(findMethod.getReturnType().getName().equals(returnType))) {
                                    errorFlag = true;
                                    LOG.log(Level.SEVERE, " You have changed the "
+ "return type in '" + methodName
                                            + "' method. Please stick to the
" + "skeleton provided");

                                } else {
                                    LOG.info("Method signature of " +
methodName + " is valid");
                                }
```

```
                                        }
                                }
                                if (!foundMethod) {
                                        errorFlag = true;
                                        LOG.log(Level.SEVERE, " Unable to find the given public method
" + methodName
                                                        + ". Do not change the " + "given public method
name. " + "Verify it with the skeleton");
                                }


                        }
                        if (!errorFlag) {
                                LOG.info("Method signature is valid");
                        }


                } catch (Exception e) {
                        LOG.log(Level.SEVERE,
                                        " There is an error in validating the " + "method structure.
Please manually verify that the "
                                                        + "Method signature is same as the skeleton
before uploading");
                }
        }


}
===========
```

```
<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"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xmlns:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="http://www.springframework.org/schema/aop
       http://www.springframework.org/schema/aop/spring-aop-3.2.xsd
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
         http://www.springframework.org/schema/context
         http://www.springframework.org/schema/context/spring-context-3.2.xsd

           http://www.springframework.org/schema/tx
           http://www.springframework.org/schema/tx/spring-tx-3.2.xsd">


<!-- Insert code here -->
   <bean class="com.cts.library.model.StudentInfo" name="stinfo">
      <property name="stDetails" >
        <map>
             <entry key="2" value="abcd" />
             <entry key="1" value="efgh" />
             <entry key="3" value="ijkl" />
             <entry key="4" value="lmop" />
             <entry key="5" value="qrst" />
           </map>
         </property>
       </bean>

   <bean class="com.cts.library.model.BookInfo"  name="bkinfo">
     <property name="bookDetails" >
        <map>
             <entry key="a" value="1" />
             <entry key="b" value="2" />
             <entry key="c" value="3" />
             <entry key="d" value="4" />
             <entry key="e" value="5" />
           </map>
        </property>
   </bean>


       <bean  class="com.cts.library.bo.LibraryBo"  name="libBo">
      <property name="stInfo" ref="stinfo" />
      <property name="bkInfo" ref="bkinfo" />
    </bean>

       <bean class="com.cts.library.service.LibraryService"  name="libService">
      <property name="libBo" ref="libBo"/>
    </bean>

</beans>


=============
```