

Donate

application.properties

```
1
2 #Database properties
3 spring.datasource.url=jdbc:mysql://localhost:3306/hospital_management_db
4 spring.datasource.username=root
5 spring.datasource.password=root
6 spring.jpa.show-sql=true
7 spring.jpa.properties.hibernate.format_sql=true
8 server.port=8765
9
10
11 #Validator Messages
12 PatientValidator.INVALID_DOB=Date of birth should not be a future date and the age should not be more than 16
13
14 #General Messages
15 General.EXCEPTION_MESSAGE=Request could not be processed due to some issue. Please try again!
16
17 #Service Messages
18 PatientService.PATIENT_UNAVAILABLE=No patient available with given diagnosis
19
```

Writable

Insert

[Postman]

[Mozilla Firefox]

workspace - iap...

Project Explorer

- ▶ iap_hospitalmanagement [user_repo master]
 - ▶ src/main/java
 - ▶ com.infy.hospitalmanagement
 - ▶ HospitalManagementApplication.java
 - ▶ com.infy.hospitalmanagement.api
 - ▶ PatientAPI.java
 - ▶ com.infy.hospitalmanagement.dto
 - ▶ PatientDTO.java
 - ▶ com.infy.hospitalmanagement.entity
 - ▶ Patient.java
 - ▶ com.infy.hospitalmanagement.exception
 - ▶ PatientAdmissionException.java
 - ▶ com.infy.hospitalmanagement.repository
 - ▶ PatientRepository.java
 - ▶ com.infy.hospitalmanagement.service
 - ▶ PatientService.java
 - ▶ PatientServiceImpl.java
 - ▶ com.infy.hospitalmanagement.utility
 - ▶ ErrorInfo.java
 - ▶ ExceptionControllerAdvice.java
 - ▶ com.infy.hospitalmanagement.validator
 - ▶ PatientValidator.java

- ▶ src/main/resources

- ▶ src/test/java

- ▶ JRE System Library [JavaSE-11]

- ▶ Maven Dependencies

com.infy.hospitalmanagement.val... ospitalmanagement/src/main/java

Project Explorer

- iap_hospitalmanagement (user_repo master)
 - src/main/java
 - com.infy.hospitalmanagement
 - > HospitalManagementApplication.java
 - com.infy.hospitalmanagement.api
 - > PatientAPI.java
 - com.infy.hospitalmanagement.dto
 - > PatientDTO.java
 - com.infy.hospitalmanagement.entity
 - > Patient.java
 - com.infy.hospitalmanagement.exception
 - > PatientAdmissionException.java
 - com.infy.hospitalmanagement.repository
 - > PatientRepository.java
 - com.infy.hospitalmanagement.service
 - > PatientService.java
 - > PatientServiceImpl.java
 - com.infy.hospitalmanagement.utility
 - > ErrorInfo.java
 - > ExceptionControllerAdvice.java
 - com.infy.hospitalmanagement.validator
 - > PatientValidator.java
 - src/main/resources
 - src/test/java
 - JRE System Library [JavaSE-11]
 - Maven Dependencies

com.infy.hospitalmanagement.val...ospitalmanagement/src/main/java

[Mozilla Firefox] workspace - iap...

```
1 package com.infy.hospitalmanagement.validator;
2 import java.time.LocalDate;
3 public class PatientValidator {
4
5
6
7     public static void validatePatient(PatientDTO patientDTO) throws PatientAdmissionException{
8         }
9
10
11
12     public static Boolean isValidDateOfBirth(LocalDate dateOfBirth) {
13         return null;
14     }
15
16
17 }
18
19
20
21 }
22
```

Writable

Smart Insert



[Postman]



[Mozilla Firefox]

workspace - lap...

```
ExceptionControllerAdvice.java - Eclipse IDE
1 public class ExceptionControllerAdvice {
2     private static final Logger LOGGER = LoggerFactory.getLogger(ExceptionControllerAdvice.class);
3
4     @ExceptionHandler
5     public ResponseEntity<ErrorInfo> patientAdmissionExceptionHandler(PatientAdmissionException exception) {
6         ErrorInfo errorInfo = new ErrorInfo();
7         errorInfo.setErrorCode(HttpStatus.BAD_REQUEST.value());
8         errorInfo.setErrorMassage(exception.getMessage());
9         return new ResponseEntity<>(errorInfo, HttpStatus.BAD_REQUEST);
10    }
11
12    @ExceptionHandler
13    public ResponseEntity<ErrorInfo> generalExceptionHandler(Exception exception) {
14        ErrorInfo errorInfo = new ErrorInfo();
15        errorInfo.setErrorCode(HttpStatus.INTERNAL_SERVER_ERROR.value());
16        errorInfo.setErrorMassage("General.EXCEPTION_MESSAGE");
17        return new ResponseEntity<>(errorInfo, HttpStatus.INTERNAL_SERVER_ERROR);
18    }
19
20    @ExceptionHandler
21    public ResponseEntity<ErrorInfo> validatorExceptionHandler(Exception exception) {
22        String errorMsg;
23        if (exception instanceof MethodArgumentNotValidException) {
24            MethodArgumentNotValidException manvException = (MethodArgumentNotValidException) exception;
25            errorMsg = manvException.getBindingResult().getAllErrors().stream().map(ObjectError::getDefaultMessage)
26                .collect(Collectors.joining(", "));
27        } else {
28            ConstraintViolationException cvException = (ConstraintViolationException) exception;
29            errorMsg = cvException.getConstraintViolations().stream().map(ConstraintViolation::getMessage)
30                .collect(Collectors.joining(", "));
31        }
32
33        ErrorInfo errorInfo = new ErrorInfo();
34        errorInfo.setErrorCode(HttpStatus.BAD_REQUEST.value());
35        errorInfo.setErrorMassage(errorMsg);
36        return new ResponseEntity<>(errorInfo, HttpStatus.BAD_REQUEST);
37    }
38
39 }
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59 }
```

Writable

Smart Insert



workspace - iap...



```
ExceptionControllerAdvice.java - Eclipse IDE
```

```
1 package com.infy.hospitalmanagement.utility;
2
3 import java.util.stream.Collectors;
4
5
6 public class ExceptionControllerAdvice {
7     private static final Log LOGGER = LogFactory.getLog(ExceptionControllerAdvice.class);
8     private Environment environment;
9
10    public ResponseEntity<ErrorInfo> patientAdmissionExceptionHandler(PatientAdmissionException exception) {
11        LOGGER.error(exception.getMessage(), exception);
12        ErrorInfo errorInfo = new ErrorInfo();
13        errorInfo.setErrorCode(HttpStatus.BAD_REQUEST.value());
14        errorInfo.setErrorMessage(environment.getProperty(exception.getMessage()));
15        return new ResponseEntity<>(errorInfo, HttpStatus.BAD_REQUEST);
16    }
17
18    public ResponseEntity<ErrorInfo> generalExceptionHandler(Exception exception) {
19        LOGGER.error(exception.getMessage(), exception);
20        ErrorInfo errorInfo = new ErrorInfo();
21        errorInfo.setErrorCode(HttpStatus.INTERNAL_SERVER_ERROR.value());
22        errorInfo.setErrorMessage(environment.getProperty("General.EXCEPTION_MESSAGE"));
23        return new ResponseEntity<>(errorInfo, HttpStatus.INTERNAL_SERVER_ERROR);
24    }
25
26    public ResponseEntity<ErrorInfo> validatorExceptionHandler(Exception exception) {
27        LOGGER.error(exception.getMessage(), exception);
28        String errorMsg;
29        if (exception instanceof MethodArgumentNotValidException) {
30            MethodArgumentNotValidException manvException = (MethodArgumentNotValidException) exception;
31            errorMsg = manvException.getBindingResult().getAllErrors().stream().map(ObjectError::getDefaultMessage)
32                .collect(Collectors.joining(", "));
33        } else {
34            ConstraintViolationException cvException = (ConstraintViolationException) exception;
35            errorMsg = cvException.getConstraintViolations().stream().map(ObjectError::getDefaultMessage)
36                .collect(Collectors.joining(", "));
37        }
38        ErrorInfo errorInfo = new ErrorInfo();
39        errorInfo.setErrorCode(HttpStatus.BAD_REQUEST.value());
40        errorInfo.setErrorMessage(errorMsg);
41        return new ResponseEntity<>(errorInfo, HttpStatus.BAD_REQUEST);
42    }
43
44    public ResponseEntity<ErrorInfo> fileUploadExceptionHandler(FileUploadException exception) {
45        LOGGER.error(exception.getMessage(), exception);
46        ErrorInfo errorInfo = new ErrorInfo();
47        errorInfo.setErrorCode(HttpStatus.BAD_REQUEST.value());
48        errorInfo.setErrorMessage(environment.getProperty("FileUpload.EXCEPTION_MESSAGE"));
49        return new ResponseEntity<>(errorInfo, HttpStatus.BAD_REQUEST);
50    }
51}
```

writable

Smart Insert

[Postman]

[Mozilla Firefox]

workspace - iap...



```
● Donate  ExceptionControllerAdvice.java
24 /**
25  * @param exception
26  * @return
27  */
28 @ExceptionHandler<ErrorInfo> patientAdmissionExceptionHandler(PatientAdmissionException exception) {
29     ErrorInfo errorInfo = new ErrorInfo();
30     errorInfo.setErrorCode(HttpStatus.BAD_REQUEST.value());
31     errorInfo.setErrorMessage(environment.getProperty(exception.getMessage()));
32     return new ResponseEntity<>(errorInfo, HttpStatus.BAD_REQUEST);
33 }
34 /**
35  * @param exception
36  * @return
37  */
38 @ExceptionHandler<ErrorInfo> generalExceptionHandler(Exception exception) {
39     ErrorInfo errorInfo = new ErrorInfo();
40     errorInfo.setErrorCode(HttpStatus.INTERNAL_SERVER_ERROR.value());
41     errorInfo.setErrorMessage(environment.getProperty("General.EXCEPTION_MESSAGE"));
42     return new ResponseEntity<>(errorInfo, HttpStatus.INTERNAL_SERVER_ERROR);
43 }
44 /**
45  * @param exception
46  * @return
47  */
48 @ExceptionHandler<ErrorInfo> validatorExceptionHandler(Exception exception) {
49     String errorMsg;
50     if (exception instanceof MethodArgumentNotValidException) {
51         MethodArgumentNotValidException manyException = (MethodArgumentNotValidException) exception;
52         errorMsg = manyException.getBindingResult().getAllErrors().stream().map(ObjectError::getDefaultMessage)
53             .collect(Collectors.joining(", "));
54     } else {
55         ConstraintViolationException cvException = (ConstraintViolationException) exception;
56         errorMsg = cvException.getConstraintViolations().stream().map(ConstraintViolation::getMessage)
57             .collect(Collectors.joining(", "));
58     }
59     ErrorInfo errorInfo = new ErrorInfo();
60     errorInfo.setErrorCode(HttpStatus.BAD_REQUEST.value());
61     errorInfo.setErrorMessage(errorMsg);
62     return new ResponseEntity<>(errorInfo, HttpStatus.BAD_REQUEST);
63 }
```

Writable

Smart Insert

[Postman]

[Mozilla Firefox]

workspace - iap...



```
● Donate  ExceptionControllerAdvice.java □
1  com.infy.hospitalmanagement.utility;
2
3  ava.util.stream.Collectors;
4
5
6
7
8
9  class ExceptionControllerAdvice {
10
11     static final Log LOGGER = LoggerFactory.getLog(ExceptionControllerAdvice.class);
12
13     Environment environment;
14
15     ResponseEntity<ErrorInfo> patientAdmissionExceptionHandler(PatientAdmissionException exception) {
16         LOGGER.error(exception.getMessage(), exception);
17         ErrorInfo errorInfo = new ErrorInfo();
18         errorInfo.setErrorCode(HttpStatus.BAD_REQUEST.value());
19         errorInfo.setErrorMassage(environment.getProperty(exception.getMessage()));
20         return new ResponseEntity<>(errorInfo, HttpStatus.BAD_REQUEST);
21     }
22
23
24
25     ResponseEntity<ErrorInfo> generalExceptionHandler(Exception exception) {
26         LOGGER.error(exception.getMessage(), exception);
27         ErrorInfo errorInfo = new ErrorInfo();
28         errorInfo.setErrorCode(HttpStatus.INTERNAL_SERVER_ERROR.value());
29         errorInfo.setErrorMassage(environment.getProperty("General.EXCEPTION_MESSAGE"));
30         return new ResponseEntity<>(errorInfo, HttpStatus.INTERNAL_SERVER_ERROR);
31     }
32
33
34
35     ResponseEntity<ErrorInfo> validatorExceptionHandler(Exception exception) {
36         LOGGER.error(exception.getMessage(), exception);
37         String errorMsg;
38         if (exception instanceof MethodArgumentNotValidException) {
39             MethodArgumentNotValidException manvException = (MethodArgumentNotValidException) exception;
40             errorMsg = manvException.getBindingResult().getAllErrors().stream().map(ObjectError::getDefaultMessage)
41             .collect(Collectors.joining(", "));
42         } else {
43             ConstraintViolationException cuException = (ConstraintViolationException) exception;
44         }
45     }
46
47
48
49
```

Writable

Smart Insert



[Postman]



[Mozilla Firefox]

workspace - iap...



Project Explorer

DomTree ErrorInfo.java

```
1 package com.infy.hospitalmanagement.utility;
2
3 public class ErrorInfo {
4     private String errorMessage;
5     private Integer errorCode;
6
7     public String getErrorMessage() {
8         return errorMessage;
9     }
10
11    public void setErrorMessage(String errorMessage) {
12        this.errorMessage = errorMessage;
13    }
14
15    public Integer getErrorCode() {
16        return errorCode;
17    }
18
19    public void setErrorCode(Integer errorCode) {
20        this.errorCode = errorCode;
21    }
22 }
23
```

Smart Insert

writable

Mozilla Firefox workspace - iap...

Project Explorer

PatientServiceimpl.java [13]

```
1 package com.infy.hospitalmanagement.service;
2 import java.util.List;
3 public class PatientServiceimpl implements PatientService{
4     private PatientRepository patientRepository;
5     @Override
6     public List<PatientDTO> getListOfPatients(String diagnosis) throws PatientAdmis-
7         return null;
8     }
9     @Override
10    public PatientDTO admitPatient(PatientDTO patientDTO) throws PatientAdmissionEx-
11        return null;
12    }
13 }
```

Writable

Smart Insert

File Explorer Project Structure Tools

Donate PatientService.java

```
1 com.infy.hospitalmanagement.service;
2
3 @java.util.List; []
4
5 interface PatientService {
6     <PatientDTO> getListOfPatients(String diagnosis) throws PatientAdmissionException;
7     PatientDTO admitPatient(PatientDTO patientDTO) throws PatientAdmissionException;
8
9 }
```

src/main/java
com.infy.hospitalmanagement
HospitalManagement
com.infy.hospitalmanagement
PatientAPI.java
com.infy.hospitalmanagement
PatientDTO.java
com.infy.hospitalmanagement
Patient.java
com.infy.hospitalmanagement
PatientAdmissionException
com.infy.hospitalmanagement
PatientRepository.java
com.infy.hospitalmanagement
PatientService.java
PatientServiceImpl.java
com.infy.hospitalmanagement
com.infy.hospitalmanagement
resources
java
Library [javaSE-
Dependencies

Smart Insert

The screenshot shows a Java development environment with the following details:

- Project Explorer:** Displays the project structure under "iap_hospitalmanagement".
 - src/main/java:** Contains packages for HospitalManagement, PatientAPI, PatientDTO, Patient, PatientAdmissionEx, and PatientRepository.
 - src/main/resources:**
 - src/test/java:**
 - JRE System Library [JavaSE-17]:**
 - Maven Dependencies:**
 - jUnit 5:**
 - target/generated-sources/all:**
 - com.infy.hospitalmanagement boot source:**
- PatientRepository.java:** The current file being edited, located at `com.infy.hospitalmanagement.repository`. The code is as follows:

```
1 package com.infy.hospitalmanagement.repository;
2
3
4 public interface PatientRepository{
5 }
6
```
- Smart Insert:** A status bar at the bottom right of the code editor.

The screenshot shows the Eclipse IDE interface. The Project Explorer view on the left displays the project structure for 'iap_hospitalmanagement' with several Java source files under 'src/main/java'. The 'PatientAdmissionException.java' file is currently selected in the editor, and its code is visible:

```
1 package com.infy.hospitalmanagement.exception;
2
3 public class PatientAdmissionException extends Exception {
4     private static final long serialVersionUID = 1L;
5     public PatientAdmissionException(String message) {
6         super(message);
7     }
8 }
9
10
11 }
```

The code editor has tabs for 'PatientAdmissionException.java' and 'PatientAdmissionException.java' (with a red X). The status bar at the bottom shows 'Writable' and 'Smart Insert'.

Project Explorer

-> iap_hospitalmanagement [4]
 -> src/main/java
 -> com.infy.hospitalmana
 -> HospitalManagement
 -> com.infy.hospitalmana
 -> PatientAPI.java
 -> com.infy.hospitalmana
 -> PatientDTO.java
 -> com.infy.hospitalmana
 -> Patient.java
 -> com.infy.hospitalmana
 -> com.infy.hospitalmana
 -> com.infy.hospitalmana
 -> com.infy.hospitalmana
 -> com.infy.hospitalmana
 -> src/main/resources
 -> src/test/java
 -> JRE System Library [JavaSE-
 -> Maven Dependencies
 -> JUnit 5
 -> target/generated-sources/a
 -> target/generated-test-sourc
 -> target\generated-sources\a
 -> target\generated-test-sourc

Donate

Patient.java

```
32  
33  public void setGender(String gender) {  
34      this.gender = gender;  
35  }  
36  
37  public LocalDate getDateOfBirth() {  
38      return dateOfBirth;  
39  }  
40  
41  public void setDateOfBirth(LocalDate dateOfBirth) {  
42      this.dateOfBirth = dateOfBirth;  
43  }  
44  
45  public LocalDate getAdmissionDate() {  
46      return admissionDate;  
47  }  
48  
49  public void setAdmissionDate(LocalDate admissionDate) {  
50      this.admissionDate = admissionDate;  
51  }  
52  
53  public String getDiagnosis() {  
54      return diagnosis;  
55  }  
56  
57  public void setDiagnosis(String diagnosis) {  
58      this.diagnosis = diagnosis;  
59  }  
60  
61  
62  
63  
64  
65 }  
66
```

Writable

Smart Insert



[Postman]



[Mozilla Firefox]



workspace - iap...

Project Explorer

iap_hospitalmanagement []

- > src/main/java
 - > com.infy.hospitalmana
 - > HospitalManagement.java
 - > PatientAPI.java
 - > PatientDTO.java
 - > Patient.java
 - > com.infy.hospitalmana
 - > src/main/resources
 - > src/test/java
- > JRE System Library [JavaSE-1.8]
- > Maven Dependencies
- > JUnit 5
- > target/generated-sources/all
- > target/generated-test-sources/all
- > target\generated-sources\all
- > target\generated-test-sources\all

Patient.java

```
1 package com.infy.hospitalmanagement.entity;
2 import java.time.LocalDate;
3
4 public class Patient {
5     private Integer patientId;
6     private String patientName;
7     private String gender;
8     private LocalDate dateOfBirth;
9     private LocalDate admissionDate;
10    private String diagnosis;
11
12    public Integer getPatientId() {
13        return patientId;
14    }
15
16    public void setPatientId(Integer patientId) {
17        this.patientId = patientId;
18    }
19
20    public String getPatientName() {
21        return patientName;
22    }
23
24    public void setPatientName(String patientName) {
25        this.patientName = patientName;
26    }
27
28    public String getGender() {
29        return gender;
30    }
31
32    public void setGender(String gender) {
33        this.gender = gender;
34    }
35 }
```

Writable

Smart Insert



The screenshot shows an IDE interface with two main panes. The left pane displays a file tree for the project `iap_hospitalmanagement`. The right pane shows the content of the file `PatientDTO.java`.

File Tree (Left):

- > `iap_hospitalmanagement [J]`
 - > `src/main/java`
 - > `com.infy.hospitalmana`
 - > `HospitalManagement`
 - > `com.infy.hospitalmana`
 - > `PatientAPI.java`
 - > `com.infy.hospitalmana`
 - > `PatientDTO.java`
 - > `com.infy.hospitalmana`
 - > `com.infy.hospitalmana`
 - > `src/main/resources`
 - > `src/test/java`
 - > `JRE System Library [JavaSE-1.8]`
 - > `Maven Dependencies`
 - > `JUnit 5`
 - > `target/generated-sources/annotations`
 - > `target/generated-test-sources/annotations`
 - > `target/generated-sources/compile`
 - > `target/generated-test-sources/compile`
 - > `log`

PatientDTO.java Content (Right):

```
53  
54  
55  
56 }  
57  
58  
59  
60 public String getDiagnosis() {  
61     return diagnosis;  
62 }  
63  
64 public void setDiagnosis(String diagnosis) {  
65     this.diagnosis = diagnosis;  
66 }  
67  
68 public static Patient prepareEntity(PatientDTO patientDTO) {  
69     Patient patient = new Patient();  
70     patient.setPatientName(patientDTO.getPatientName());  
71     patient.setGender(patientDTO.getGender());  
72     patient.setDiagnosis(patientDTO.getDiagnosis());  
73     patient.setAdmissionDate(patientDTO.getAdmissionDate());  
74     patient.setDateOfBirth(patientDTO.getDateOfBirth());  
75     return patient;  
76 }  
77  
78 public static PatientDTO prepareDTO(Patient patient) {  
79     PatientDTO patientDTO = new PatientDTO();  
80     patientDTO.setPatientId(patient.getId());  
81     patientDTO.setPatientName(patient.getName());  
82     patientDTO.setGender(patient.getGender());  
83     patientDTO.setDiagnosis(patient.getDiagnosis());  
84     patientDTO.setAdmissionDate(patient.getAdmissionDate());  
85     patientDTO.setDateOfBirth(patient.getDateOfBirth());  
86 }  
87 }
```

Writable

Smart Insert



[Postman]



[Mozilla Firefox]



workspace - iap...



The screenshot shows a Java project structure on the left and the content of PatientDTO.java on the right.

Project Structure:

- iap_hospitalmanagement [4]
- src/main/java
 - com.infy.hospitalmanag
 - > HospitalManagement
 - > PatientAPI.java
 - > com.infy.hospitalmanag
 - > PatientDTO.java
 - > com.infy.hospitalmanag
 - > com.infy.hospitalmanag
 - > src/main/resources
 - > src/test/java- JRE System Library [JavaSE-11]
- Maven Dependencies
- JUnit 5
- target/generated-sources/avro
- target/generated-test-sources/avro
- target\generated-sources\java
- target\generated-test-sources\java

PatientDTO.java Content:

```
1 package com.infy.hospitalmanagement.dto;
2 import java.time.LocalDate;
3
4 public class PatientDTO {
5     private Integer patientId;
6     private String patientName;
7     private String gender;
8     private LocalDate dateOfBirth;
9     private LocalDate admissionDate;
10    private String diagnosis;
11
12    public Integer getPatientId() {
13        return patientId;
14    }
15
16    public void setPatientId(Integer patientId) {
17        this.patientId = patientId;
18    }
19
20    public String getPatientName() {
21        return patientName;
22    }
23
24    public void setPatientName(String patientName) {
25        this.patientName = patientName;
26    }
27
28    public String getGender() {
29        return gender;
30    }
31
32    public void setGender(String gender) {
33        this.gender = gender;
34    }
35
36}
```

Writable

Smart Insert



[Postman]

[Mozilla Firefox]

workspace - iap...

```
1 package com.infy.hospitalmanagement.api;
2
3 import java.util.List;
4
5 public class PatientAPI {
6     private PatientService patientService;
7
8     public ResponseEntity<List<PatientDTO>> getPatientsByDiagnosis(String diagnosis)
9         throws PatientAdmissionException {
10        return null;
11    }
12
13    public ResponseEntity<PatientDTO> admitPatient(PatientDTO patientDTO)
14        throws PatientAdmissionException {
15        return null;
16    }
17
18 }
19
20
21
22
23
24 }
25
```



```
1 package com.infy.hospitalmanagement;
2
3 import org.springframework.boot.SpringApplication;
4
5 @SpringBootApplication
6 public class HospitalManagementApplication {
7
8     public static void main(String[] args) {
9         SpringApplication.run(HospitalManagementApplication.class, args);
10    }
11 }
12
13 }
14
```

writable

Smart Insert

workspace - iap...

[Mozilla Firefox]

[Postman]

```
"patientName": "Arne  
Johansson",  
"gender": "Female",  
"dateOfBirth": "1995-06-10",  
"admissionDate": "**,"  
}
```

Replace ** with today's date

Invalid test data 7

api/patients

Sample JSON:

```
{  
    "patientName": "Arne  
Johansson",  
    "gender": "Female",  
    "dateOfBirth": "1995-06-10",  
    "admissionDate": "**,"  
    "diagnosis": "Cancer,"  
}
```

Replace ** with today's date

```
mandatory,  
"errorCode": 400  
}
```

```
{  
    "errorMessage": "Diagnosis  
should contain only numbers,  
alphabets and spaces",  
    "errorCode": 400  
}
```

PREV



PROJECT INSTRUCTIONS		PROBLEM STATEMENT
Invalid test data 5	<p>api/patients</p> <p>Sample JSON:</p> <pre>{ "patientName": "Arne Johansson", "gender": "Female", "dateOfBirth": "1995-06-10", "admissionDate": "2022-06-14", "diagnosis": "Cancer", }</pre>	<pre>{ "errorMessage": "Date of admission should be today's date", "errorCode": 400 }</pre>
Invalid test data 6	<p>api/patients</p> <p>Sample JSON:</p> <pre>{ "patientName": "Arne Johansson", "gender": "Female", "dateOfBirth": "1995-06-10", "admissionDate": "**", }</pre>	<pre>{ "errorMessage": "Diagnosis is mandatory", "errorCode": 400 }</pre>
Invalid test data 7	<p>Replace ** with today's date</p>	
	<p>api/patients</p> <p>Sample JSON:</p>	<pre>{ "errorMessage": "Diagnos should contain only numbers,"</pre>

PROJECT INSTRUCTIONS		PROBLEM STATEMENT
Invalid test data 3	Replace ** with today's date api/patients Sample JSON: { "patientName": "Arne Johansson", "gender": "Neutral", "dateOfBirth": "1995-06-10", "admissionDate": "**", "diagnosis": "Cancer" } Replace ** with today's date	{ "errorMessage": "Gender should be Male, Female or Others", "errorCode": 400 }
Invalid test data 4	api/patients Sample JSON: { "patientName": "Arne Johansson", "gender": "Female", "dateOfBirth": "1880-06-10", "admissionDate": "**", "diagnosis": "Cancer" } Replace ** with today's date	{ "errorMessage": "Date of birth should not be a future date and the age should not be more than 100", "errorCode": 400 }



Invalid test data 1

PROJECT INSTRUCTIONS

PROBLEM STATEMENT

api/patients

Sample JSON:
{
 "dateOfBirth": "1995-06-10",
 "admissionDate": "**",
 "diagnosis": "Cancer"
}

Replace ** with today's date

{
 "errorMessage": "Gender is mandatory for admission, Patient name is mandatory",
 "errorCode": 400
}

Invalid test data 2

api/patients

Sample JSON:
{
 "patientName": "Arne2 Johansson",
 "gender": "Female",
 "dateOfBirth": "1995-06-10",
 "admissionDate": "**",
 "diagnosis": "Cancer"
}

Replace ** with today's date

{
 "errorMessage": "Patient name should have only words and each word should be separated by a space",
 "errorCode": 400
}

Invalid test data 3

api/patients

Sample JSON:

{
 "patientName": "Arne

{
 "errorMessage": "Gender should be Male, Female or Others",
 "errorCode": 400
}



Postman

Mozilla Firefox

[workspace - ht...]



Admit a patient

Valid test data

api/patients

Sample JSON:

```
{  
  "patientName": "Arne  
Johansson",  
  "gender": "Female",  
  "dateOfBirth": "1995-06-10",  
  "admissionDate": "**",  
  "diagnosis": "Cancer"  
}
```

Replace ** with today's date

```
  "errorMessage": "No patient  
available with given diagnosis",  
  "errorCode": 400  
}
```

```
{  
  "participantId": 1007,  
  "patientName": "Arne  
Johansson",  
  "gender": "Female",  
  "dateOfBirth": "1995-06-10",  
  "admissionDate": "**",  
  "diagnosis": "Cancer",  
}
```

** will be replaced by today's date

Invalid test data 1

api/patients

Sample JSON:

```
{  
  "dateOfBirth": "1995-06-10",  
  "admissionDate": "**",  
  "diagnosis": "Cancer"  
}
```

Replace ** with today's date 

```
{  
  "errorMessage": "Gender is  
mandatory for admission, Patient  
name is mandatory",  
  "errorCode": 400  
}
```



Postman

Mozilla Firefox

[workspace - ht...]



PROJECT INSTRUCTIONS		PROBLEM STATEMENT
	Invalid test data 1	<pre>api/patients/Typhoid\$</pre> <pre> } { "errorMessage": "Diagnosis should contain only numbers, alphabets and spaces", "errorCode": 400 } </pre>
	Invalid test data 2	<pre>api/patients/Cholera</pre> <pre> { "errorMessage": "No patient available with given diagnosis", "errorCode": 400 } </pre>
Admit a patient	Valid test data	<p>api/patients</p> <p>Sample JSON:</p> <pre>{ "patientName": "Arne Johansson", "gender": "Female", "dateOfBirth": "1995-06-10", "admissionDate": "**", "diagnosis": "Cancer" }</pre> <p>Replace ** with today's date</p> <p>** will be replaced by today's date</p>



point

PROJECT INSTRUCTIONS

PROBLEM STATEMENT

Get all
the
patients
for the
diagnosis

Valid test data

api/patients/Typhoid

```
[  
 {  
 "patientId": 1006,  
 "patientName": "Mark",  
 "gender": "Male",  
 "dateOfBirth": "2015-09-12",  
 "admissionDate":  
 "2021-06-14",  
 "diagnosis": "Typhoid"  
 },  
 {  
 "patientId": 1001,  
 "patientName": "Matthew",  
 "gender": "Male",  
 "dateOfBirth": "1998-08-21",  
 "admissionDate":  
 "2021-01-02",  
 "diagnosis": "Typhoid"  
 },  
 {  
 "patientId": 1003,  
 "patientName": "John",  
 "gender": "Male",  
 "dateOfBirth": "2015-09-12",  
 "admissionDate":  
 "2020-09-20",  
 "diagnosis": "Typhoid"  
 }]
```

Go



- Add appropriate annotation to make this method as exception handler method for **MethodArgumentNotValidException** and **ConstraintViolationException** exceptions

Please find in the table below sample input and output for REST end points.

Note: The date values have been used considering the present date (today) to be 14th June, 2021. Please change the dates appropriately to test your implementation.

REST end point	Request Type(valid/invalid)	End point URI with sample input	Sample Response
Get all the patients for the diagnosis	Valid test data	api/patients/Typhoid	[{ "patientId": 1006, "patientName": "Mark", "gender": "Male", "dateOfBirth": "2015-09-12", "admissionDate": "2021-06-14", "diagnosis": "Typhoid" }, { "patientId": 1001, "patientName": "Matthew", "gender": "Male" }]

Go to

07



- Inject Environment objects using appropriate annotation
- The body of the methods have been implemented for you

Method Description:

1. **ResponseEntity<ErrorInfo> patientAdmissionExceptionHandler(PatientAdmissionException exception)**
 - This method is for handling **PatientAdmissionException** exception
 - Add appropriate annotation to make this method as exception handler for the PatientAdmissionException
2. **ResponseEntity<ErrorInfo> generalExceptionHandler(Exception exception)**
 - This method is for handling general exceptions
 - Add appropriate annotation to make this method as exception handler for any type of exceptions
3. **ResponseEntity<ErrorInfo> validatorExceptionHandler(Exception exception)**
 - This method is for handling **MethodArgumentNotValidException** and **ConstraintViolationException** exceptions
 - Add appropriate annotation to make this method as exception handler method for **MethodArgumentNotValidException** and **ConstraintViolationException** exceptions

Please find in the table below sample input and output for REST end points.

Note: The date values have been used considering the present date (today) to be 14th June, 2021. Please change the dates appropriately to test your implementation.

Postman

Mozilla Firefox

[workspace - ht...]



ASSESSMENT INSTRUCTIONS

PROJECT INSTRUCTIONS

PROBLEM STATEMENT

- **admissionDate:**

If admissionDate is not provided then set **Date of admission is mandatory** as the validation message

If it is provided, it should be today's date only. If invalid, set **Date of admission should be today's date** as the validation message

- **diagnosis:**

If diagnosis is not provided then set **Diagnosis is mandatory** as the validation message

If it is provided, it should contain only numbers, alphabets and spaces as the validation message

- Then invoke the appropriate service class method to admit the patient by passing the **PatientDTO** object which will return the same object with the **patientId** updated.
- Return an object of **ResponseEntity** with the PatientDTO object obtained in the previous step as the response and the HTTP response status as **CREATED**.

7. Utility Classes

com.infy.hospitalmanagement.utility.ExceptionControllerAdvice.java

Annotate the following ExceptionControllerAdvice to log and handle all the exceptions of the API class,

com.infy.hospitalmanagement.utility.ExceptionControllerAdvice	
▫	LOGGER: Log
▫	environment: Environment
●	patientAdmissionExceptionHandler(exception: PatientAdmissionException): ResponseEntity<ErrorInfo>
●	generalExceptionHandler(exception: Exception): ResponseEntity<ErrorInfo>
●	validatorExceptionHandler(exception: Exception): ResponseEntity<ErrorInfo>

- Inject Environment objects using appropriate annotation
- The body of the methods have been implemented for you



- PROJECT INSTRUCTIONS PROBLEM STATEMENT
- Return an object of **ResponseEntity** with the List<PatientDTO> object obtained in the previous step as a response and the HTTP response status as **OK**.

2. **public ResponseEntity<PatientDTO> admitPatient (PatientDTO patientDTO)**
- This method admits a patient in the HospitalManagement application
 - It accepts **POST** request with URI **patients**
 - It receives patient details in the form of JSON. The JSON data should be populated in **PatientDTO** parameter of this method after successful validation.
 - The patient details will consist of the patient name, gender, date of birth, admission date and diagnosis. Using **Bean Validation API** validate the details according to the description given below (Note: The validation message should not be hard-coded. It should be read from the **ValidationMessages.properties** file),

- **patientName:**

If **patientName** is not provided then set **Patient name is mandatory** as the validation message
If it is provided, it should contain only words of English alphabets and each word should be separated by a space. If invalid, set **Patient name should have only words and each word should be separated by a space** as the validation message

- **gender:**

If **gender** is not provided then set **Gender is mandatory for admission** as the validation message
If it is provided, it should be Male, Female or Others. If invalid, set **Gender should be Male, Female or Others** as the validation message

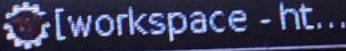
- **admissionDate:**

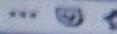
If **admissionDate** is not provided then set **Date of admission is mandatory** as the validation message

If it is provided, it should be today's date only. If invalid, set **Date of admission should be t**

Go to

07



**Method Description:**

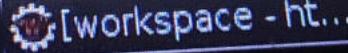
- 1. ResponseEntity<List<PatientDTO>> getPatientsByDiagnosis (String diagnosis)**
 - This method is to retrieve all the patients having the given diagnosis
 - It accepts **GET** request with URI **patients/{diagnosis}**
 - The path variable diagnosis should be validated as below (use **Bean Validation API** for validation).
 - **diagnosis** can have words containing alphabets and numbers, and each word should be separated by a single space. If invalid, set **Diagnosis should contain only numbers, alphabets and spaces** as the validation message

Note: The validation message should not be hard-coded. It should be read from the **ValidationMessages.properties** file

 - Once validated, invoke the appropriate service class method by passing the diagnosis to get the **List<PatientDTO>**
 - Return an object of **ResponseEntity** with the **List<PatientDTO>** object obtained in the previous step as a response and the HTTP response status as **OK**.

2. public ResponseEntity<PatientDTO> admitPatient (PatientDTO patientDTO)

- This method admits a patient in the HospitalManagement application
- It accepts **POST** request with URI **patients**
- It receives patient details in the form of JSON. The JSON data should be populated in **PatientDTO** parameter of this method after successful validation.
- The patient details will consist of the patient name, gender, date of birth, admission date and diagnosis. Using **Bean Validation API** validate the details according to the description given below (**Note:** The validation message should not be hard-coded. It should be read from the **ValidationMessages.properties** file)



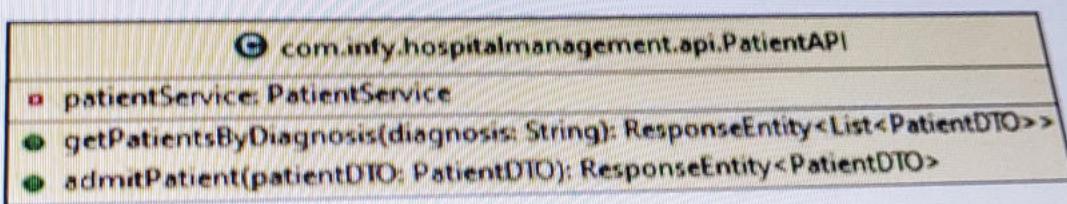
2. PatientDTO admitPatient(PatientDTO patientDTO)

- This method is used to admit a new patient details to the database
- Invoke the validatePatient() method of the PatientValidator class to validate the object of the PatientDTO class received as parameter
- Create and populate a new object of **Patient** with the appropriate values from the **PatientDTO** object
(Hint: You can use the prepareEntity() method of the PatientDTO class for this)
- Then invoke the appropriate method of the PatientRepository to add the **Patient** object to the database which will return the same object with the auto-generated **patientId**
- Set the **patientId** of the **Patient** object obtained in the previous step to the existing **PatientDTO** object
- Return the modified **PatientDTO** object

6. API Class

com.infy.hospitalmanagement.api.PatientAPI.java

Implement this class according to the class diagram and description given below:



- Annotate this class with appropriate annotation to declare it as REST controller class
- Annotate this class with appropriate annotation so that all its methods are mapped with **api** as base URL
- Annotate this class with appropriate annotation to enable the bean validation support
- Inject instance of PatientService class using appropriate annotation



ASSESSMENT INSTRUCTIONS PROJECT INSTRUCTIONS PROBLEM STATEMENT

5. Service Class

com.infy.hospitalmanagement.service.PatientServiceImpl.java

Implement this class according to the class diagram and the description given below:

com.infy.hospitalmanagement.service.PatientServiceImpl

- patientRepository: PatientRepository
- getListOfPatients(diagnosis: String): List<PatientDTO>
- admitPatient(patientDTO: PatientDTO): PatientDTO

- Inject the **PatientRepository** objects using the proper annotation
- Mark it with the appropriate stereotype annotation with the value **patientService**
- Mark this class with necessary annotation to manage transactions

Method Description:

1. List<PatientDTO> getListOfPatients(String diagnosis)

- This method is used to get all the patients having similar diagnosis
- Invoke the appropriate method of the PatientRepository interface which will return all the patients on the basis of their diagnosis
- If no patients details are available, throw an object of the **PatientAdmissionException** with the message **PatientService.PATIENT_UNAVAILABLE**
- Else, populate a list, **List<PatientDTO>** with all the patient details in the descending order of the admission date and return the sorted list

Hint: You can use the prepareDTO() method of the PatientDTO class to convert the respective objects to PatientDTO



1. void validatePatient(PatientDTO patientDTO)

This method receives an object of PatientDTO and calls the respective methods to validate the values and throws an object of PatientAdmissionException with the message, based on the violation, as given below,

Violation for	Message
dateOfBirth	PatientValidator.INVALID_DOB

2. Boolean IsValidDateOfBirth(LocalDate dateOfBirth)

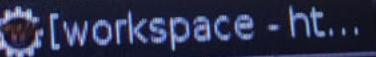
- This method validates the **dateOfBirth** received
- **dateOfBirth** should not be a future date.
- **dateOfBirth** should not be a date which is lesser than 100 years from today's date.
- If the above conditions fail then return false, else return true

4. Repository**com.infy.hospitalmanagement.repository.PatientRepository.java**

- This interface should extend the appropriate repository interface
- Add the following method to the interface
 - This method should accept the name of the diagnosis and return the list of patients having same value of **diagnosis**.

5. Service Class**com.infy.hospitalmanagement.service.PatientServiceImpl.java**

Implement this class according to the class diagram and the description given below:



2. DTO Class

com.infy.hospitalmanagement.dto.PatientDTO.java

This is a DTO (Data Transfer Object) class that holds patient details such as patient id, patient name, gender, date of birth, admission date and diagnosis.

- Use appropriate annotations from the Java Bean Validation API for validating patient details attributes of this class. (Refer below admitPatient() method of PatientAPI class for validation rules).

3. Validator Class

com.infy.hospitalmanagement.validator.PatientValidator.java

Implement this class according to the class diagram and the description given below:

G com.infy.hospitalmanagement.validator.PatientValidator
• validatePatient(patientDTO: PatientDTO): void
• isValidDateOfBirth(dateOfBirth: LocalDate): Boolean

Method Description:

1. void validatePatient(PatientDTO patientDTO)

This method receives an object of PatientDTO and calls the respective methods to validate the values and throws an object of PatientAdmissionException with the message, based on the violation, as shown below,

Violation for	Message
	[workspace - ht...]



1. Entity Classes

com.infy.hospitalmanagement.entity.Patient.java

- This class has been implemented for you. You have to map this class to the patient table with **patientId** as the **primary key**
- Generate the **patientId** using the **IDENTITY** strategy

com.infy.hospitalmanagement.entity.Patient

- patientId: Integer
- patientName: String
- gender: String
- dateOfBirth: LocalDate
- admissionDate: LocalDate
- diagnosis: String

- Patient()
- Patient(patientId: Integer, patientName: String, gender: String, dateOfBirth: LocalDate, admissionDate: LocalDate, diagnosis: String)
- getPatientId(): Integer
- setPatientId(patientId: Integer): void
- getPatientName(): String
- setPatientName(patientName: String): void
- getGender(): String
- setGender(gender: String): void
- getDateOfBirth(): LocalDate
- setDateOfBirth(dateOfBirth: LocalDate): void
- getAdmissionDate(): LocalDate
- setAdmissionDate(admissionDate: LocalDate): void
- getDiagnosis(): String
- setDiagnosis(diagnosis: String): void

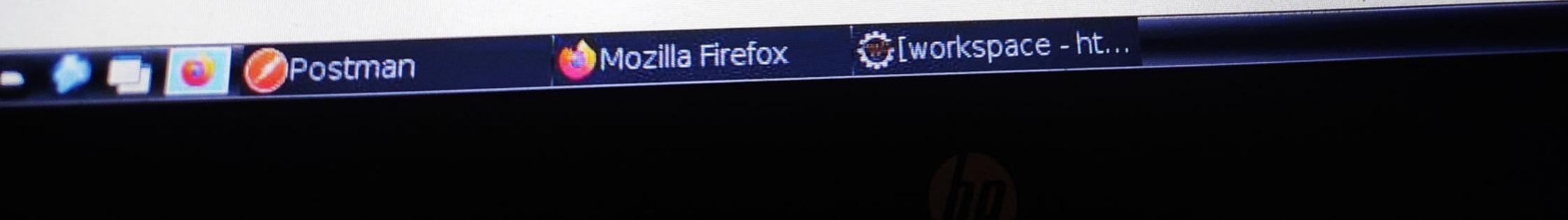


Postman

Mozilla Firefox

[workspace - ht...]

File Name	Status
application.properties	
com.infy.hospitalmanagement.HospitalManagementApplication.java	
com.Infy.hospitalmanagement.exception.PatientAdmissionException.java	
com.infy.hospitalmanagement.service.PatientService.java	
com.infy.hospitalmanagement.utility.ErrorInfo.java	
log4j2.properties	Implemented
HospitalManagement_TableScripts.sql	
ValidationMessages.properties	
com.infy.hospitalmanagement.dto.PatientDTO.java	
com.infy.hospitalmanagement.entity.Patient.java	To be modified
com.infy.hospitalmanagement.utility.ExceptionControllerAdvice.java	
com.infy.hospitalmanagement.api.PatientAPI.java	
com.infy.hospitalmanagement.repository.PatientRepository.java	
com.infy.hospitalmanagement.service.PatientServiceImpl.java	To be implemented
com.infy.hospitalmanagement.validator.PatientValidator.java	



Infosys Certified Spring Associate Hands-On Assessment

Problem Statement:

In this assessment, you are going to build an **Hospital Management** application which can be used to admit a patient and view all the patients of a given diagnosis. You will have to implement the following functionalities in the application,

1. Admit a new patient
2. Get all patients from a diagnosis

Project Structure provided:

File Name	Status
application.properties	
com.infy.hospitalmanagement.HospitalManagementApplication.java	
com.infy.hospitalmanagement.exception.PatientAdmissionException.java	
com.infy.hospitalmanagement.service.PatientService.java	



Assumptions:

1. All the necessary imports and configurations are done
2. Port number used is 8765

What will be the output when a HTTP POST request is made to the above controller with url as <http://localhost:8765/infy> and below data?

```
{  
    "employeeId": 1,  
    "emailId": "andrew@infy.com",  
    "employeeName": "Andrew Wyatt",  
    "designation": "Senior Associate"  
}
```

- EmployeeDTO [employeeId=null, emailId=null, name=null, designation=null]
- EmployeeDTO [employeeId=1, emailId=andrew@infy.com, name=null, designation=Senior Associate] 
- EmployeeDTO [employeeId=1, emailId= andrew@infy.com, name=Andrew Wyatt, designation= Senior Associate]
- 404 NOT_FOUND error

Reset

Save



- 2
- 3
- 5
- 6
- 8
- 9

Question 10

Consider the following EmployeeDTO and EmployeeAPI classes:

```
public class EmployeeDTO {  
    private Integer employeeId;  
    private String emailId;  
    private String name;  
    private String designation;  
  
    //getters, setters, toString  
}  
  
@RestController  
@RequestMapping(value = "/infy")  
public class EmployeeAPI {  
    @PostMapping  
    public void addEmployee(EmployeeDTO employeeDTO) {  
        System.out.println(employeeDTO);  
    }  
}
```

Assumptions:

1. All the necessary imports and configurations are done
2. Port number used is 8765

What will be the output when a HTTP POST request is made to the above controller with url as <http://localhost:8765/infy>

1 2 3
4 5 6
8 9

cust_id	sequence_name
27-NOV-2022	ratnadeep.lande

Assumptions:

1. All necessary configurations are done
2. MySQL database is used

What will be value of the primary key generated when a new customer is added?

- 1001
- 1000
- 2000
- 1

Reset

Save



Consider the below Entity class and tables:

```
@Entity
public class Customer{
    @Id
    @GeneratedValue
    private Integer customerId;
    //rest of the code
}
```

CUSTOMER (customer_id is primary key):

customer_id	email_id	name	date_of_birth
1999	martin@infy.com	Martin	1993-10-02

HIBERNATE_SEQUENCE(sequence_name is primary key):

sequence_name	next_val
cust_id	1000

Assumptions:

You're being p



```
private Integer patientId;  
private String patientName;  
private Long patientPhoneNumber;  
// getters and setters
```

Assumption:

1. All necessary imports and configurations are done

What will be the value of pat_id in id_gen table after persisting details of three patients in PATIENT table?

- 110
- 117
- 103
- 115

Reset

Save



Consider the below Patient entity class where TABLE strategy is used to generate primary key values.

```
@Entity  
public class Patient {  
    @Id  
    @TableGenerator(  
        name="pkgen",  
        table="id_gen",  
        pkColumnName="gen_key",  
        valueColumnName="gen_value",  
        pkColumnValue="pat_id",  
        allocationSize=1)  
    @GeneratedValue(generator="pkgen", strategy=GenerationType.TABLE)  
    private Integer patientId;  
    private String patientName;  
    private Long patientPhoneNumber;  
    // getters and setters  
}
```

Assumption:

1. All necessary imports and configurations are done

What will be the value of pat_id in id_gen table after persisting details of three patients in PATIENT table?

2

3

5

6

8

9

[1 Mark]

ratnadeep.landy~
27-Nov-2022

Consider the following PATIENT table and id_gen table as given below:

PATIENT

patient_id	name	contact_number
51001	Peter	8907645284

id_gen



gen_key

gen_value

pat_id

100

Consider the below Patient entity class where TABLE strategy is used to generate primary key values.

```
@Entity  
public class Patient {
```

Focus Area 2

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9

- ratnadeep.landy~
27-Nov-2022
- Before advice is called
In display method
 After advice is called
- Before advice is called
 In display method
- In display method
 After advice is called
- Before advice is called
After advice is called
 In display method

Reset

Save

Focus Area 2

1

2

3

4

5

6

7

8

Assumption:

1. All necessary imports and configurations are done.

What is the output when the below code is executed?

```
package com.infy;
@SpringBootApplication
public class SpringBootApplication implements CommandLineRunner {

    public static void main(String[] args) {
        SpringApplication.run(SpringBootApplication.class, args);
    }

    public void run(String... args) throws Exception {
        display();
    }

    public void display() {
        System.out.println("In display method");
    }
}
```

In display method

- 2
- 3
- 5
- 6
- 8
- 9

[1 Mark]

Question 7

Consider the following LoggingAspect class:

```
package com.infy.utility;
@Component
@Aspect
public class LoggingAspect {

    @Before("execution(* com.infy.SpringApplication.*(..))")
    public void before(){
        System.out.println("Before advice is called");
    }

    @After("execution(* com.infy.*Application.*(..))")
    public void after(){
        System.out.println("After advice is called");
    }
}
```

Assumption:

1. All necessary imports and configurations are done.

What is the output when the below code is executed?

What should be inserted at Line 1 so that it creates a correct named query for Order using following repository?

```
public interface OrderRepository extends CrudRepository<Order, Integer> {
    String findNameByCustomerEmailId(@Param("emailId") String emailId);
}
```

Assumption:

1. All necessary configurations and imports are done

- @NamedQuery(name="Order.findNameByCustomerEmailId", query="SELECT o.customerName FROM Order o WHERE p.customerEmailId = :emailId")
- @NamedQuery(name="Order.findNameByCustomerEmailId", query="SELECT o.customername FROM Order o WHERE p.emailId = :emailId")
- @Query(name="Order.findNameByCustomerEmailId", query="SELECT o.customername FROM Order o WHERE p.customerEmailId = :emailId")
- @Query(name="Order.findNameByEmailId", query="SELECT o.customername FROM Order o WHERE p.customerEmailId = :emailId")

Reset

Save



Focus Area 2

- 1
- 2
- 3
- 4
- 5
- 6
- 8
- 9

[2 Marks]

ratnadeep.lande~
27-NOV-2022

Consider the following entity class:

```
@Entity  
//Line 1  
public class Order {  
    @Id  
    private Integer orderId;  
    private String customerName;  
    private String customerEmailId;  
    //getter and setter methods  
}
```

Question 6

ratnadeep.lande~
27-NOV-2022

43%

What should be inserted at Line 1 so that it creates a correct named query for Order using following repository?

```
public interface OrderRepository extends CrudRepository<Order, Integer> {  
    String findNameByCustomerEmailId(@Param("emailId") String emailId);  
}
```

Assumption:

1. All necessary configurations and imports are done



2 3
5 6
8 9

```
public void main(String[] args) {  
    SpringApplication.run(DemoApplication.class, args);  
    System.out.println("I am a Spring Boot Application");  
}  
  
@Override  
public void run(String... args) throws Exception {  
    System.out.println(environment.getProperty("welcome")); //Line1  
    System.out.println(environment.getProperty("hello"));  
}
```

- Welcome to Spring Boot
- Hello World
- I am a Spring Boot Application
- Welcome to DemoApplication
- Hello World
- I am a Spring Boot Application
- Welcome to DemoApplication
- Hello World
- Exception generated at Line 1 due to ambiguity in the key "welcome"



[2 Marks]

Consider the following properties files present in classpath of the Spring Boot application:

```
application.properties  
welcome=Welcome to Spring Boot  
  
messages.properties  
welcome=Welcome to DemoApplication  
hello=Hello World
```

Assumption:

1. All necessary imports and configurations are done

What will be the output when following code is executed?

```
@SpringBootApplication  
 @PropertySource("classpath:messages.properties")  
 public class DemoApplication implements CommandLineRunner {  
     @Autowired  
     Environment environment;  
     public static void main(String[] args) {  
         SpringApplication.run(DemoApplication.class, args);  
         System.out.println("I am a Spring Boot Application");  
    }  
}
```



Consider the following class:

```
class Account {
    double balance;
    void withdraw(double amount) {
        if (amount > balance)
            throw new InsufficientBalanceException("Insufficient balance");
        else
            balance -= amount;
    }
}
```

Question 4

Which annotation can be inserted at Line1 and Line2 so that `InsufficientBalanceException` class can handle the appropriate exceptions thrown from the application?

- Line1 - `@WebService`
Line2 - `@WebMethod(operationName = "withdraw")`
- Line1 - `@WebService`
Line2 - `@WebFault(name = "InsufficientBalanceException")`
- Line1 - `@WebService`
Line2 - `@WebFault(name = "InsufficientBalanceException")`
- Line1 - `@WebService`
Line2 - `@WebFault(name = "InsufficientBalanceException")`

Question 3

53%

Consider the following controller class:

```
@RestController  
@RequestMapping(value = "/api")  
public class ProductAPI {  
    @PutMapping(value = "/products")  
    public String updateDetails(@RequestParam Integer id, @RequestParam String name) {  
        //rest of the code  
    }  
}
```

Which of the following HTTP PUT requests are mapped to above controller? Note: Consider all necessary imports are added.

- http://localhost:8080/api/products?id=1001?name=Laptop
- http://localhost:8080/api/products/1001/Laptop
- http://localhost:8080/api/products?id=1001&name=Laptop
- http://localhost:8080/api/products/id=1001/name=Laptop

Reset

Save

Assumptions:

1. All necessary imports and configurations are done
2. restTemplate is a reference of RestTemplate class
3. 8765 is the port number used

Which of the following code snippet can be used to hit the getUsers() method in above controller to get the below response?

```
[{userId=101, userName=Nick Eriksen, contactNumber=894576132}]
```

- String url = "http://localhost:8765/users/users";
List<UserDTO> userDTOs = restTemplate.getForObject(url, List<UserDTO>.class);
- String url = "http://localhost:8765/users/users";
List<UserDTO> userDTOs = restTemplate.getForObject(url, List.class);
- String url = "http://localhost:8765/users";
List<UserDTO> userDTOs = restTemplate.getForObject(url, List.class);
- String url = "http://localhost:8765/users/users";
UserDTO userDTOs = restTemplate.getForObject(url, UserDTO.class);

Reset

Save

Question 2

Consider the following DTO and REST API classes:

```
public class UserDTO {  
    private Integer userId;  
    private String userName;  
    private String contactNumber;  
    //parameterized constructor  
    //getters, setters, toString method  
}  
  
@RestController  
@RequestMapping("/users")  
public class UserAPI {  
    @GetMapping("/users")  
    public ResponseEntity<List<UserDTO>> getUsers(){  
        List<UserDTO> userDTOs = new ArrayList<UserDTO>();  
        UserDTO userDTO = new UserDTO(101, "Nick Eriksen", "894576132");  
        userDTOs.add(userDTO);  
        return new ResponseEntity<>(userDTOs, HttpStatus.OK);  
    }  
}
```

Assumptions:

15%

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

398

399

400

401

402

403

404

405

406

407

408

409

410

411

412

413

414

415

416

417

418

419

420

421

422

423

424

425

426

427

428

429

430

431

432

433

434

435

436

437

438

439

440

441

442

443

444

445

446

447

448

449

450

451

452

453

454

455

456

457

458

459

460

461

462

463

464

465

466

467

468

469

470

471

472

473

474

475

476

477

478

479

480

481

482

483

484

485

486

487

488

489

490

491

492

493

494

495

496

497

498

499

500

501

502

503

504

505

506

507

508

509

510

511

512

513

514

515

516

517

518

519

520

521

522

523

524

525

526

527

528

529

530

531

532

533

534

535

536

537

538

539

540

541

542

543

544

545

546

547

548

549

550

551

552

553

554

555

556

557

558

559

560

561

562

563

564

565

566

567

568

569

570

571

572

573

574

575

576

577

578

579

580

581

582

583

584

585

586

587

588

589

590

591

592

593

594

595

596

597

598

599

600

601

602

603

604

605

606

607

608

609

610

611

612

613

614

615

616

617

618

619

620

621

622

623

624

625

626

627

628

629

630

631

632

633

634

635

636

637

638

639

640

641

642

643

644

645

646

647

648

649

650

651

652

653

654

655

656

657

658

659

660

661

662

663

664

665

666

667

668

669

660

661

662

663

664

665

666

667

668

669

670

671

672

673

674

675

676

677

678

679

680

681

682

683

684

685

686

687

688

689

690

691

692

693

694

695

696

697

698

699

700

701

702

703

704

705

706

707

708

709

710

711

712

713

714

715

716

717

718

719

720

721

722

723

724

725

726

727

728

729

720

721

722

723

724

725

726

727

728

729

730

731

732

733

734

735

736

737

738

739

730

731

732

733

734

735

736

737

738

739

740

741

742

743

744

745

746

747

748

749

740

741

742

743

744

745

746

747

748

749

750

751

752

753

754

755

756

757

758

759

750

751

752

753

754

755

756

757

758

759

760

761

762

763

764

765

766

767

768

769

760

761

762

763

764

765

766

767

768

769

770

771

772

773

774

775

776

777

778

779

770

771

772

773

774

775

776

777

778

779

780

781

782

783

784

785

786

787

788

789

780

781

782

783

784

785

786

787

788

789

790

791

792

793

794

795

796

797

798

799

790

791

792

793

794

795

796

797

798

799

800

801

802

803

804

805

806

807

808

809

800

801

802

803

804

805

806

807

808

809

810

811

812

813

814

815

816

817

818

819

810

811

812

813

814

815

816

817

818

819

820

821

822

823

824

825

826

827

828

829

820

821

822

823

824

825

826

827

828

829

830

831

832

833

834

835

836

837

838

839

830

831

832

833

834

835

836

837

838

839

840

841

842

843

844

845

846

847

848

849

840

841

842

843

844

845

846

847

848

849

850

851

852

853

854

855

856

857

858

859

850

851

852

853

854

855

856

857

858

859

860

861

862

863

864

865

866

867

868

869

860

861

862

863

864

865

866

867

868

869

870

871

872

873

874

875

876

877

878

879

870

871

872

873

874

875

876

877

878

879

880

881

882

883

884

885

886

887

888

889

880

881

882

883

884

885

886

887

888

889

890

891

892

893

894

895

896

897

898

899

890

891

892

893

894

895

896

897

898

899

900

901

902

903

904

905

906

907

908

909

900

901

902

903

904

905

906

907

908

909

910

911

912

913

914

915

916

917

918

919

910

911

912

913

914

915

916

917

918

919

920

921

922

923

924

925

926

927

928

929

920

921

922

923

924

925

926

927

928

929

930

931

932

933

934

935

936

937

938

939

930

931

932

933

934

935

936

937

938

939

940

941

942

943

944

945

946

947

948

949

940

941

942

943

944

945

946

947

948

949

950

951

952

953

954

955

956

957

958

959

950

951

952

953

954

955

956

957

958

959

960

961

962

963

964

965

966

967

968

969

960

961

962

963

964

965

966

967

968

969

970

971

972

973

974

975

976

977

978

979

970

971

972

973

974

975

976

977

978

979

980

981

982

983

984

985

986

987

988

989

980

981

982

983

984

985

986

987

988

989

990

991

992

993

994

995

996

997

998

999

990

991

992

993

994

995

996

997

998

999

1000

1001

1002

1003

1004

1005

1006

1007

1008

1009

1000

1001

1002

1003

1004

1005

1006

1007

1008

1009

1010

1011

1012

1013

1014

1015

1016

1017

1018

1019

1010

1011

1012

1013

1014

1015

1016

1017

1018

1019

1020

1021

1022

1023

1024

1025

1026

1027

1028

1029

1020

1021

1022

1023

1024

1025

1026

1027

1028

1029

1030

1031

1032

1033

1034

1035

1036

1037

1038

1039

1030

1031

1032

1033

1034

1035

1036

1037

1038

1039</

```
list = new ArrayList<>();
list.add(book);
author.setBooks(list);
authorRepository.save(author);
```

AUTHOR:

author_id	name
1001	James
1002	Tom

BOOK:

book_id	book_name	auth_id
2001	Right Thing	1001
2002	Wonderland	1002
2003	Potter Bloom	1002
2004	Key To Success	1002

AUTHOR:

author_id	name
1001	James
1002	Tom

BOOK:

```
@Entity  
public class Author {  
    @Id  
    private Integer authorId;  
    private String name;  
    @OneToMany(cascade=CascadeType.ALL)  
    @JoinColumn(name="auth_id")  
    private List<Book> books;  
    //getters and setters  
}
```

Assumptions:

1. All necessary configurations and imports are done
2. authorRepository is a valid reference of AuthorRepository which extends CrudRepository interface
3. Transaction is managed by Spring

What will be the state of both the tables after the below code is executed?

```
Author author = authorRepository.findById(1002);  
Book book = new Book();  
book.setBookId(2004);  
book.setBookName("Key To Success");  
List<Book> list = new ArrayList<>();  
list.add(book);  
author.setBooks(list);  
authorRepository.save(author);
```

[2 Marks]

Consider the following tables and corresponding entity classes:

AUTHOR (author_id is primary key):

author_id
1001
1002

name
James
Tom

BOOK (book_id is primary key and auth_id is foreign key referring to author_id of AUTHOR table):

book_id	book_name	auth_id
2001	Right Thing	1001
2002	Wonderland	1002
2003	Potter Bloom	1002

```
@Entity  
public class Book{  
    @Id  
    private Integer bookId  
    private String bookName;  
    //getters and setters  
}
```

```
public class Tester2 {  
    public static void main(String a[]) {  
        List<Customer> list = new ArrayList<Customer>();  
        Customer c1 = new Customer(9001, "Patrick");  
        Customer c2 = new Customer(9002, "Mary");  
        Customer c3 = new Customer(9003, "Anupam");  
        list.add(c1);  
        list.add(c2);  
        list.add(c3);  
        Stream<Customer> stream = list.stream().filter(cust -> cust.getCustomerName().length() >= 5);  
        stream.forEach((customer) -> System.out.print(customer.getCustomerId() + " "));  
    }  
}
```

- 9001
- 9001 9003
- 9001 9002 9003
- 9002 9003

Reset

Save

[2 Marks]

What will be the output of the below code? Assume **Customer** class exists as below

```
class Customer{  
    int customerId;  
    String customerName;  
    Customer(int cId, String name){  
        this.customerId=cId;  
        this.customerName=name;  
    }  
    //Getters and setters  
}  
  
public class Tester2 {  
    public static void main(String a[]){  
        List<Customer> list = new ArrayList<Customer>();  
        Customer c1 = new Customer(9001, "Patrick");  
        Customer c2 = new Customer(9002, "Mary");  
        Customer c3 = new Customer(9003, "Anupam");  
        list.add(c1);  
        list.add(c2);  
        list.add(c3);  
        Stream<Customer> stream = list.stream().filter(cust -> cust.getCustomerName().length() >= 5);  
        stream.forEach((customer) -> System.out.print(customer.getCustomerId() + " "));  
    }  
}
```

```
        return true;
    return false;
}
```

Following is the test case written for testing the above validatePassword() method:

```
public class ValidatorClassTest{
    @Test
    public void validatePasswordTest(){
        // Line1
    }
}
```

Which of the following assertion statements would be appropriate at 'Line1' for valid test cases. Choose TWO correct options.

- Assertions.assertEquals(Boolean.TRUE,Validator.validatePassword("Abc123"));
- Assertions.assertTrue(Validator.validatePassword("Abc"));
- Assertions.assertEquals(TRUE,Validator.validatePassword("Abc123"));
- Assertions.assertTrue(Validator.validatePassword("Abc1"));

Reset

Save

Question 4

Given is a Validator class to test a password:

```
public class Validator{  
    public static Boolean validatePassword(String password){  
        if(password.matches(".*[A-Z].*")&&password.matches(".*[0-9].*"))  
            return true;  
        return false;  
    }  
}
```

Following is the test case written for testing the above validatePassword() method:

```
public class ValidatorClassTest{  
    @Test  
    public void validatePasswordTest(){  
        // Line1  
    }  
}
```

Which of the following assertion statements would be appropriate at 'Line1' for valid test cases. Choose TWO correct options.

- Assertions.assertEquals(Boolean.TRUE,Validator.validatePassword("Abc123"));



Question 4

Given is a Validator class to test a password:

```
public class Validator{  
    public static Boolean validatePassword(String password){  
        if(password.matches("[A-Z]+")&&password.matches("[0-9]+"))  
            return true;  
        return false;  
    }  
}
```

Following is the test case written for testing the above validatePassword() method:

```
public class ValidatorClassTest{  
    @Test  
    public void validatePasswordTest(){  
        // Line1  
    }  
}
```

Which of the following assertion statements would be appropriate at 'Line1' for valid test cases. Choose TWO correct options.

- Assertions.assertEquals(Boolean.TRUE,Validator.validatePassword("Abc123"));

```
public static void main(String[] args) {  
    Map<String, String> nameMap = new HashMap<String, String>();  
  
    nameMap.put("A", "Andre");  
    nameMap.put("B", "Bob");  
    nameMap.put(null, null);  
    nameMap.put("C", "Catlyn");  
    nameMap.put(new String("A"), "Avan");  
    System.out.println(nameMap);  
}
```

Assumption:

1. All necessary configuration and imports are done

- {null=null, A=Avan, B=Bob, C=Catlyn}
- {A=Avan, B=Bob, C=Catlyn}
- The code will throw run time exception as the same key cannot be used for 2 values.
- The code will throw NullPointerException as the HashMap cannot have null as key value.

Reset

Save

Question 3

What will be the output when following code is executed?

```
public static void main(String[] args) {  
    Map<String, String> nameMap = new HashMap<String, String>();  
  
    nameMap.put("A", "Andre");  
    nameMap.put("B", "Bob");  
    nameMap.put(null, null);  
    nameMap.put("C", "Catlyn");  
    nameMap.put(new String("A"), "Avan");  
    System.out.println(nameMap);  
}
```

Assumption:

1. All necessary configuration and imports are done

- {null=null, A=Avan, B=Bob, C=Catlyn}
- {A=Avan, B=Bob, C=Catlyn}
- The code will throw run time exception as the same key cannot be used for 2 values.

What is the output of the following code?

```
class Customer {  
    private String custName;  
    Customer(String fName) {  
        this.custName = fName;  
    }  
    public String getCustName() {  
        return custName;  
    }  
    public String toString() {  
        return custName;  
    }  
}  
  
public class Tester1 {  
    public static void main(String[] args) {  
        Customer cust1 = new Customer("John");  
        Customer cust2 = new Customer("Joe");  
        Customer cust3 = new Customer("Sam");  
        List<Customer> list = new ArrayList<Customer>();  
        list.add(cust1);  
        list.add(cust2);  
        list.add(cust3);  
        System.out.print("Before: " + list);  
        for (int j = 0; j < list.size(); j++) {  
            Customer e = list.get(j);  
        }  
    }  
}
```



2

3

5

```
List<Customer> list = new ArrayList<Customer>();
list.add(cust1);
list.add(cust2);
list.add(cust3);
System.out.print("Before: " + list);
for (int j = 0; j < list.size(); j++) {
    Customer e = list.get(j);
    if (e.getCustName().equals(cust1.getCustName()))
        list.remove(j+1);
}
System.out.println(" After: " + list);
```

- Before: [John, Joe, Sam] After: [John, Sam]
- Before: [John, Joe] After: [John]
- Before: [John, Sam] After: [Sam]
- Before: [John, Sam] After: [Joe]

Reset

Save

2

3

5

What is the output of the following code?

```
class Customer {  
    private String custName;  
    Customer(String fName) {  
        this.custName = fName;  
    }  
    public String getCustName() {  
        return custName;  
    }  
    public String toString() {  
        return custName;  
    }  
}  
  
public class Tester1 {  
    public static void main(String[] args) {  
        Customer cust1 = new Customer("John");  
        Customer cust2 = new Customer("Joe");  
        Customer cust3 = new Customer("Sam");  
        List<Customer> list = new ArrayList<Customer>();  
        list.add(cust1);  
        list.add(cust2);  
        list.add(cust3);  
        System.out.print("Before: " + list);  
        for (int j = 0; j < list.size(); j++) {  
            Customer e = list.get(j);  
        }  
    }  
}
```



[1 Mark]

ratnadeep.landy~
27-Nov-2022

Question 1

74%

Assuming the below code executes in the main method. What will be the output of the following code?

```
LocalDate localDate = LocalDate.of(2012, 12, 31);
localDate=localDate.plus(2, ChronoUnit.MONTHS);
System.out.println(localDate);
```

- 2013-03-02
- 2013-02-28
- 2013-03-01
- Error: Cannot add month

ResetSave