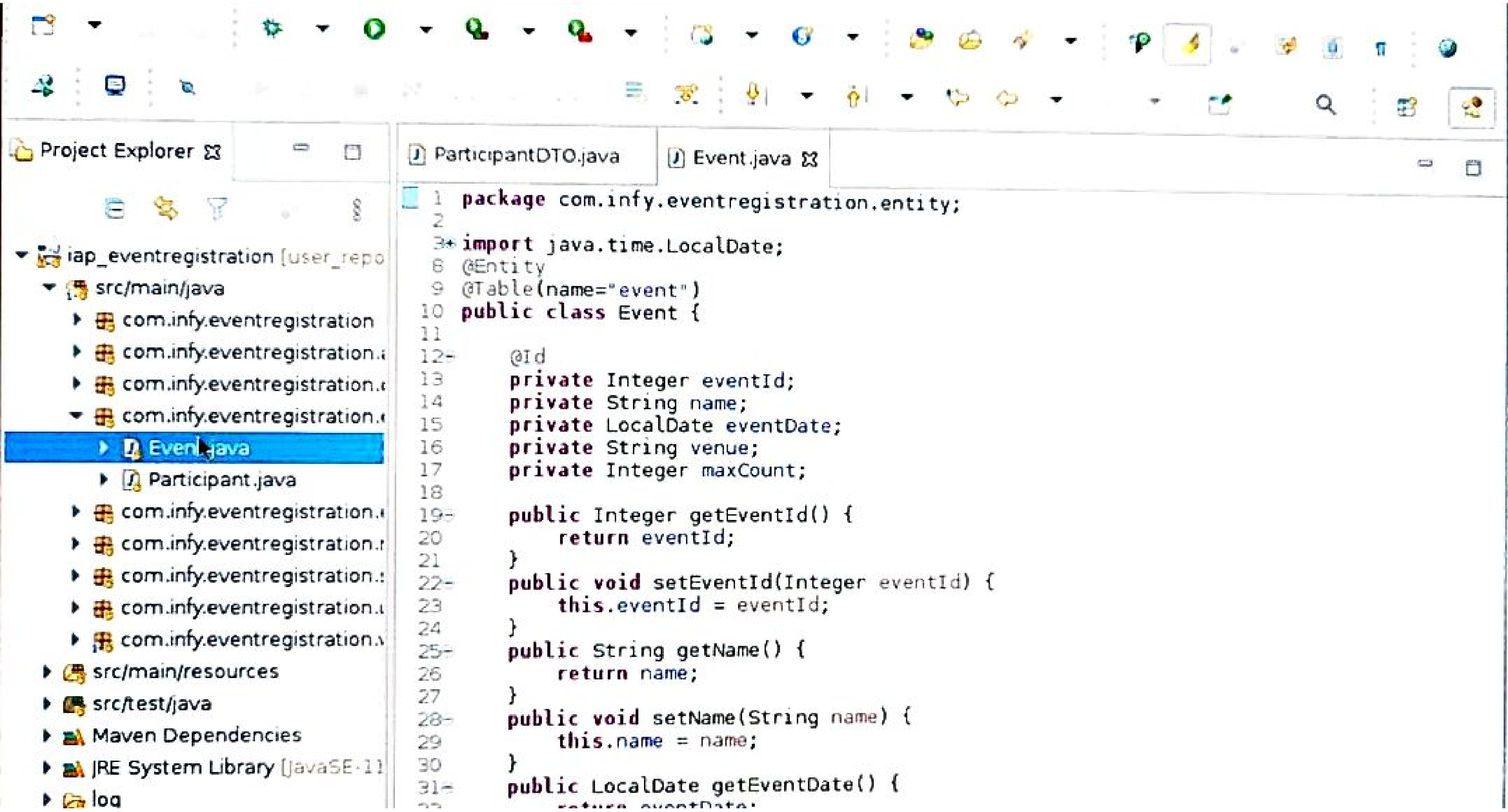
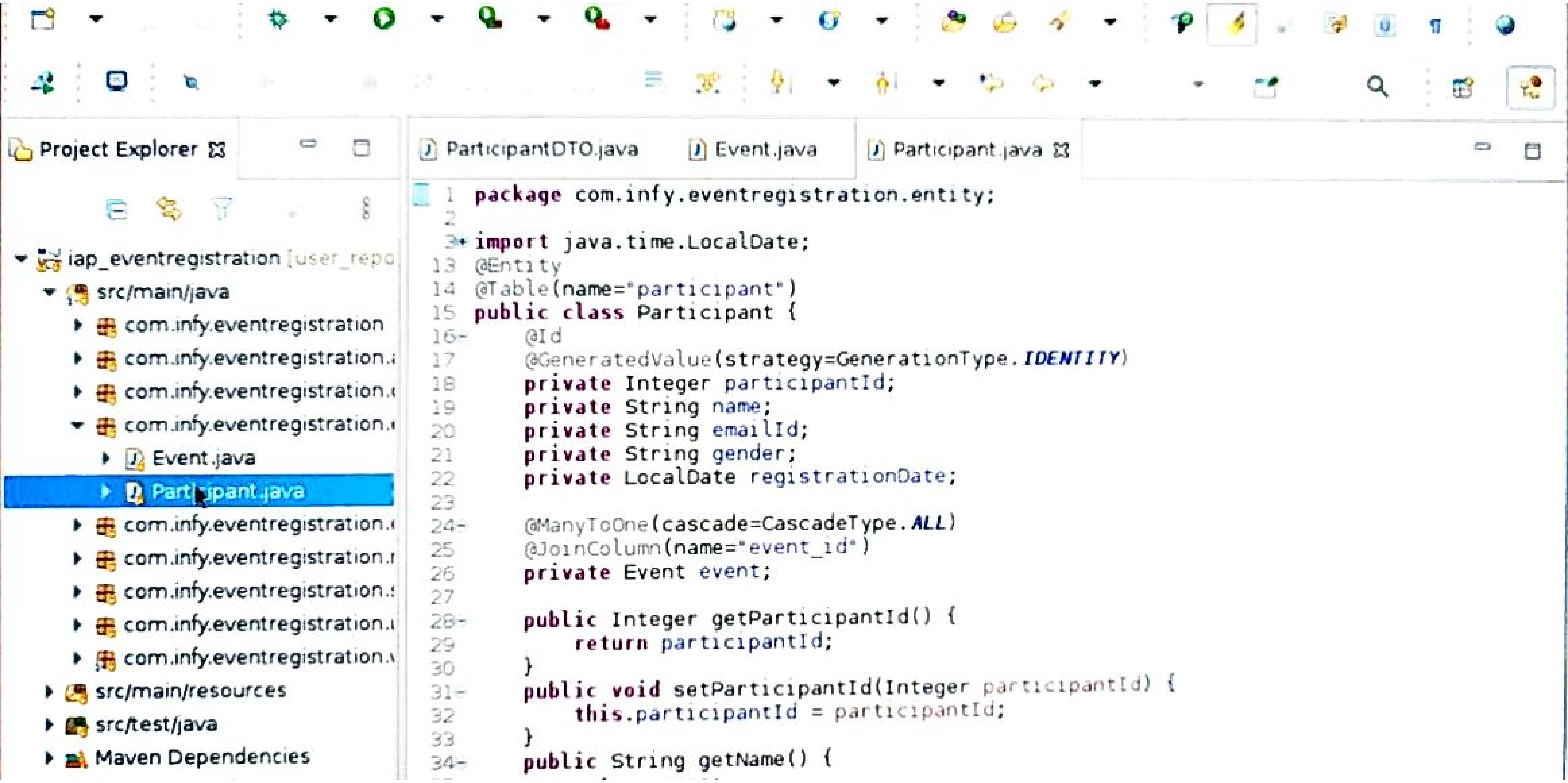
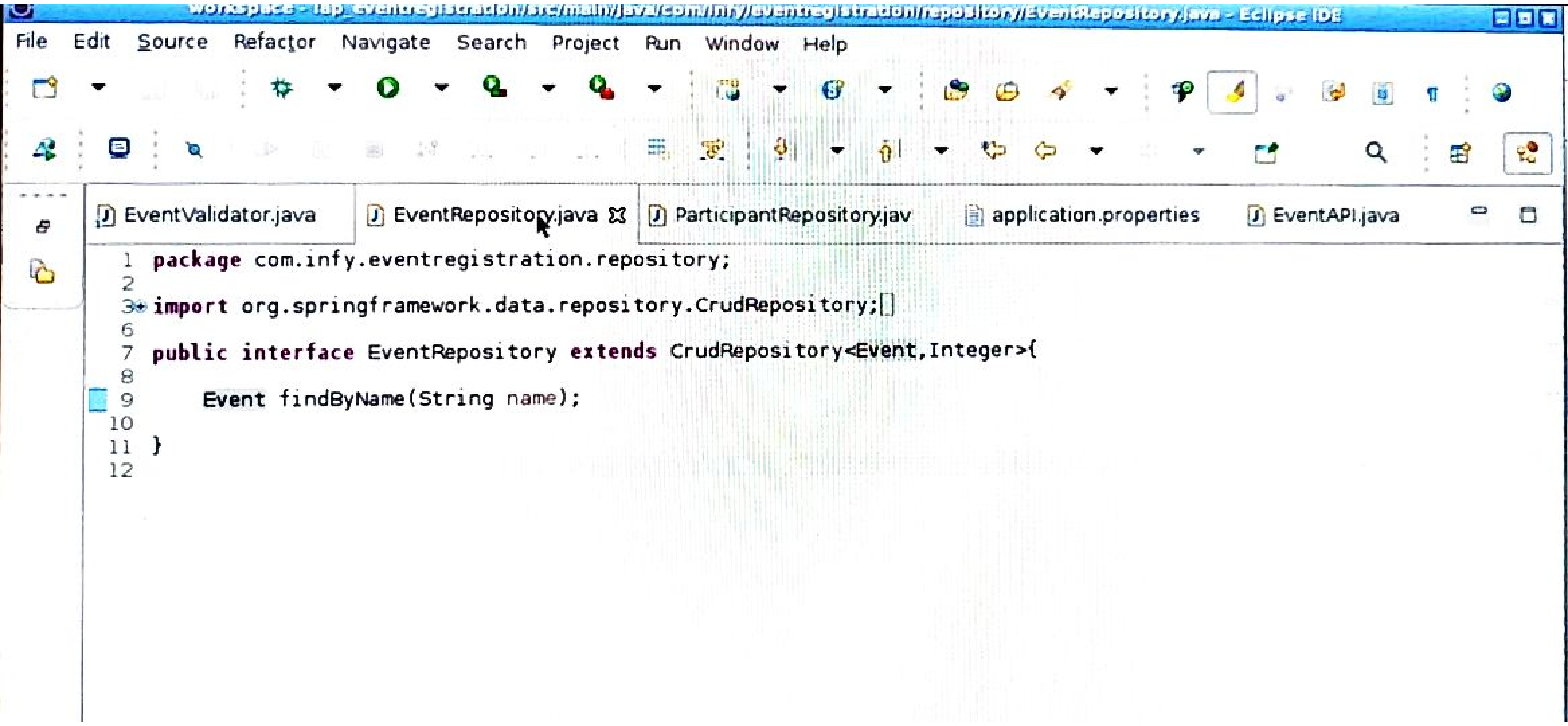

Project Explorer
iap_eventregistration [user_repo
src/main/java
com.infy.eventregistration
com.infy.eventregistration.
com.infy.eventregistration.
com.infy.eventregistration.
Event.java
Participant.java
com.infy.eventregistration.
com.infy.eventregistration.
com.infy.eventregistration.
com.infy.eventregistration.
com.infy.eventregistration.
src/main/resources
src/test/java
Maven Dependencies
JRE System Library [JavaSE-11]
log
ParticipantDTO.java
Event.java

```java
package com.infy.eventregistration.entity;

import java.time.LocalDate;
@Entity
@Table(name="event")
public class Event {

    @Id
    private Integer eventId;
    private String name;
    private LocalDate eventDate;
    private String venue;
    private Integer maxCount;

    public Integer getEventId() {
        return eventId;
    }
    public void setEventId(Integer eventId) {
        this.eventId = eventId;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public LocalDate getEventDate() {
```

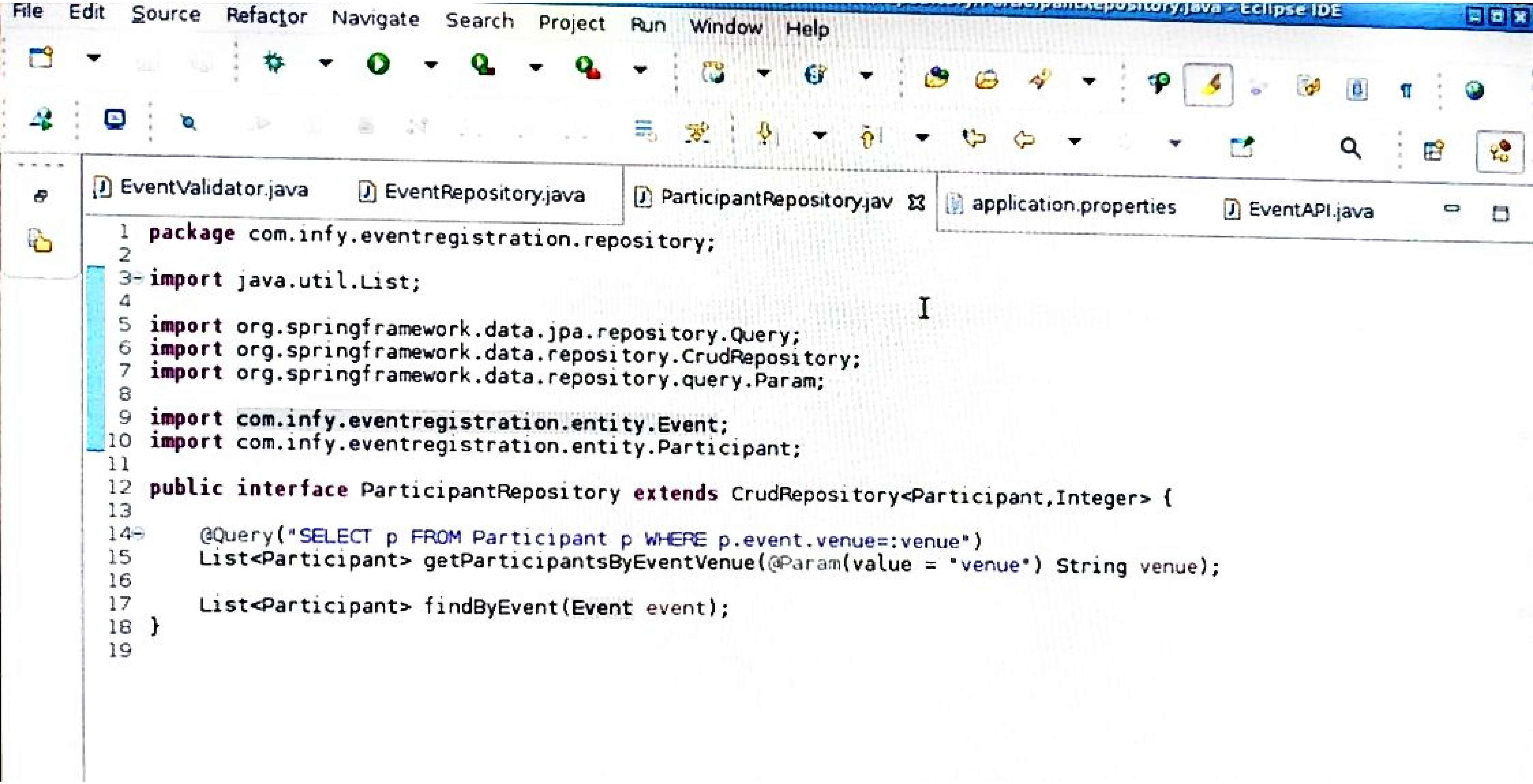


Project Explorer
iap_eventregistration [user_repo
src/main/java
com.infy.eventregistration
com.infy.eventregistration.
com.infy.eventregistration.
com.infy.eventregistration.
Event.java
Participant.java
com.infy.eventregistration.
com.infy.eventregistration.
com.infy.eventregistration.
com.infy.eventregistration.
com.infy.eventregistration.
src/main/resources
src/test/java
Maven Dependencies
ParticipantDTO.java
Event.java
Participant.java

```java
package com.infy.eventregistration.entity;

import java.time.LocalDate;
@Entity
@Table(name="participant")
public class Participant {
    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    private Integer participantId;
    private String name;
    private String emailId;
    private String gender;
    private LocalDate registrationDate;

    @ManyToOne(cascade=CascadeType.ALL)
    @JoinColumn(name="event_id")
    private Event event;

    public Integer getParticipantId() {
        return participantId;
    }
    public void setParticipantId(Integer participantId) {
        this.participantId = participantId;
    }
    public String getName() {
```

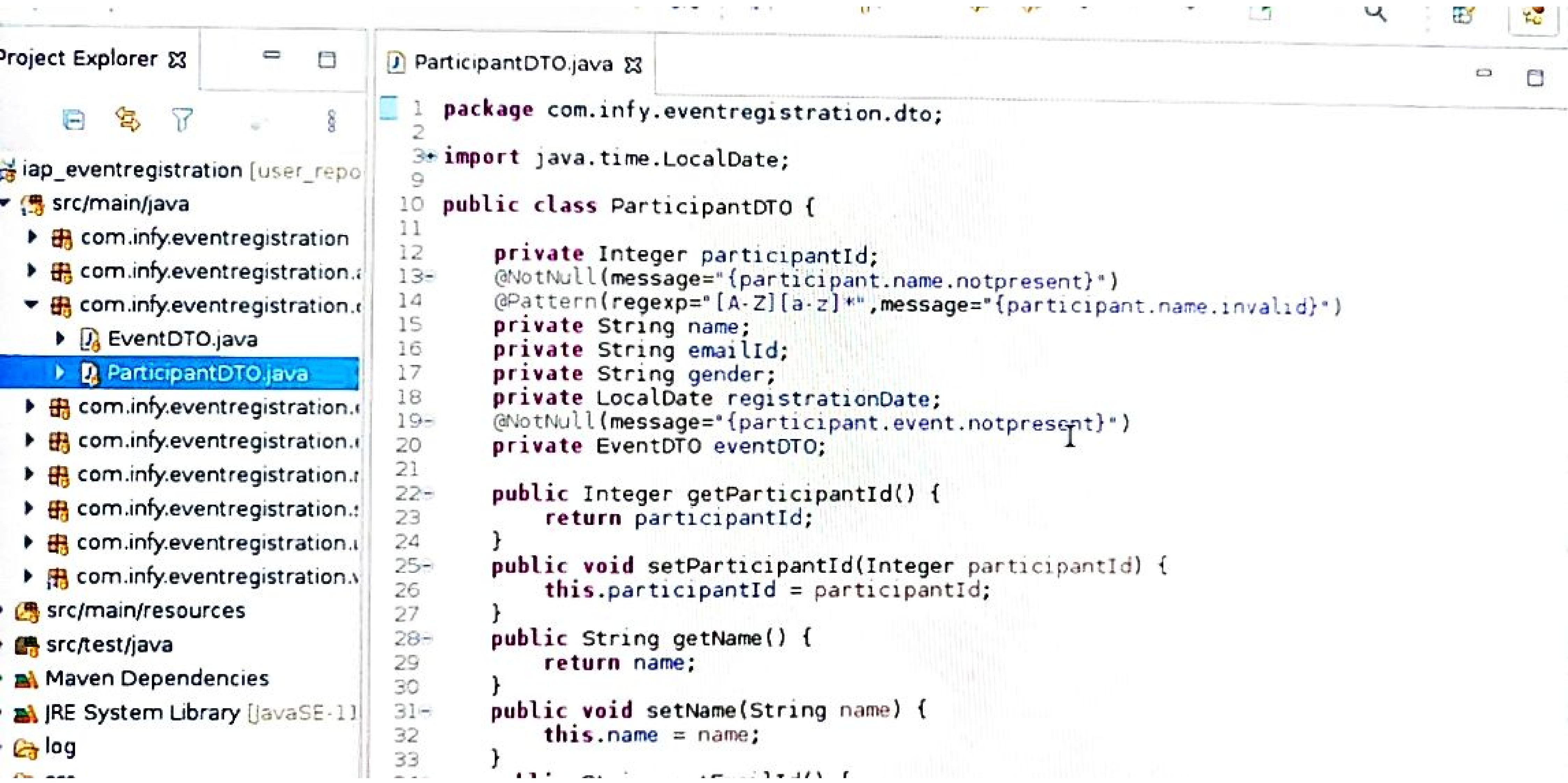

workspace - fap_eventregistration/src/main/java/com/infy/eventregistration/repository/EventRepository.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
EventValidator.java
EventRepository.java
ParticipantRepository.jav
application.properties
EventAPI.java

```
package com.infy.eventregistration.repository;

import org.springframework.data.repository.CrudRepository;

public interface EventRepository extends CrudRepository<Event,Integer>{

    Event findByName(String name);

}

```

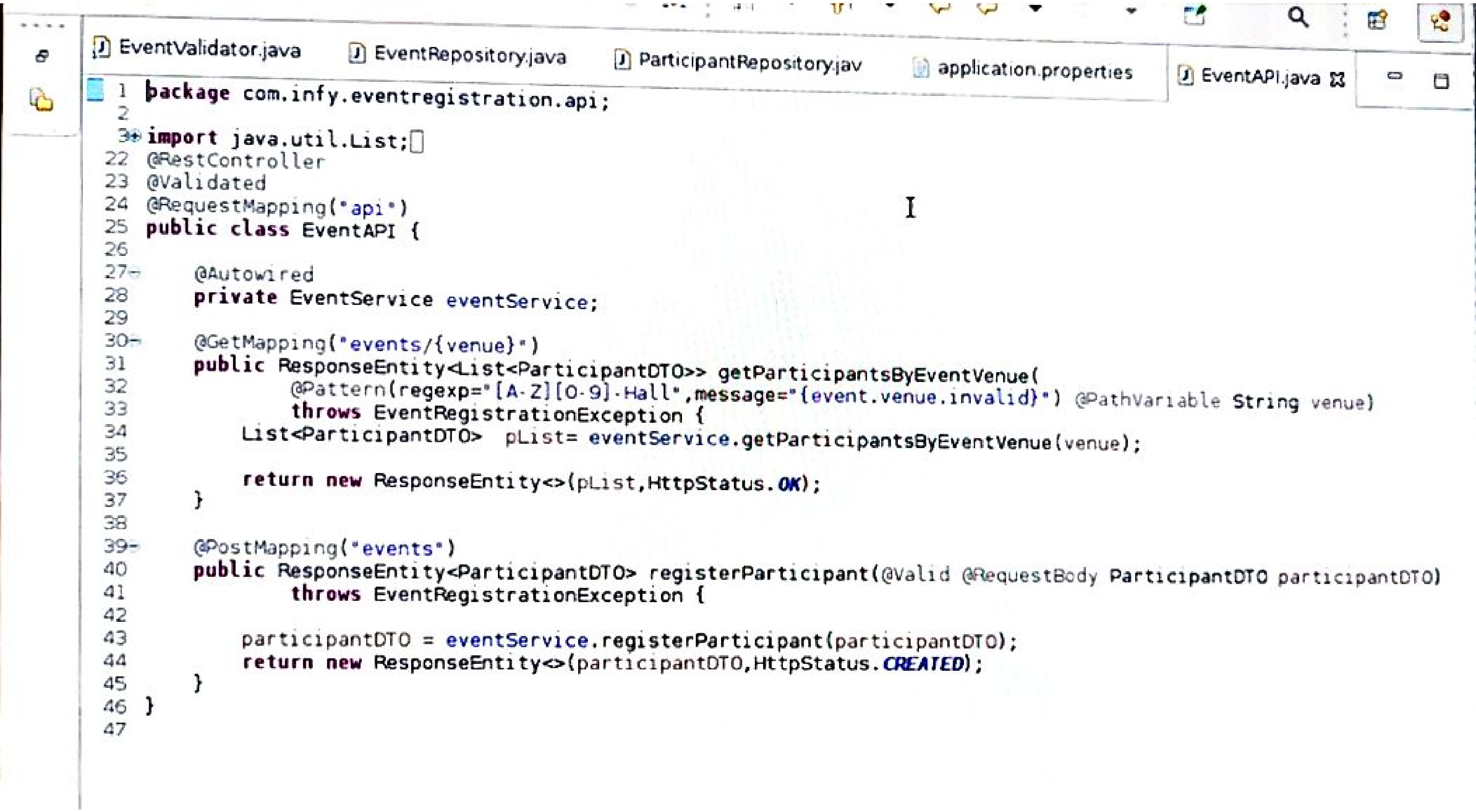


File Edit Source Refactor Navigate Search Project Run Window Help
Eclipse IDE
EventValidator.java
EventRepository.java
ParticipantRepository.jav
application.properties
EventAPI.java

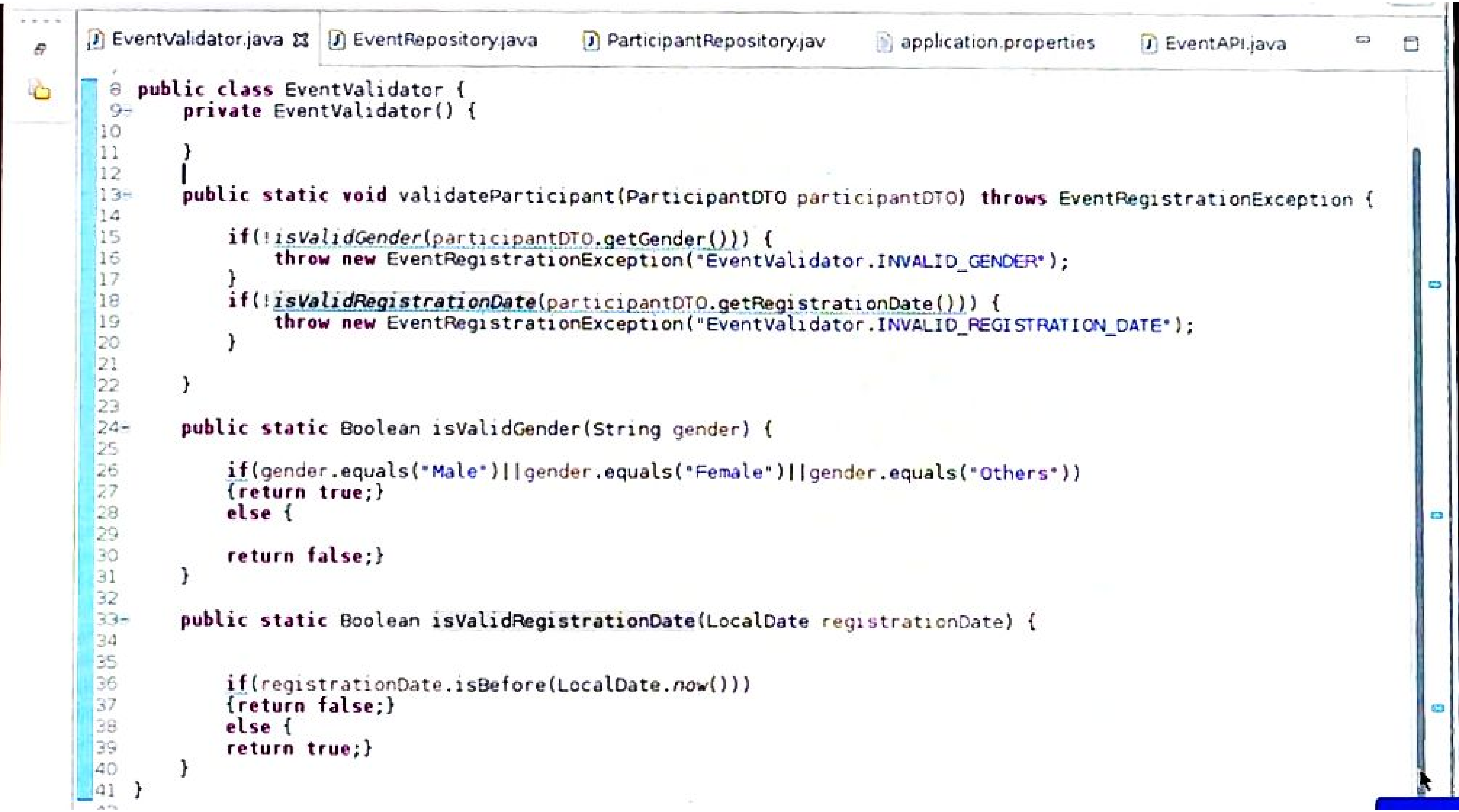
```java
package com.infy.eventregistration.repository;

import java.util.List;

import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.data.repository.query.Param;

import com.infy.eventregistration.entity.Event;
import com.infy.eventregistration.entity.Participant;

public interface ParticipantRepository extends CrudRepository<Participant,Integer> {

    @Query("SELECT p FROM Participant p WHERE p.event.venue=:venue")
    List<Participant> getParticipantsByEventVenue(@Param(value = "venue") String venue);

    List<Participant> findByEvent(Event event);
}
```


Project Explorer
iap_eventregistration [user_repo
src/main/java
com.infy.eventregistration
com.infy.eventregistration.
com.infy.eventregistration.
EventDTO.java
ParticipantDTO.java
com.infy.eventregistration.
com.infy.eventregistration.
com.infy.eventregistration.
com.infy.eventregistration.
com.infy.eventregistration.
com.infy.eventregistration.
src/main/resources
src/test/java
Maven Dependencies
JRE System Library [JavaSE-11]
log
ParticipantDTO.java

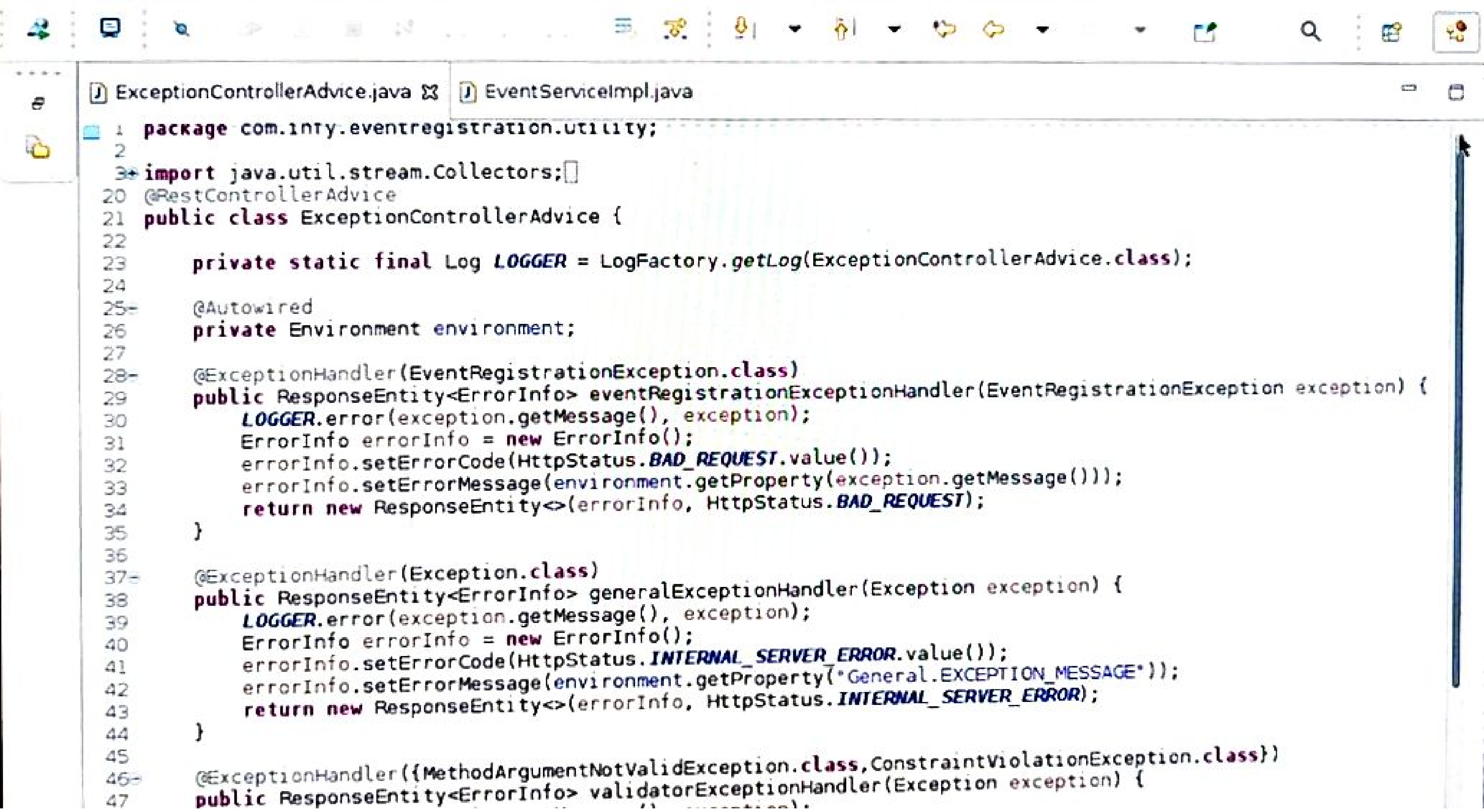
```java
package com.infy.eventregistration.dto;

import java.time.LocalDate;

public class ParticipantDTO {

    private Integer participantId;
    @NotNull(message="{participant.name.notpresent}")
    @Pattern(regexp="[A-Z][a-z]*",message="{participant.name.invalid}")
    private String name;
    private String emailId;
    private String gender;
    private LocalDate registrationDate;
    @NotNull(message="{participant.event.notpresent}")
    private EventDTO eventDTO;

    public Integer getParticipantId() {
        return participantId;
    }
    public void setParticipantId(Integer participantId) {
        this.participantId = participantId;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
```



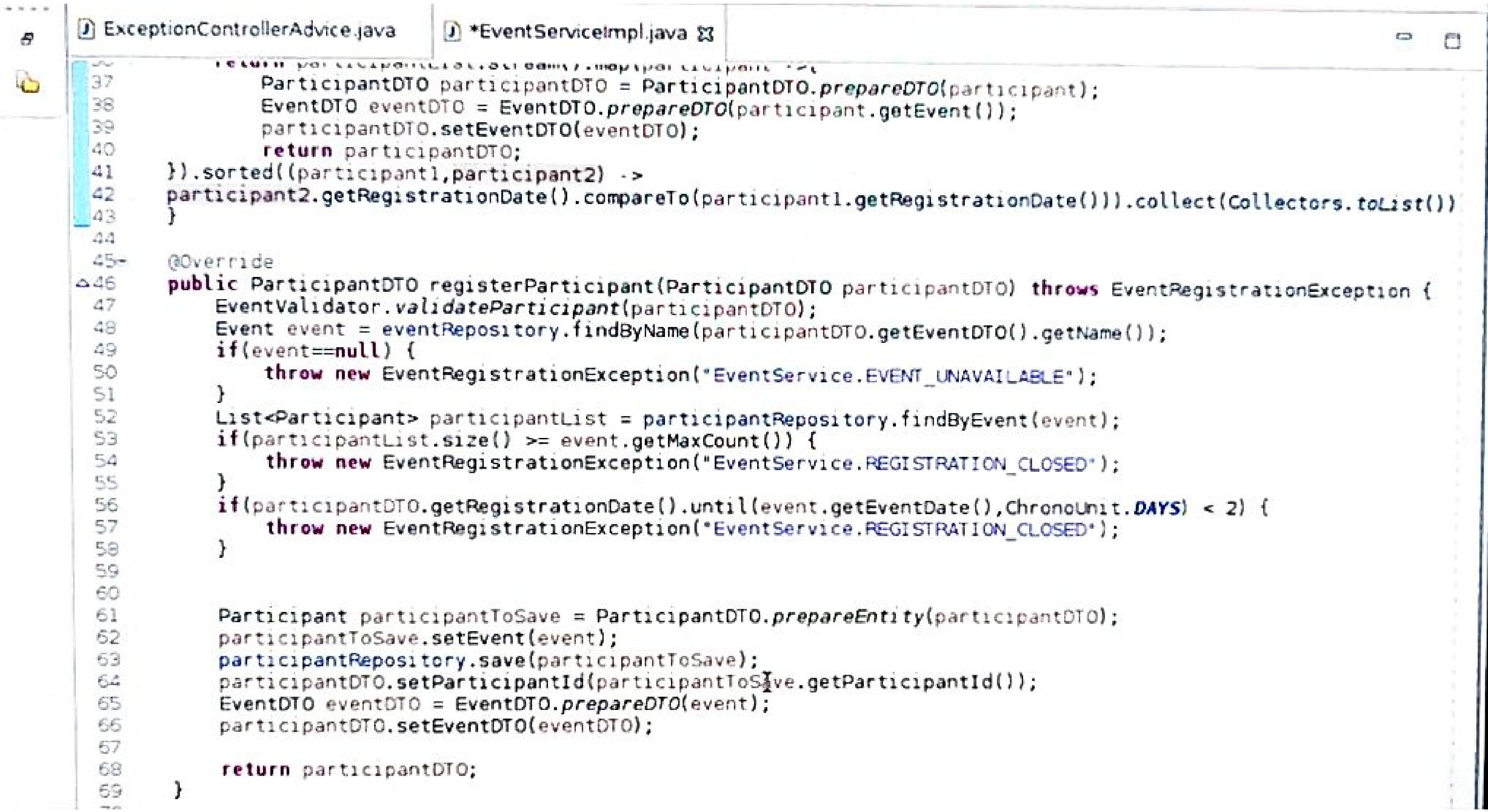
```java
package com.infy.eventregistration.api;

import java.util.List;
@RestController
@Validated
@RequestMapping("api")
public class EventAPI {

    @Autowired
    private EventService eventService;

    @GetMapping("events/{venue}")
    public ResponseEntity<List<ParticipantDTO>> getParticipantsByEventVenue(
            @Pattern(regexp="[A-Z][0-9]-Hall",message="{event.venue.invalid}") @PathVariable String venue)
            throws EventRegistrationException {
        List<ParticipantDTO>  pList= eventService.getParticipantsByEventVenue(venue);

        return new ResponseEntity<>(pList,HttpStatus.OK);
    }

    @PostMapping("events")
    public ResponseEntity<ParticipantDTO> registerParticipant(@Valid @RequestBody ParticipantDTO participantDTO)
            throws EventRegistrationException {

        participantDTO = eventService.registerParticipant(participantDTO);
        return new ResponseEntity<>(participantDTO,HttpStatus.CREATED);
    }
}

```



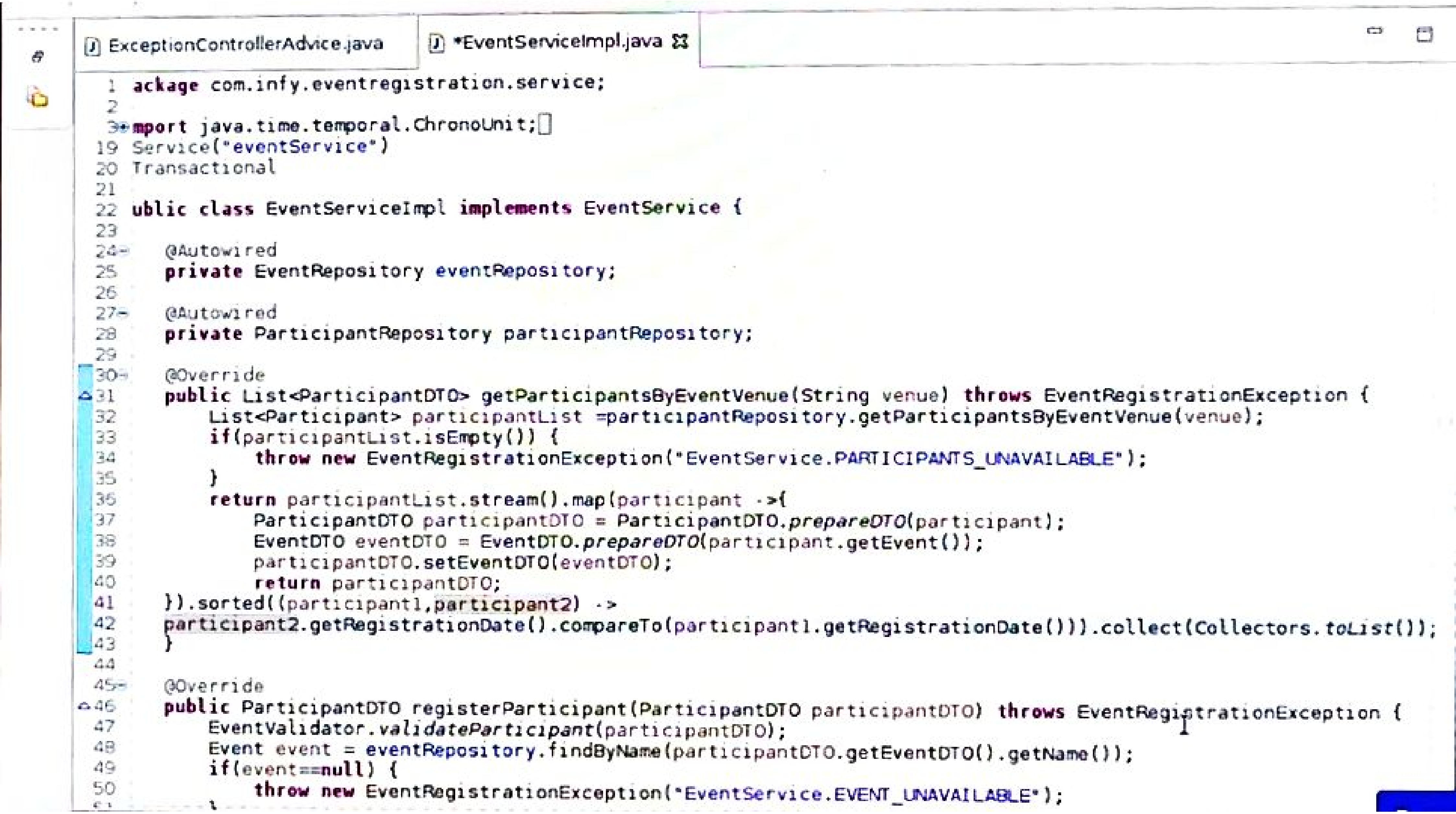
```java
public class EventValidator {
    private EventValidator() {

    }

    public static void validateParticipant(ParticipantDTO participantDTO) throws EventRegistrationException {

        if(!isValidGender(participantDTO.getGender())) {
            throw new EventRegistrationException("EventValidator.INVALID_GENDER");
        }
        if(!isValidRegistrationDate(participantDTO.getRegistrationDate())) {
            throw new EventRegistrationException("EventValidator.INVALID_REGISTRATION_DATE");
        }

    }

    public static Boolean isValidGender(String gender) {

        if(gender.equals("Male")||gender.equals("Female")||gender.equals("Others"))
        {return true;}
        else {

        return false;}
    }

    public static Boolean isValidRegistrationDate(LocalDate registrationDate) {


        if(registrationDate.isBefore(LocalDate.now()))
        {return false;}
        else {
        return true;}
    }
}
```



```java
package com.infy.eventregistration.utility;

import java.util.stream.Collectors;
@RestControllerAdvice
public class ExceptionControllerAdvice {

    private static final Log LOGGER = LogFactory.getLog(ExceptionControllerAdvice.class);

    @Autowired
    private Environment environment;

    @ExceptionHandler(EventRegistrationException.class)
    public ResponseEntity<ErrorInfo> eventRegistrationExceptionHandler(EventRegistrationException exception) {
        LOGGER.error(exception.getMessage(), exception);
        ErrorInfo errorInfo = new ErrorInfo();
        errorInfo.setErrorCode(HttpStatus.BAD_REQUEST.value());
        errorInfo.setErrorMessage(environment.getProperty(exception.getMessage()));
        return new ResponseEntity<>(errorInfo, HttpStatus.BAD_REQUEST);
    }

    @ExceptionHandler(Exception.class)
    public ResponseEntity<ErrorInfo> generalExceptionHandler(Exception exception) {
        LOGGER.error(exception.getMessage(), exception);
        ErrorInfo errorInfo = new ErrorInfo();
        errorInfo.setErrorCode(HttpStatus.INTERNAL_SERVER_ERROR.value());
        errorInfo.setErrorMessage(environment.getProperty("General.EXCEPTION_MESSAGE"));
        return new ResponseEntity<>(errorInfo, HttpStatus.INTERNAL_SERVER_ERROR);
    }

    @ExceptionHandler({MethodArgumentNotValidException.class,ConstraintViolationException.class})
    public ResponseEntity<ErrorInfo> validatorExceptionHandler(Exception exception) {
```

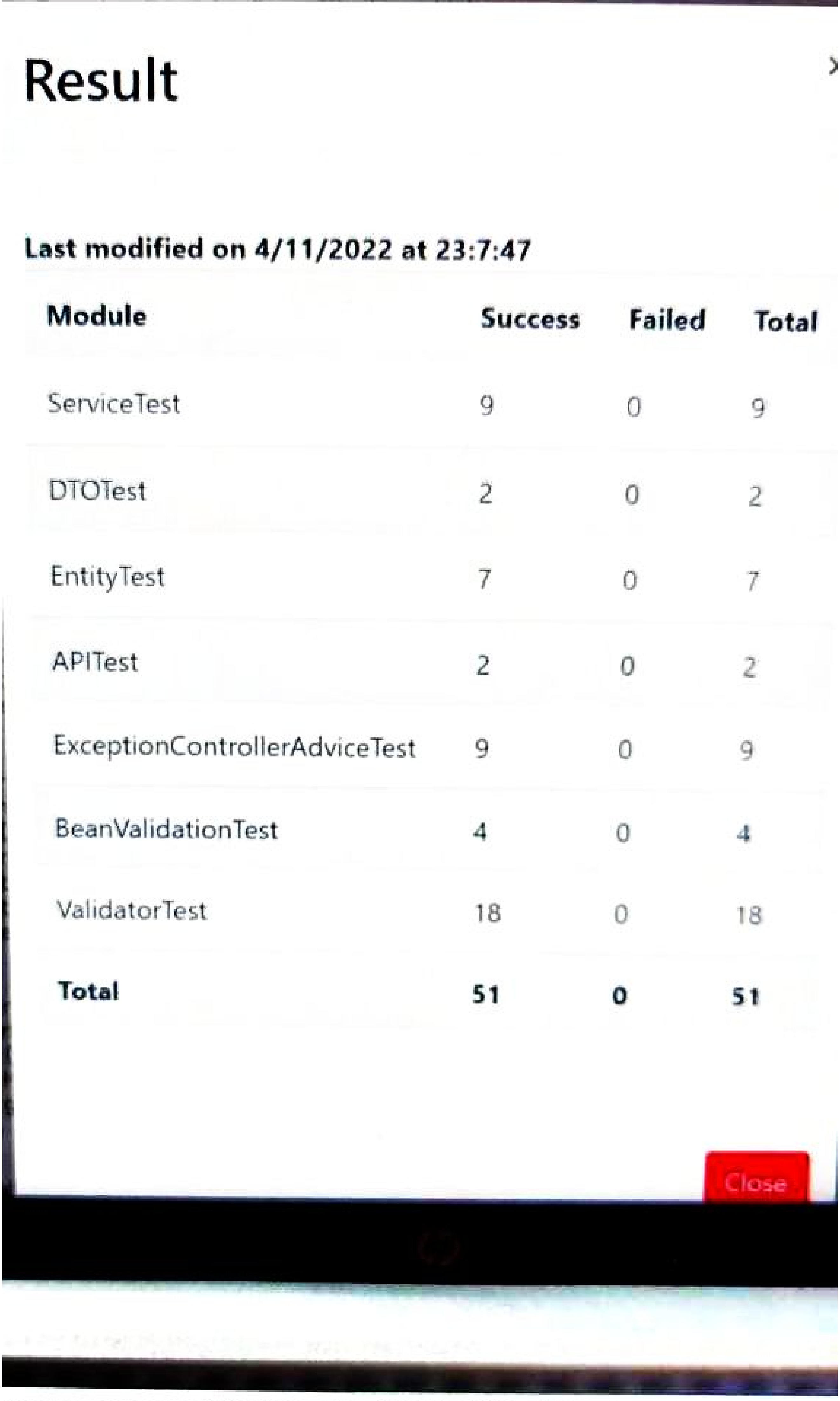


```java
        return participantList.stream().map(participant ->{
            ParticipantDTO participantDTO = ParticipantDTO.prepareDTO(participant);
            EventDTO eventDTO = EventDTO.prepareDTO(participant.getEvent());
            participantDTO.setEventDTO(eventDTO);
            return participantDTO;
    }).sorted((participant1,participant2) ->
    participant2.getRegistrationDate().compareTo(participant1.getRegistrationDate())).collect(Collectors.toList())
    }

    @Override
    public ParticipantDTO registerParticipant(ParticipantDTO participantDTO) throws EventRegistrationException {
        EventValidator.validateParticipant(participantDTO);
        Event event = eventRepository.findByName(participantDTO.getEventDTO().getName());
        if(event==null) {
            throw new EventRegistrationException("EventService.EVENT_UNAVAILABLE");
        }
        List<Participant> participantList = participantRepository.findByEvent(event);
        if(participantList.size() >= event.getMaxCount()) {
            throw new EventRegistrationException("EventService.REGISTRATION_CLOSED");
        }
        if(participantDTO.getRegistrationDate().until(event.getEventDate(),ChronoUnit.DAYS) < 2) {
            throw new EventRegistrationException("EventService.REGISTRATION_CLOSED");
        }


        Participant participantToSave = ParticipantDTO.prepareEntity(participantDTO);
        participantToSave.setEvent(event);
        participantRepository.save(participantToSave);
        participantDTO.setParticipantId(participantToSave.getParticipantId());
        EventDTO eventDTO = EventDTO.prepareDTO(event);
        participantDTO.setEventDTO(eventDTO);

        return participantDTO;
    }
```



ExceptionControllerAdvice.java   *EventServiceImpl.java

```
ackage com.infy.eventregistration.service;

mport java.time.temporal.ChronoUnit;
Service("eventService")
Transactional

ublic class EventServiceImpl implements EventService {

    @Autowired
    private EventRepository eventRepository;

    @Autowired
    private ParticipantRepository participantRepository;

    @Override
    public List<ParticipantDTO> getParticipantsByEventVenue(String venue) throws EventRegistrationException {
        List<Participant> participantList =participantRepository.getParticipantsByEventVenue(venue);
        if(participantList.isEmpty()) {
            throw new EventRegistrationException("EventService.PARTICIPANTS_UNAVAILABLE");
        }
        return participantList.stream().map(participant ->{
            ParticipantDTO participantDTO = ParticipantDTO.prepareDTO(participant);
            EventDTO eventDTO = EventDTO.prepareDTO(participant.getEvent());
            participantDTO.setEventDTO(eventDTO);
            return participantDTO;
    }).sorted((participant1,participant2) ->
    participant2.getRegistrationDate().compareTo(participant1.getRegistrationDate())).collect(Collectors.toList());
    }

    @Override
    public ParticipantDTO registerParticipant(ParticipantDTO participantDTO) throws EventRegistrationException {
        EventValidator.validateParticipant(participantDTO);
        Event event = eventRepository.findByName(participantDTO.getEventDTO().getName());
        if(event==null) {
            throw new EventRegistrationException("EventService.EVENT_UNAVAILABLE");
```



Result

Last modified on 4/11/2022 at 23:7:47

Module	**Success**	**Failed**	**Total**
ServiceTest	9	0	9
DTOTest	2	0	2
EntityTest	7	0	7
APITest	2	0	2
ExceptionControllerAdviceTest	9	0	9
BeanValidationTest	4	0	4
ValidatorTest	18	0	18
Total	**51**	**0**	**51**