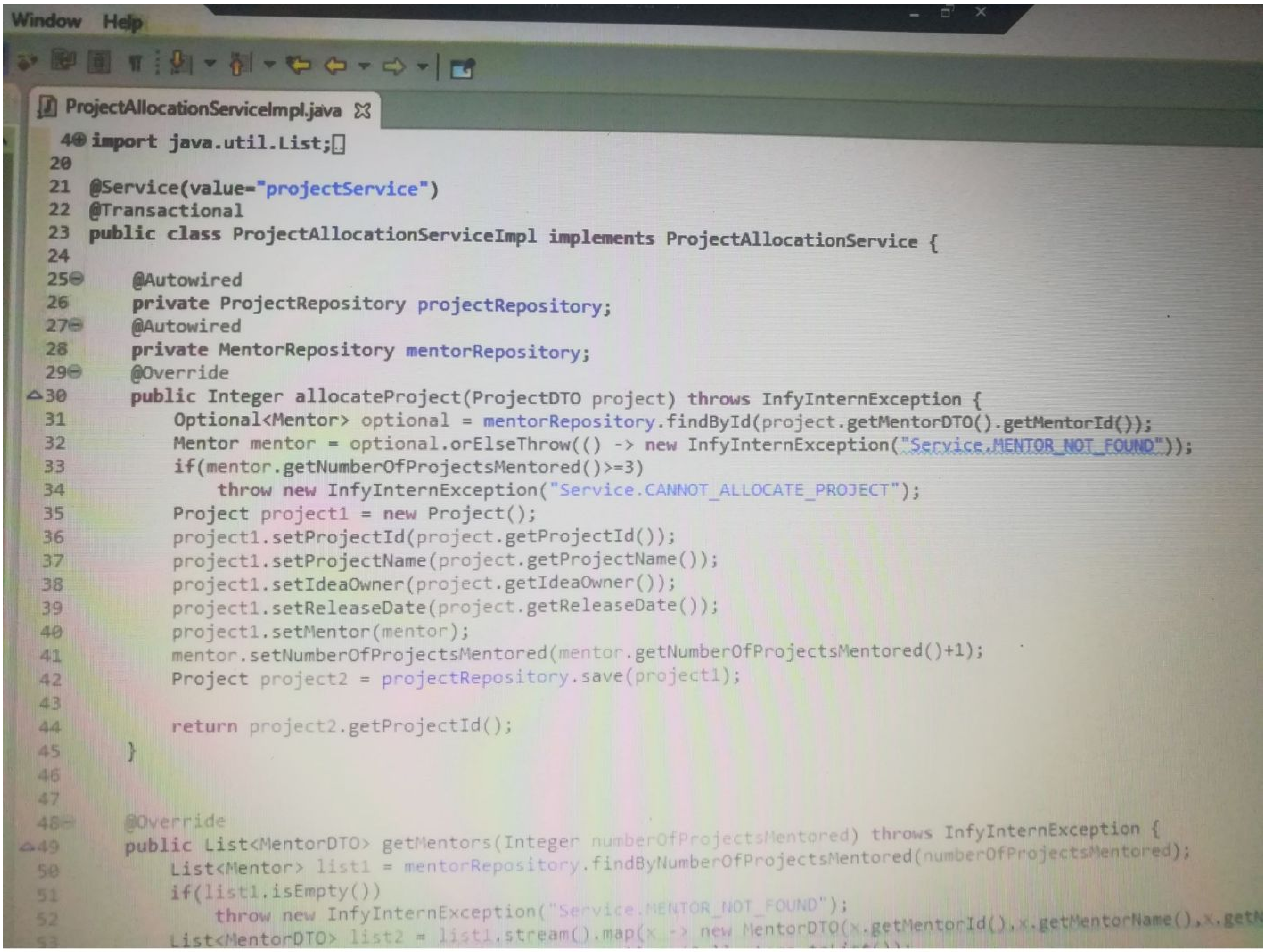
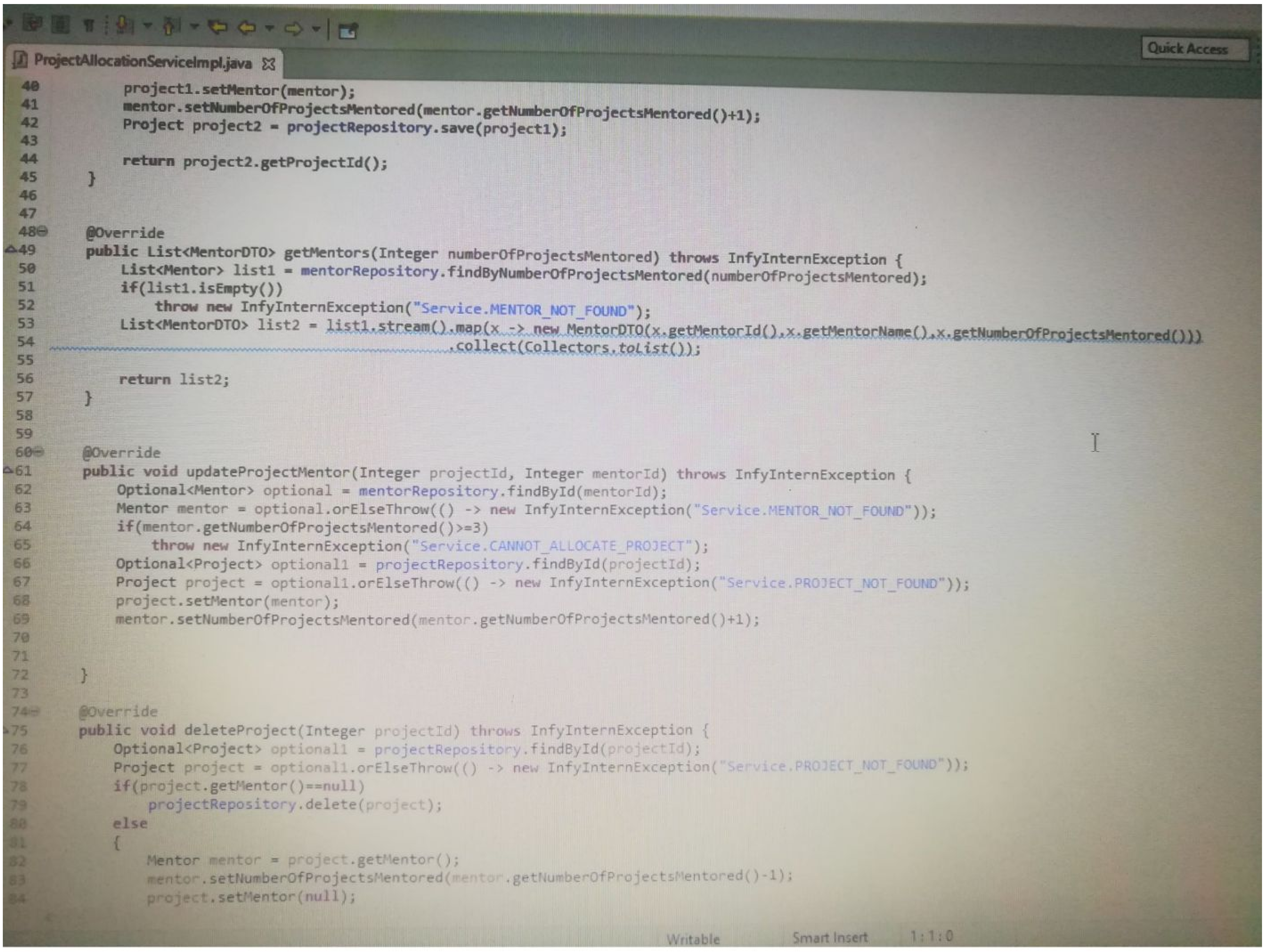


Window Help

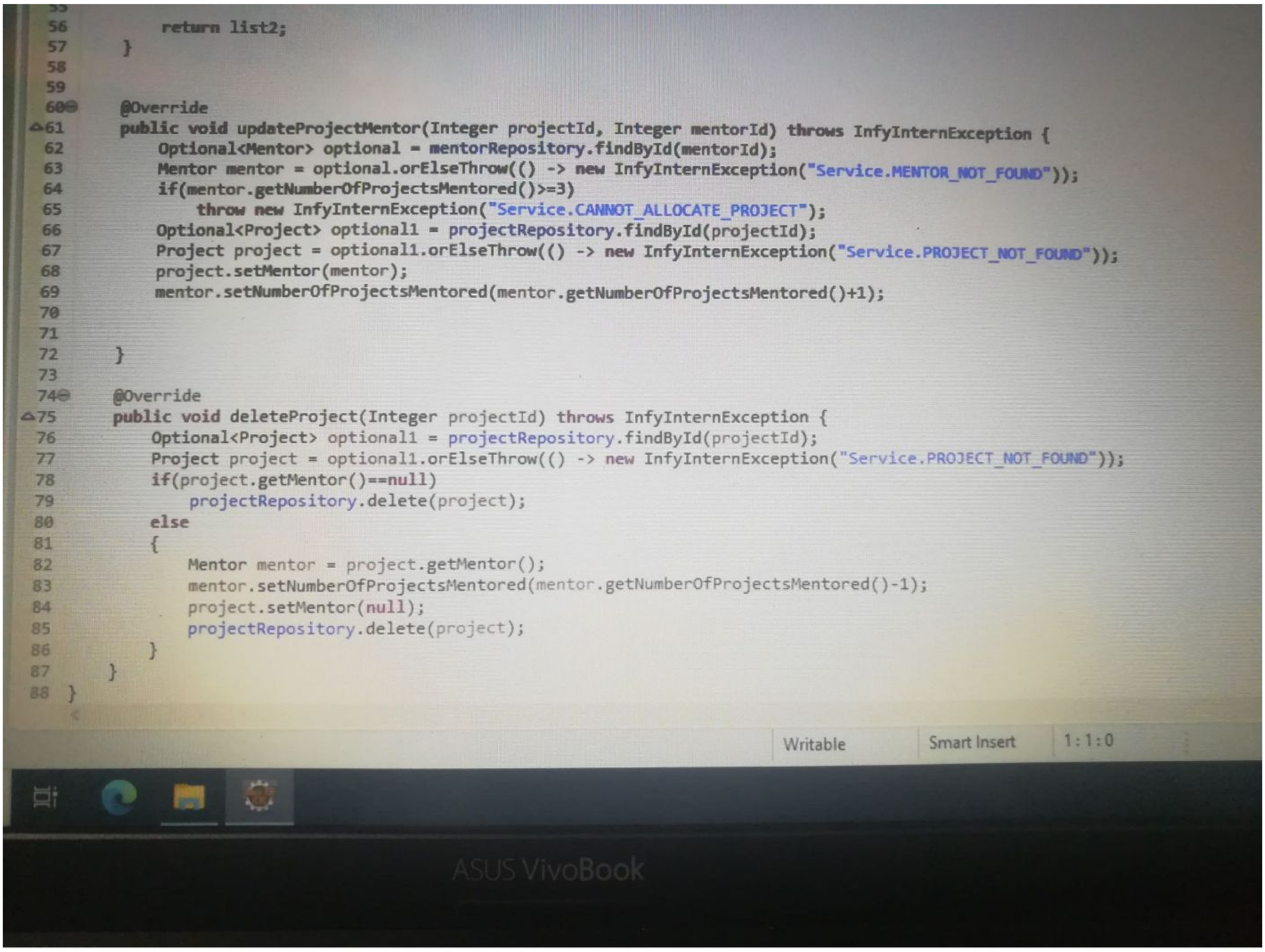
ProjectAllocationServiceImpl.java

```java
import java.util.List;

@Service(value="projectService")
@Transactional
public class ProjectAllocationServiceImpl implements ProjectAllocationService {

    @Autowired
    private ProjectRepository projectRepository;
    @Autowired
    private MentorRepository mentorRepository;
    @Override
    public Integer allocateProject(ProjectDTO project) throws InfyInternException {
        Optional<Mentor> optional = mentorRepository.findById(project.getMentorDTO().getMentorId());
        Mentor mentor = optional.orElseThrow(() -> new InfyInternException("Service.MENTOR_NOT_FOUND"));
        if(mentor.getNumberOfProjectsMentored()>=3)
            throw new InfyInternException("Service.CANNOT_ALLOCATE_PROJECT");
        Project project1 = new Project();
        project1.setProjectId(project.getProjectId());
        project1.setProjectName(project.getProjectName());
        project1.setIdeaOwner(project.getIdeaOwner());
        project1.setReleaseDate(project.getReleaseDate());
        project1.setMentor(mentor);
        mentor.setNumberOfProjectsMentored(mentor.getNumberOfProjectsMentored()+1);
        Project project2 = projectRepository.save(project1);

        return project2.getProjectId();
    }


    @Override
    public List<MentorDTO> getMentors(Integer numberOfProjectsMentored) throws InfyInternException {
        List<Mentor> list1 = mentorRepository.findByNumberOfProjectsMentored(numberOfProjectsMentored);
        if(list1.isEmpty())
            throw new InfyInternException("Service.MENTOR_NOT_FOUND");
        List<MentorDTO> list2 = list1.stream().map(x -> new MentorDTO(x.getMentorId(),x.getMentorName(),x.getN
```

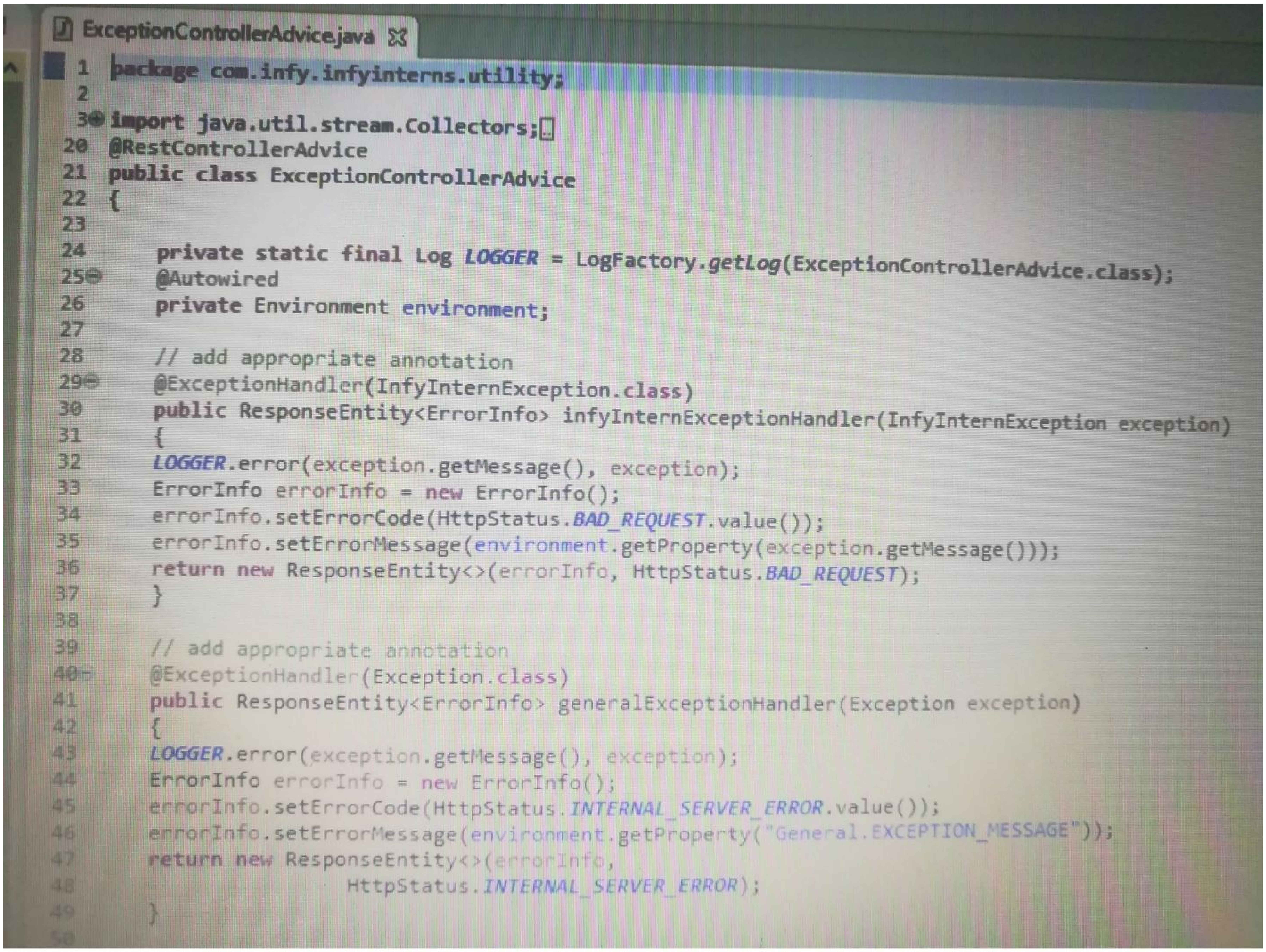
Scanned by TapScanner



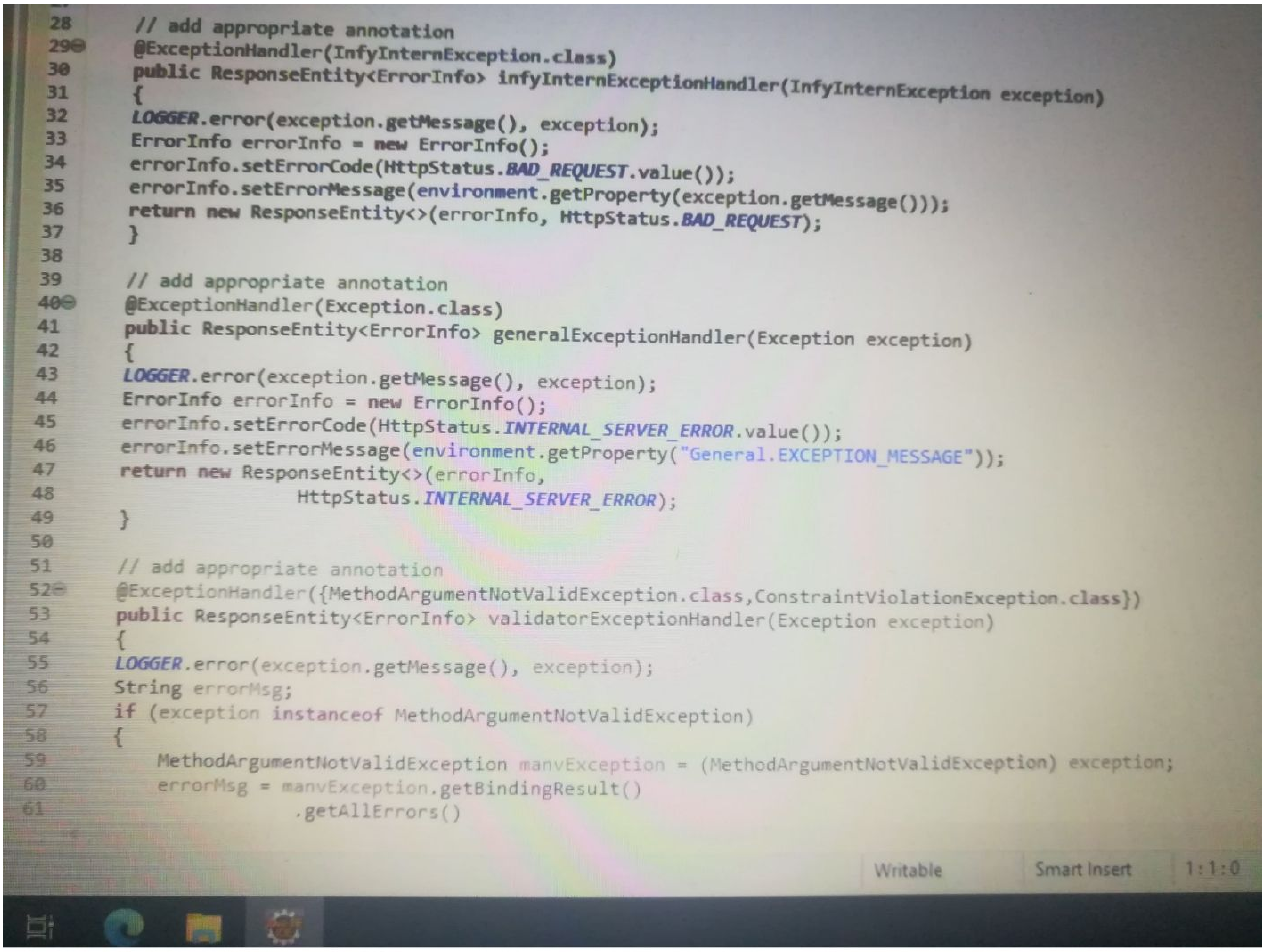
```java
        project1.setMentor(mentor);
        mentor.setNumberOfProjectsMentored(mentor.getNumberOfProjectsMentored()+1);
        Project project2 = projectRepository.save(project1);

        return project2.getProjectId();
    }


    @Override
    public List<MentorDTO> getMentors(Integer numberOfProjectsMentored) throws InfyInternException {
        List<Mentor> list1 = mentorRepository.findByNumberOfProjectsMentored(numberOfProjectsMentored);
        if(list1.isEmpty())
            throw new InfyInternException("Service.MENTOR_NOT_FOUND");
        List<MentorDTO> list2 = list1.stream().map(x -> new MentorDTO(x.getMentorId(),x.getMentorName(),x.getNumberOfProjectsMentored()))
                                                 .collect(Collectors.toList());

        return list2;
    }


    @Override
    public void updateProjectMentor(Integer projectId, Integer mentorId) throws InfyInternException {
        Optional<Mentor> optional = mentorRepository.findById(mentorId);
        Mentor mentor = optional.orElseThrow(() -> new InfyInternException("Service.MENTOR_NOT_FOUND"));
        if(mentor.getNumberOfProjectsMentored()>=3)
            throw new InfyInternException("Service.CANNOT_ALLOCATE_PROJECT");
        Optional<Project> optional1 = projectRepository.findById(projectId);
        Project project = optional1.orElseThrow(() -> new InfyInternException("Service.PROJECT_NOT_FOUND"));
        project.setMentor(mentor);
        mentor.setNumberOfProjectsMentored(mentor.getNumberOfProjectsMentored()+1);


    }

    @Override
    public void deleteProject(Integer projectId) throws InfyInternException {
        Optional<Project> optional1 = projectRepository.findById(projectId);
        Project project = optional1.orElseThrow(() -> new InfyInternException("Service.PROJECT_NOT_FOUND"));
        if(project.getMentor()==null)
            projectRepository.delete(project);
        else
        {
            Mentor mentor = project.getMentor();
            mentor.setNumberOfProjectsMentored(mentor.getNumberOfProjectsMentored()-1);
            project.setMentor(null);
```



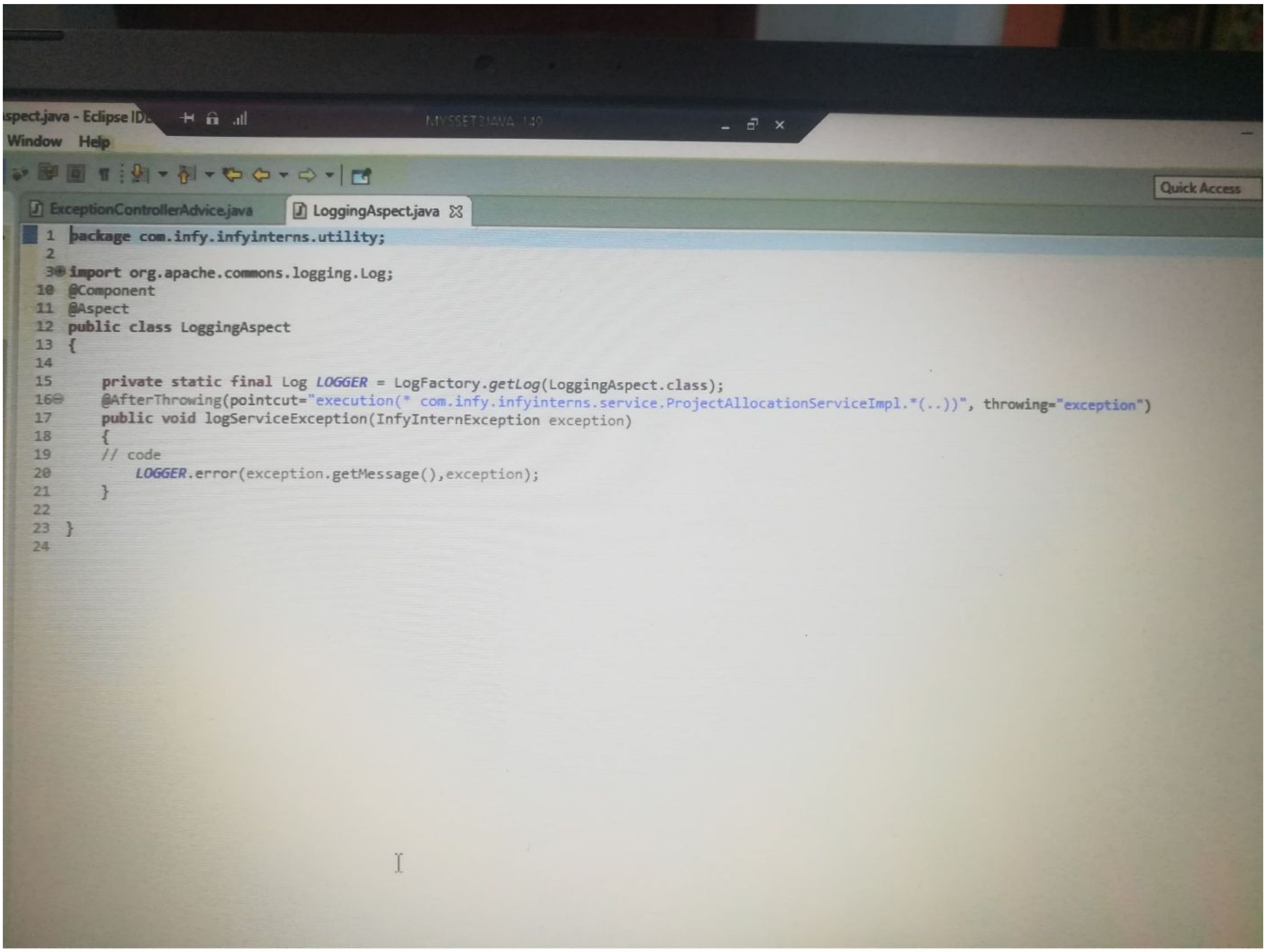
```java
        return list2;
    }


    @Override
    public void updateProjectMentor(Integer projectId, Integer mentorId) throws InfyInternException {
        Optional<Mentor> optional = mentorRepository.findById(mentorId);
        Mentor mentor = optional.orElseThrow(() -> new InfyInternException("Service.MENTOR_NOT_FOUND"));
        if(mentor.getNumberOfProjectsMentored()>=3)
            throw new InfyInternException("Service.CANNOT_ALLOCATE_PROJECT");
        Optional<Project> optional1 = projectRepository.findById(projectId);
        Project project = optional1.orElseThrow(() -> new InfyInternException("Service.PROJECT_NOT_FOUND"));
        project.setMentor(mentor);
        mentor.setNumberOfProjectsMentored(mentor.getNumberOfProjectsMentored()+1);


    }

    @Override
    public void deleteProject(Integer projectId) throws InfyInternException {
        Optional<Project> optional1 = projectRepository.findById(projectId);
        Project project = optional1.orElseThrow(() -> new InfyInternException("Service.PROJECT_NOT_FOUND"));
        if(project.getMentor()==null)
            projectRepository.delete(project);
        else
        {
            Mentor mentor = project.getMentor();
            mentor.setNumberOfProjectsMentored(mentor.getNumberOfProjectsMentored()-1);
            project.setMentor(null);
            projectRepository.delete(project);
        }
    }
}
```



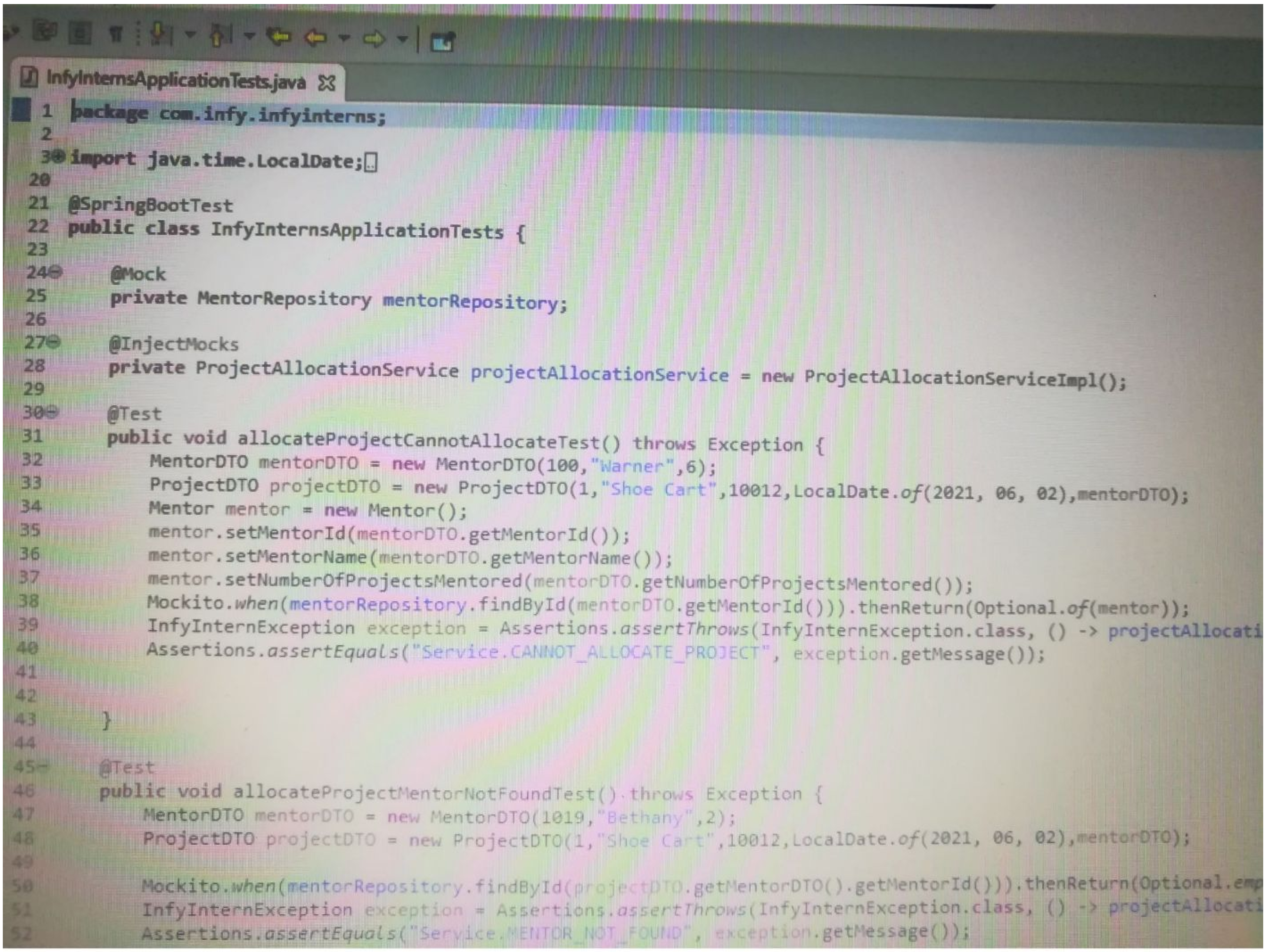
```
package com.infy.infyinterns.utility;

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

    private static final Log LOGGER = LogFactory.getLog(ExceptionControllerAdvice.class);
    @Autowired
    private Environment environment;

    // add appropriate annotation
    @ExceptionHandler(InfyInternException.class)
    public ResponseEntity<ErrorInfo> infyInternExceptionHandler(InfyInternException 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);
    }

    // add appropriate annotation
    @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);
    }
```



```
    // add appropriate annotation
    @ExceptionHandler(InfyInternException.class)
    public ResponseEntity<ErrorInfo> infyInternExceptionHandler(InfyInternException 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);
    }

    // add appropriate annotation
    @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);
    }

    // add appropriate annotation
    @ExceptionHandler({MethodArgumentNotValidException.class,ConstraintViolationException.class})
    public ResponseEntity<ErrorInfo> validatorExceptionHandler(Exception exception)
    {
    LOGGER.error(exception.getMessage(), exception);
    String errorMsg;
    if (exception instanceof MethodArgumentNotValidException)
    {
        MethodArgumentNotValidException manvException = (MethodArgumentNotValidException) exception;
        errorMsg = manvException.getBindingResult()
                .getAllErrors()
```



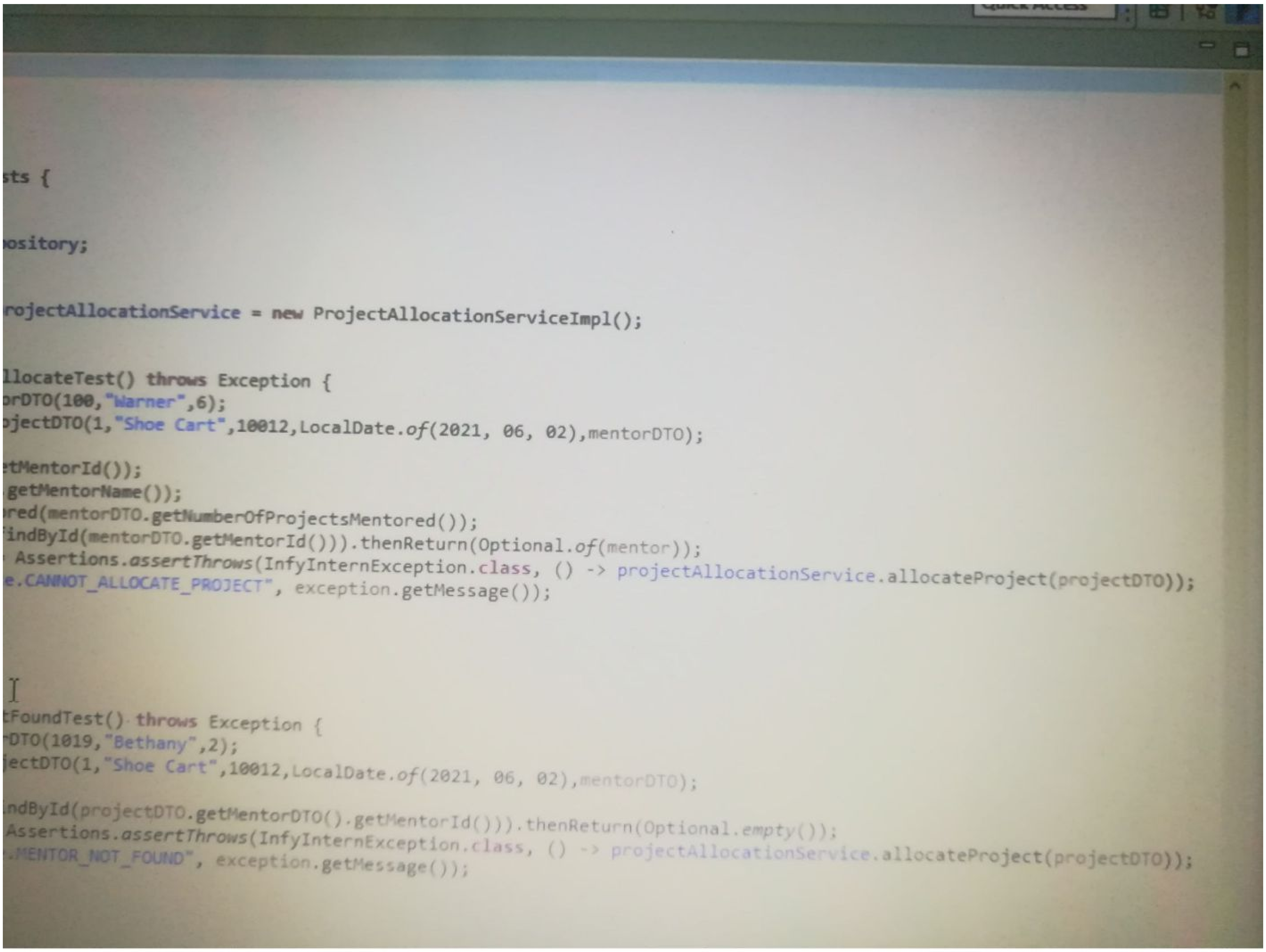
```java
package com.infy.infyinterns.utility;

import org.apache.commons.logging.Log;
@Component
@Aspect
public class LoggingAspect
{

    private static final Log LOGGER = LogFactory.getLog(LoggingAspect.class);
    @AfterThrowing(pointcut="execution(* com.infy.infyinterns.service.ProjectAllocationServiceImpl.*(..))", throwing="exception")
    public void logServiceException(InfyInternException exception)
    {
    // code
        LOGGER.error(exception.getMessage(),exception);
    }

}
```



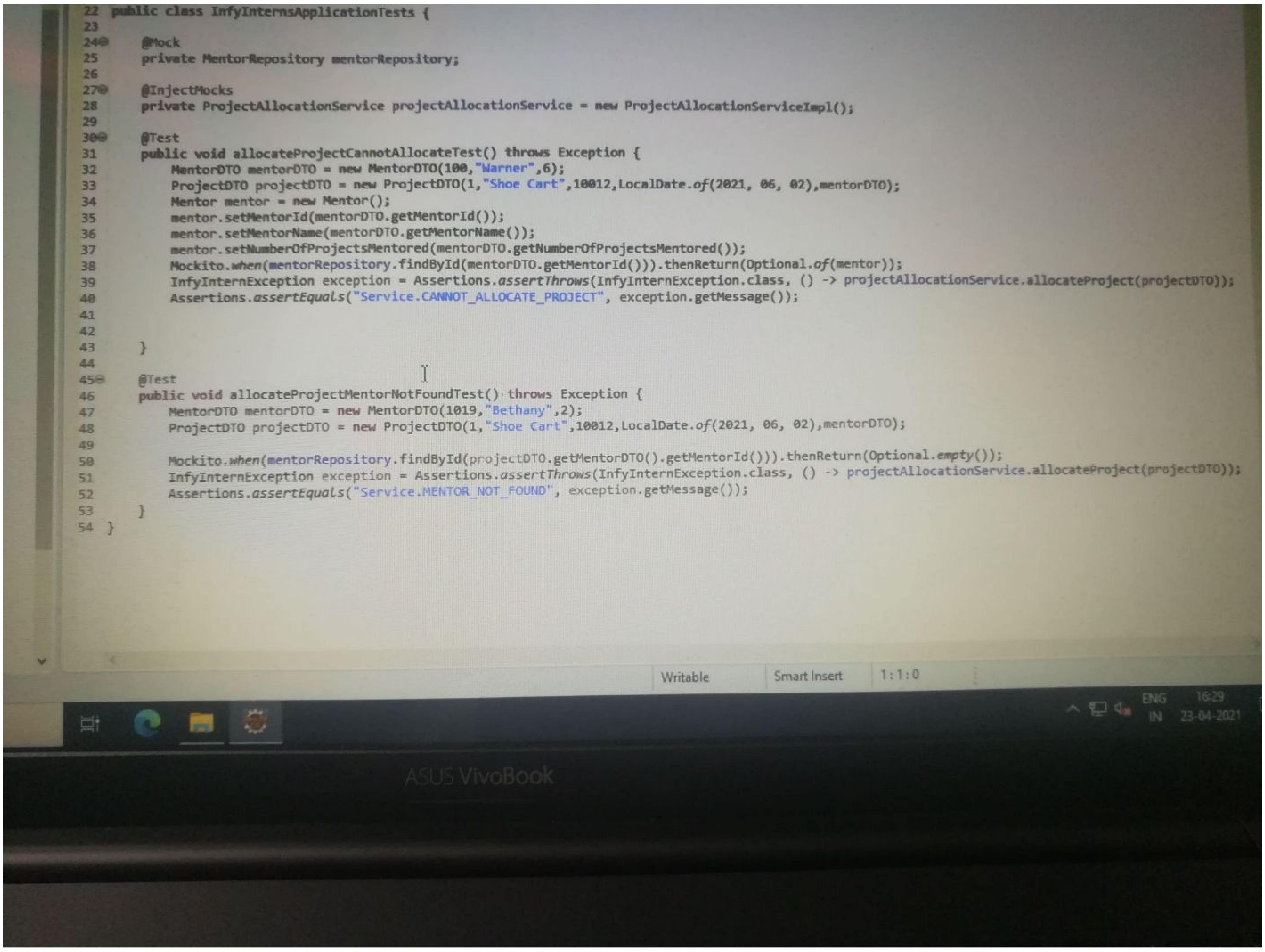
InfyInternsApplicationTests.java

```java
package com.infy.infyinterns;

import java.time.LocalDate;

@SpringBootTest
public class InfyInternsApplicationTests {

    @Mock
    private MentorRepository mentorRepository;

    @InjectMocks
    private ProjectAllocationService projectAllocationService = new ProjectAllocationServiceImpl();

    @Test
    public void allocateProjectCannotAllocateTest() throws Exception {
        MentorDTO mentorDTO = new MentorDTO(100,"Warner",6);
        ProjectDTO projectDTO = new ProjectDTO(1,"Shoe Cart",10012,LocalDate.of(2021, 06, 02),mentorDTO);
        Mentor mentor = new Mentor();
        mentor.setMentorId(mentorDTO.getMentorId());
        mentor.setMentorName(mentorDTO.getMentorName());
        mentor.setNumberOfProjectsMentored(mentorDTO.getNumberOfProjectsMentored());
        Mockito.when(mentorRepository.findById(mentorDTO.getMentorId())).thenReturn(Optional.of(mentor));
        InfyInternException exception = Assertions.assertThrows(InfyInternException.class, () -> projectAllocati
        Assertions.assertEquals("Service.CANNOT_ALLOCATE_PROJECT", exception.getMessage());


    }

    @Test
    public void allocateProjectMentorNotFoundTest() throws Exception {
        MentorDTO mentorDTO = new MentorDTO(1019,"Bethany",2);
        ProjectDTO projectDTO = new ProjectDTO(1,"Shoe Cart",10012,LocalDate.of(2021, 06, 02),mentorDTO);

        Mockito.when(mentorRepository.findById(projectDTO.getMentorDTO().getMentorId())).thenReturn(Optional.emp
        InfyInternException exception = Assertions.assertThrows(InfyInternException.class, () -> projectAllocati
        Assertions.assertEquals("Service.MENTOR_NOT_FOUND", exception.getMessage());
```

Scanned by TapScanner



```
sts {

pository;

rojectAllocationService = new ProjectAllocationServiceImpl();

llocateTest() throws Exception {
orDTO(100,"Warner",6);
ojectDTO(1,"Shoe Cart",10012,LocalDate.of(2021, 06, 02),mentorDTO);

tMentorId());
 getMentorName());
ored(mentorDTO.getNumberOfProjectsMentored());
findById(mentorDTO.getMentorId())).thenReturn(Optional.of(mentor));
 Assertions.assertThrows(InfyInternException.class, () -> projectAllocationService.allocateProject(projectDTO));
e.CANNOT_ALLOCATE_PROJECT", exception.getMessage());



tFoundTest() throws Exception {
rDTO(1019,"Bethany",2);
jectDTO(1,"Shoe Cart",10012,LocalDate.of(2021, 06, 02),mentorDTO);

ndById(projectDTO.getMentorDTO().getMentorId())).thenReturn(Optional.empty());
 Assertions.assertThrows(InfyInternException.class, () -> projectAllocationService.allocateProject(projectDTO));
.MENTOR_NOT_FOUND", exception.getMessage());
```



```java
public class InfyInternsApplicationTests {

    @Mock
    private MentorRepository mentorRepository;

    @InjectMocks
    private ProjectAllocationService projectAllocationService = new ProjectAllocationServiceImpl();

    @Test
    public void allocateProjectCannotAllocateTest() throws Exception {
        MentorDTO mentorDTO = new MentorDTO(100,"Warner",6);
        ProjectDTO projectDTO = new ProjectDTO(1,"Shoe Cart",10012,LocalDate.of(2021, 06, 02),mentorDTO);
        Mentor mentor = new Mentor();
        mentor.setMentorId(mentorDTO.getMentorId());
        mentor.setMentorName(mentorDTO.getMentorName());
        mentor.setNumberOfProjectsMentored(mentorDTO.getNumberOfProjectsMentored());
        Mockito.when(mentorRepository.findById(mentorDTO.getMentorId())).thenReturn(Optional.of(mentor));
        InfyInternException exception = Assertions.assertThrows(InfyInternException.class, () -> projectAllocationService.allocateProject(projectDTO));
        Assertions.assertEquals("Service.CANNOT_ALLOCATE_PROJECT", exception.getMessage());


    }

    @Test
    public void allocateProjectMentorNotFoundTest() throws Exception {
        MentorDTO mentorDTO = new MentorDTO(1019,"Bethany",2);
        ProjectDTO projectDTO = new ProjectDTO(1,"Shoe Cart",10012,LocalDate.of(2021, 06, 02),mentorDTO);

        Mockito.when(mentorRepository.findById(projectDTO.getMentorDTO().getMentorId())).thenReturn(Optional.empty());
        InfyInternException exception = Assertions.assertThrows(InfyInternException.class, () -> projectAllocationService.allocateProject(projectDTO));
        Assertions.assertEquals("Service.MENTOR_NOT_FOUND", exception.getMessage());
    }
}
```