BILKENT UNIVERSITY
COMPUTER SCIENCE DEPARTMENT
CS 319 OBJECT-ORIENTED SOFTWARE ENGINEERING
DELIVERABLE 4

SPRING 2025

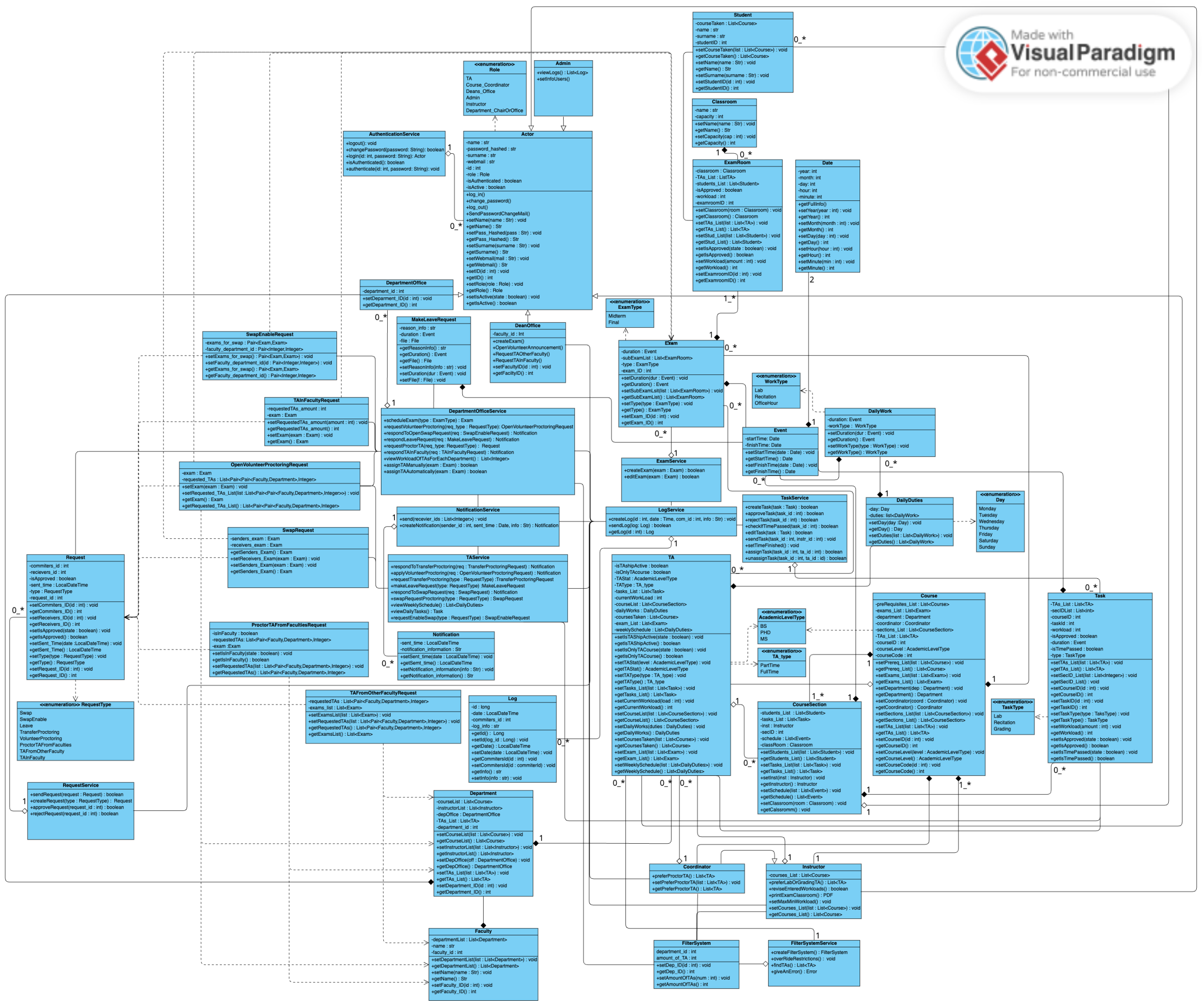
GROUP 2 SECTION 1

10.05.2025

Full Name	ID
Perhat Amanlyyev	22201007
Emiralp İlgen	22203114
Anıl Keskin	22201915
İlmay Taş	22201715
Simay Uygur	22203328

Table of Contents

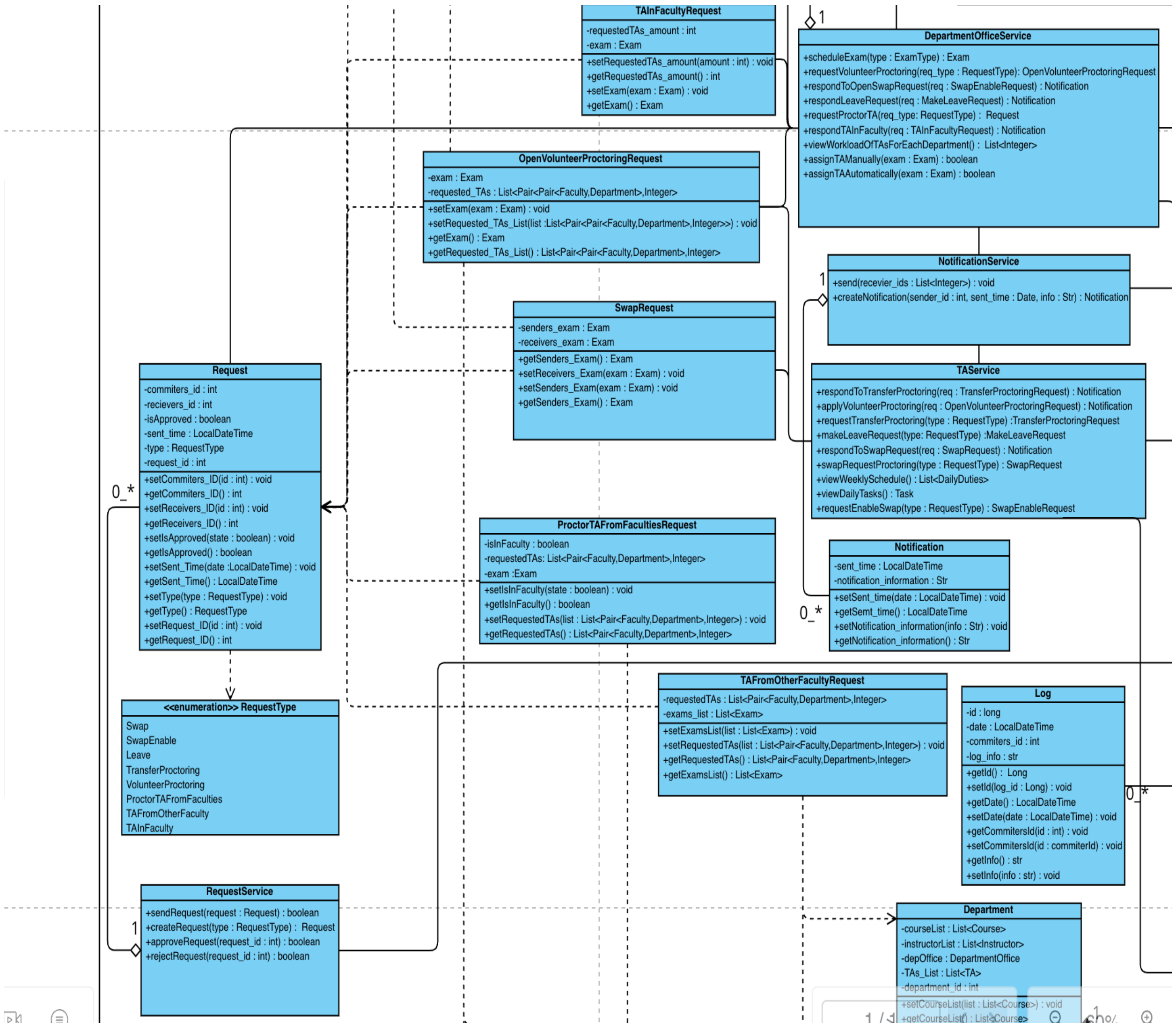
A. Class Diagram



The link for diagram: https://online.visual-paradigm.com/share.jsp?id=333439313030372d3131#diagram:workspace=patxxmja&proj=0&id=11

B. Software Design Patterns Used

1. Command Design Pattern for Handling Requests


TAInFacultyRequest
-requestedTAs_amount : int
-exam : Exam
+setRequestedTAs_amount(amount : int) : void
+getRequestedTAs_amount() : int
+setExam(exam : Exam) : void
+getExam() : Exam
DepartmentOfficeService
+scheduleExam(type : ExamType) : Exam
+requestVolunteerProctoring(req_type : RequestType): OpenVolunteerProctoringRequest
+respondToOpenSwapRequest(req : SwapEnableRequest) : Notification
+respondLeaveRequest(req : MakeLeaveRequest) : Notification
+requestProctorTA(req_type: RequestType) : Request
+respondTAInFaculty(req : TAInFacultyRequest) : Notification
+viewWorkloadOfTAsForEachDepartment() : List<Integer>
+assignTAManually(exam : Exam) : boolean
+assignTAAutomatically(exam : Exam) : boolean
OpenVolunteerProctoringRequest
-exam : Exam
-requested_TAs : List<Pair<Pair<Faculty,Department>,Integer>
+setExam(exam : Exam) : void
+setRequested_TAs_List(list :List<Pair<Pair<Faculty,Department>,Integer>>) : void
+getExam() : Exam
+getRequested_TAs_List() : List<Pair<Pair<Faculty,Department>,Integer>
NotificationService
+send(recevier_ids : List<Integer>) : void
+createNotification(sender_id : int, sent_time : Date, info : Str) : Notification
SwapRequest
-senders_exam : Exam
-receivers_exam : Exam
+getSenders_Exam() : Exam
+setReceivers_Exam(exam : Exam) : void
+setSenders_Exam(exam : Exam) : void
+getSenders_Exam() : Exam
TAService
+respondToTransferProctoring(req : TransferProctoringRequest) : Notification
+applyVolunteerProctoring(req : OpenVolunteerProctoringRequest) : Notification
+requestTransferProctoring(type : RequestType) :TransferProctoringRequest
+makeLeaveRequest(type: RequestType) :MakeLeaveRequest
+respondToSwapRequest(req : SwapRequest) : Notification
+swapRequestProctoring(type : RequestType) : SwapRequest
+viewWeeklySchedule() : List<DailyDuties>
+viewDailyTasks() : Task
+requestEnableSwap(type : RequestType) : SwapEnableRequest
Request
-commiters_id : int
-recievers_id : int
-isApproved : boolean
-sent_time : LocalDateTime
-type : RequestType
-request_id : int
+setCommiters_ID(id : int) : void
+getCommiters_ID() : int
+setReceivers_ID(id : int) : void
+getReceivers_ID() : int
+setIsApproved(state : boolean) : void
+getIsApproved() : boolean
+setSent_Time(date :LocalDateTime) : void
+getSent_Time() : LocalDateTime
+setType(type : RequestType) : void
+getType() : RequestType
+setRequest_ID(id : int) : void
+getRequest_ID() : int
0_*
ProctorTAFromFacultiesRequest
-isInFaculty : boolean
-requestedTAs: List<Pair<Faculty,Department>,Integer>
-exam :Exam
+setIsInFaculty(state : boolean) : void
+getIsInFaculty() : boolean
+setRequestedTAs(list : List<Pair<Faculty,Department>,Integer>) : void
+getRequestedTAs() : List<Pair<Faculty,Department>,Integer>
Notification
-sent_time : LocalDateTime
-notification_information : Str
+setSent_time(date : LocalDateTime) : void
+getSemt_time() : LocalDateTime
+setNotification_information(info : Str) : void
+getNotification_information() : Str
0_*
1
<<enumeration>> RequestType
Swap
SwapEnable
Leave
TransferProctoring
VolunteerProctoring
ProctorTAFromFaculties
TAFromOtherFaculty
TAInFaculty
TAFromOtherFacultyRequest
-requestedTAs : List<Pair<Faculty,Department>,Integer>
-exams_list : List<Exam>
+setExamsList(list : List<Exam>) : void
+setRequestedTAs(list : List<Pair<Faculty,Department>,Integer>) : void
+getRequestedTAs() : List<Pair<Faculty,Department>,Integer>
+getExamsList() : List<Exam>
Log
-id : long
-date : LocalDateTime
-commiters_id : int
-log_info : str
+getId() : Long
+setId(log_id : Long) : void
+getDate() : LocalDateTime
+setDate(date : LocalDateTime) : void
+getCommitersId(id : int) : void
+setCommitersId(id : commiterId) : void
+getInfo() : str
+setInfo(info : str) : void
0_*
RequestService
+sendRequest(request : Request) : boolean
+createRequest(type : RequestType) : Request
+approveRequest(request_id : int) : boolean
+rejectRequest(request_id : int) : boolean
1
Department
-courseList : List<Course>
-instructorList : List<Instructor>
-depOffice : DepartmentOffice
-TAs_List : List<TA>
-department_id : int
+setCourseList(list : List<Course>) : void
+getCourseList() : List<Course>

The **Command Design Pattern** is applied in this system to encapsulate various types of user- and system-initiated actions—such as Swap, VolunteerProctoring, and TransferProctoring—as independent command objects. This architectural approach decouples the sender of a request from the object that performs the action, leading to a modular, extensible, and maintainable system structure.

Pattern Components in the Class Diagram:

- **Command Interface:**
  The abstract Request class serves as the base interface for all command types. It contains

shared properties and methods (e.g., request_id, sent_time, isApproved) that define the structure and metadata of a command.

- **Concrete Commands:**
  Subclasses such as SwapRequest, OpenVolunteerProctoringRequest, and ProctorTAFromFacultiesRequest implement specific command logic corresponding to different request types.
- **Invoker:**
  The RequestService class functions as the invoker. It is responsible for triggering commands, validating them, and forwarding them to appropriate receivers. For example, it handles sendRequest(), approveRequest(), and rejectRequest() calls.
- **Receivers:**
  Classes such as TAService and DepartmentOfficeService act as receivers. They execute the core business logic associated with each request.

Motivations for Using Command Pattern:

- **Encapsulation:**
  Each request is encapsulated in an object, making it possible to queue, log, delay, or undo operations.
- **Separation of Concerns:**
  The creation and execution of requests are cleanly separated, improving testability and modularity.
- **Extensibility:**
  New request types can be easily added by extending the Request class, without modifying existing logic.

Example Use Case:

When RequestService.sendRequest(request) is invoked with an instance of SwapRequest, the system treats it generically as a Request. It then delegates processing to the corresponding receiver (e.g., TAService.respondToSwapRequest(...)), which executes the domain-specific logic. This abstraction ensures uniform request handling while preserving flexibility and reducing coupling.

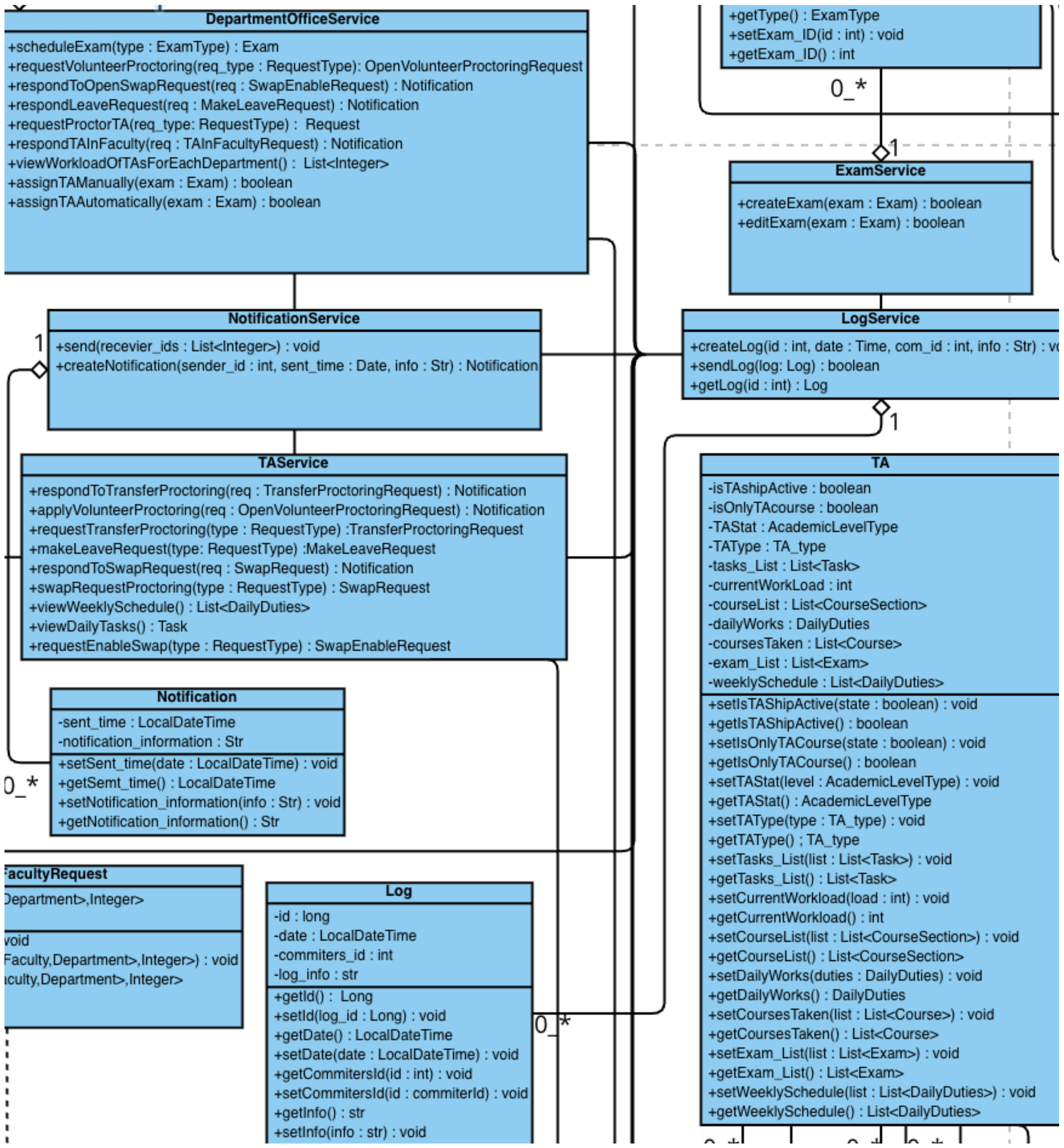
Spring Boot Adaptation:

In a Spring Boot application, this pattern is commonly implemented using @RestController endpoints that accept command objects (e.g., SwapRequestCommand) via @RequestBody. The service layer acts as the invoker, while business services handle the execution:

```
@PostMapping("/swap")
public ResponseEntity<?> swap(@RequestBody SwapRequestCommand command) {
   requestService.sendRequest(command);
   return ResponseEntity.ok().build();
}
```

This approach can be further extended using custom handler classes (in CQRS-based architectures) or Spring's event publishing model for asynchronous command execution.

2. Singleton Design Pattern for Service Classes


DepartmentOfficeService
+scheduleExam(type : ExamType) : Exam
+requestVolunteerProctoring(req_type : RequestType): OpenVolunteerProctoringRequest
+respondToOpenSwapRequest(req : SwapEnableRequest) : Notification
+respondLeaveRequest(req : MakeLeaveRequest) : Notification
+requestProctorTA(req_type: RequestType) : Request
+respondTAInFaculty(req : TAInFacultyRequest) : Notification
+viewWorkloadOfTAsForEachDepartment() : List<Integer>
+assignTAManually(exam : Exam) : boolean
+assignTAAutomatically(exam : Exam) : boolean
+getType() : ExamType
+setExam_ID(id : int) : void
+getExam_ID() : int
0_*
1
ExamService
+createExam(exam : Exam) : boolean
+editExam(exam : Exam) : boolean
NotificationService
+send(recevier_ids : List<Integer>) : void
+createNotification(sender_id : int, sent_time : Date, info : Str) : Notification
1
LogService
+createLog(id : int, date : Time, com_id : int, info : Str) : vo
+sendLog(log: Log) : boolean
+getLog(id : int) : Log
1
TAService
+respondToTransferProctoring(req : TransferProctoringRequest) : Notification
+applyVolunteerProctoring(req : OpenVolunteerProctoringRequest) : Notification
+requestTransferProctoring(type : RequestType) :TransferProctoringRequest
+makeLeaveRequest(type: RequestType) :MakeLeaveRequest
+respondToSwapRequest(req : SwapRequest) : Notification
+swapRequestProctoring(type : RequestType) : SwapRequest
+viewWeeklySchedule() : List<DailyDuties>
+viewDailyTasks() : Task
+requestEnableSwap(type : RequestType) : SwapEnableRequest
TA
-isTAshipActive : boolean
-isOnlyTAcourse : boolean
-TAStat : AcademicLevelType
-TAType : TA_type
-tasks_List : List<Task>
-currentWorkLoad : int
-courseList : List<CourseSection>
-dailyWorks : DailyDuties
-coursesTaken : List<Course>
-exam_List : List<Exam>
-weeklySchedule : List<DailyDuties>
+setIsTAShipActive(state : boolean) : void
+getIsTAShipActive() : boolean
+setIsOnlyTACourse(state : boolean) : void
+getIsOnlyTACourse() : boolean
+setTAStat(level : AcademicLevelType) : void
+getTAStat() : AcademicLevelType
+setTAType(type : TA_type) : void
+getTAType() ; TA_type
+setTasks_List(list : List<Task>) : void
+getTasks_List() : List<Task>
+setCurrentWorkload(load : int) : void
+getCurrentWorkload() : int
+setCourseList(list : List<CourseSection>) : void
+getCourseList() : List<CourseSection>
+setDailyWorks(duties : DailyDuties) : void
+getDailyWorks() : DailyDuties
+setCoursesTaken(list : List<Course>) : void
+getCoursesTaken() : List<Course>
+setExam_List(list : List<Exam>) : void
+getExam_List() : List<Exam>
+setWeeklySchedule(list : List<DailyDuties>) : void
+getWeeklySchedule() : List<DailyDuties>
Notification
-sent_time : LocalDateTime
-notification_information : Str
+setSent_time(date : LocalDateTime) : void
+getSemt_time() : LocalDateTime
+setNotification_information(info : Str) : void
+getNotification_information() : Str
0_*
acultyRequest
epartment>,Integer>
void
Faculty,Department>,Integer>) : void
culty,Department>,Integer>
Log
-id : long
-date : LocalDateTime
-commiters_id : int
-log_info : str
+getId() : Long
+setId(log_id : Long) : void
+getDate() : LocalDateTime
+setDate(date : LocalDateTime) : void
+getCommitersId(id : int) : void
+setCommitersId(id : commiterId) : void
+getInfo() : str
+setInfo(info : str) : void
0_*

In this system, core service classes such as TAService, ExamService, LogService, NotificationService, and DepartmentOfficeService are conceptually aligned with the **Singleton Design Pattern**. These services coordinate centralized, stateless operations across the application, such as TA assignment, exam scheduling, logging, and notification management. Since these responsibilities require a single source of truth and global access, creating multiple instances would lead to inconsistent behavior and resource overhead.

While the getInstance() method is not explicitly illustrated in the class diagram, the design and usage patterns indicate singleton-like behavior.

Spring Boot Implementation Note:

In Spring Boot, service classes are typically annotated with @Service, and managed as **Singleton Beans** by the Spring IoC container by default. This means each service is instantiated only once and injected wherever required using dependency injection (@Autowired or constructor injection), thereby implicitly enforcing the Singleton pattern.

```
@Service
public class TAService {
  public void respondToSwapRequest(SwapRequest request) {
    // handle swap
  }
}
```

Advantages of Singleton in this context:

- Ensures a single shared instance for centralized service logic.
- Promotes consistency in cross-cutting concerns like logging and messaging.
- Reduces memory footprint and instantiation overhead.
- Enables clean, testable, and modular service injection using Spring's dependency management.