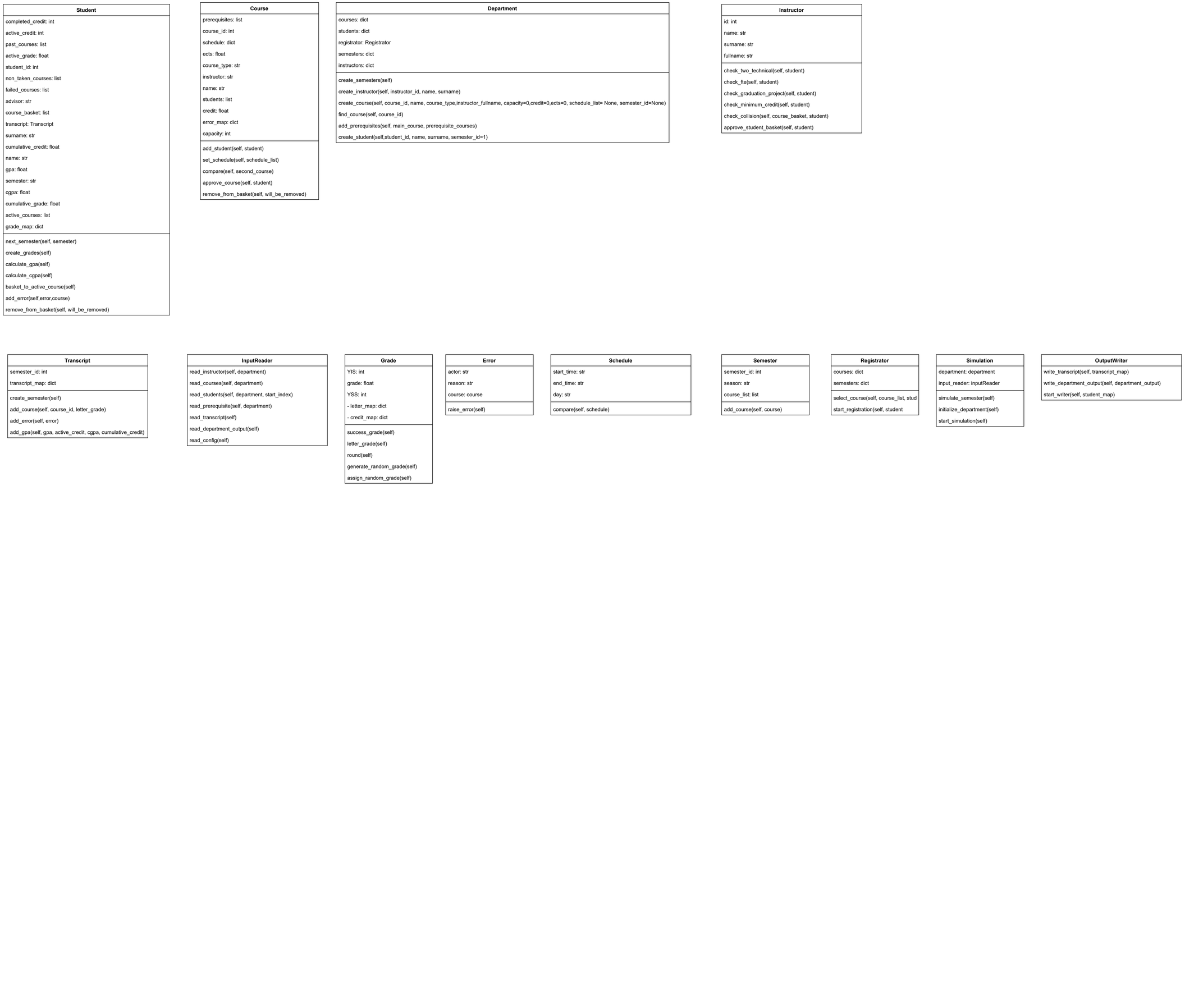
Student
completed_credit: int
active_credit: int
past_courses: list
active_grade: float
student_id: int
non_taken_courses: list
failed_courses: list
advisor: str
course_basket: list
transcript: Transcript
surname: str
cumulative_credit: float
name: str
gpa: float
semester: str
cgpa: float
cumulative_grade: float
active_courses: list
grade_map: dict
next_semester(self, semester)
create_grades(self)
calculate_gpa(self)
calculate_cgpa(self)
basket_to_active_course(self)
add_error(self,error,course)
remove_from_basket(self, will_be_removed)
Course
prerequisites: list
course_id: int
schedule: dict
ects: float
course_type: str
instructor: str
name: str
students: list
credit: float
error_map: dict
capacity: int
add_student(self, student)
set_schedule(self, schedule_list)
compare(self, second_course)
approve_course(self, student)
remove_from_basket(self, will_be_removed)
Department
courses: dict
students: dict
registrator: Registrator
semesters: dict
instructors: dict
create_semesters(self)
create_instructor(self, instructor_id, name, surname)
create_course(self, course_id, name, course_type,instructor_fullname, capacity=0,credit=0,ects=0, schedule_list= None, semester_id=None)
find_course(self, course_id)
add_prerequisites(self, main_course, prerequisite_courses)
create_student(self,student_id, name, surname, semester_id=1)
Instructor
id: int
name: str
surname: str
fullname: str
check_two_technical(self, student)
check_fte(self, student)
check_graduation_project(self, student)
check_minimum_credit(self, student)
check_collision(self, course_basket, student)
approve_student_basket(self, student)
Transcript
semester_id: int
transcript_map: dict
create_semester(self)
add_course(self, course_id, letter_grade)
add_error(self, error)
add_gpa(self, gpa, active_credit, cgpa, cumulative_credit)
InputReader
read_instructor(self, department)
read_courses(self, department)
read_students(self, department, start_index)
read_prerequisite(self, department)
read_transcript(self)
read_department_output(self)
read_config(self)
Grade
YIS: int
grade: float
YSS: int
- letter_map: dict
- credit_map: dict
success_grade(self)
letter_grade(self)
round(self)
generate_random_grade(self)
assign_random_grade(self)
Error
actor: str
reason: str
course: course
raise_error(self)
Schedule
start_time: str
end_time: str
day: str
compare(self, schedule)
Semester
semester_id: int
season: str
course_list: list
add_course(self, course)
Registrator
courses: dict
semesters: dict
select_course(self, course_list, stud
start_registration(self, student
Simulation
department: department
input_reader: inputReader
simulate_semester(self)
initialize_department(self)
start_simulation(self)
OutputWriter
write_transcript(self, transcript_map)
write_department_output(self, department_output)
start_writer(self, student_map)