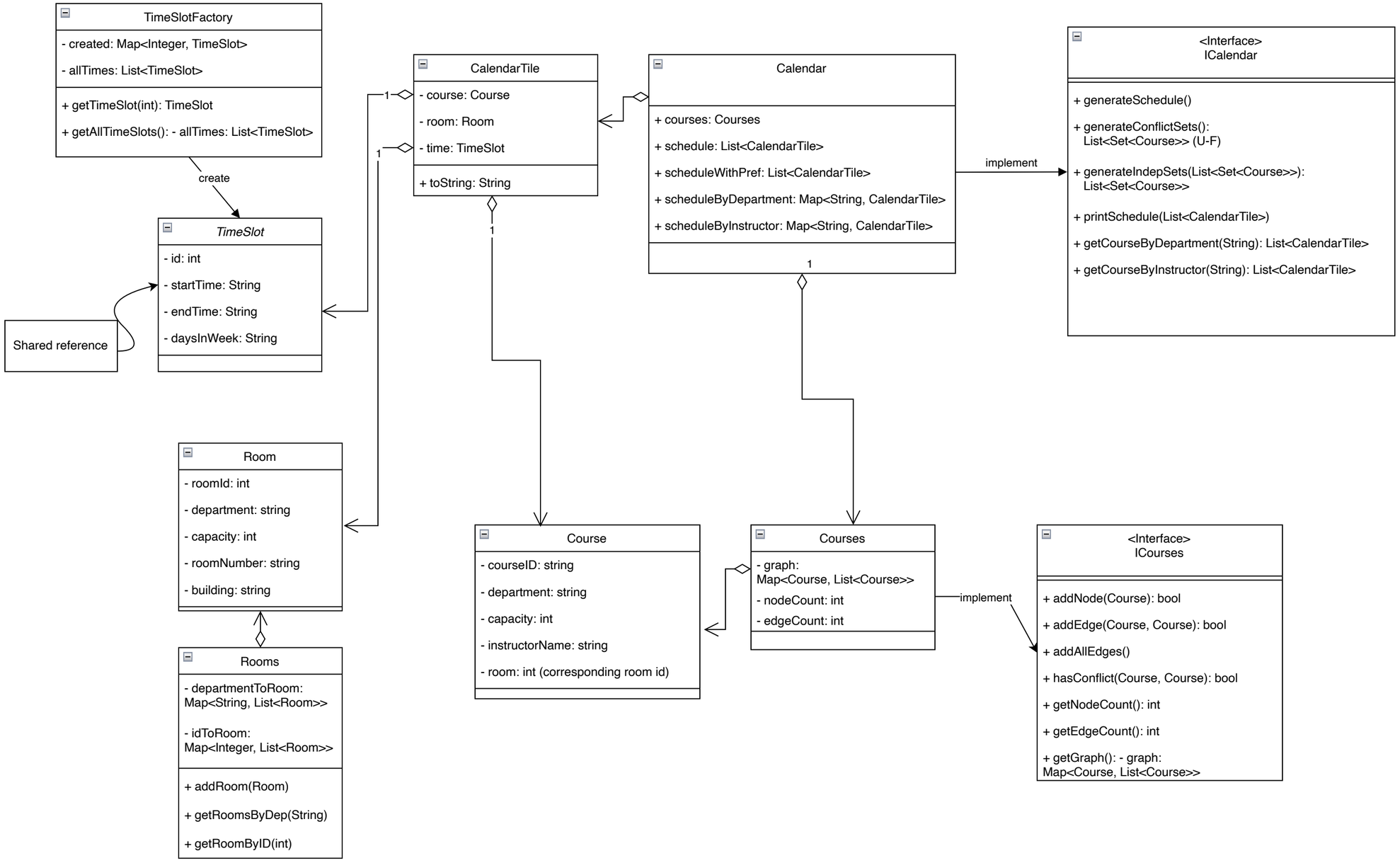
TimeSlotFactory
- created: Map<Integer, TimeSlot>
- allTimes: List<TimeSlot>
+ getTimeSlot(int): TimeSlot
+ getAllTimeSlots(): - allTimes: List<TimeSlot>
create
TimeSlot
- id: int
- startTime: String
- endTime: String
- daysInWeek: String
Shared reference
CalendarTile
- course: Course
- room: Room
- time: TimeSlot
+ toString: String
1
1
1
Calendar
+ courses: Courses
+ schedule: List<CalendarTile>
+ scheduleWithPref: List<CalendarTile>
+ scheduleByDepartment: Map<String, CalendarTile>
+ scheduleByInstructor: Map<String, CalendarTile>
1
implement
<Interface>
ICalendar
+ generateSchedule()
+ generateConflictSets():
List<Set<Course>> (U-F)
+ generateIndepSets(List<Set<Course>>):
List<Set<Course>>
+ printSchedule(List<CalendarTile>)
+ getCourseByDepartment(String): List<CalendarTile>
+ getCourseByInstructor(String): List<CalendarTile>
Room
- roomId: int
- department: string
- capacity: int
- roomNumber: string
- building: string
Rooms
- departmentToRoom:
Map<String, List<Room>>
- idToRoom:
Map<Integer, List<Room>>
+ addRoom(Room)
+ getRoomsByDep(String)
+ getRoomByID(int)
Course
- courseID: string
- department: string
- capacity: int
- instructorName: string
- room: int (corresponding room id)
Courses
- graph:
Map<Course, List<Course>>
- nodeCount: int
- edgeCount: int
implement
<Interface>
ICourses
+ addNode(Course): bool
+ addEdge(Course, Course): bool
+ addAllEdges()
+ hasConflict(Course, Course): bool
+ getNodeCount(): int
+ getEdgeCount(): int
+ getGraph(): - graph:
Map<Course, List<Course>>