Task 1: Identifying Code Metrics

Tool for Computing Code Metrics

CodeMR was chosen as the main tool to compute code metrics, as it could be easily integrated into the current development environment (IntelliJ) and provided a large number of code metrics that would be useful to consider.

Thresholds for the Code Metrics

None of the classes were highly problematic, so we decided to lower the thresholds for some of them in order to better identify which classes to refactor.

TUD Paper on Code Metric Thresholds
http://pure.tudelft.nl/ws/portalfiles/portal/12555765/TUD_SERG_2016_023.pdf

CodeMR Metric Documentations
https://codemr.medium.com/supported-metric-list-by-codemr-a119c3cffc2d
https://www.codemr.co.uk/docs/codemr-intellij-userguide.pdf

Method metric thresholds
McCabe cyclomatic complexity (MCC): 8, we decided on this threshold because in the paper in which McCabe defined cyclomatic complexity he also gave a threshold of 10. Since our project was a bit simpler than an average project, we decided that our threshold should be a bit lower and therefore we put it at 8.

Method lines of code (LOC): 25, as not only will this not fit in everyone's screen, but it will lead to clearer methods that focus on implementing one functionality. Most professional developers / researchers agree with similar thresholds, though of course this depends on the programming language that is being used as some are more verbose than others.

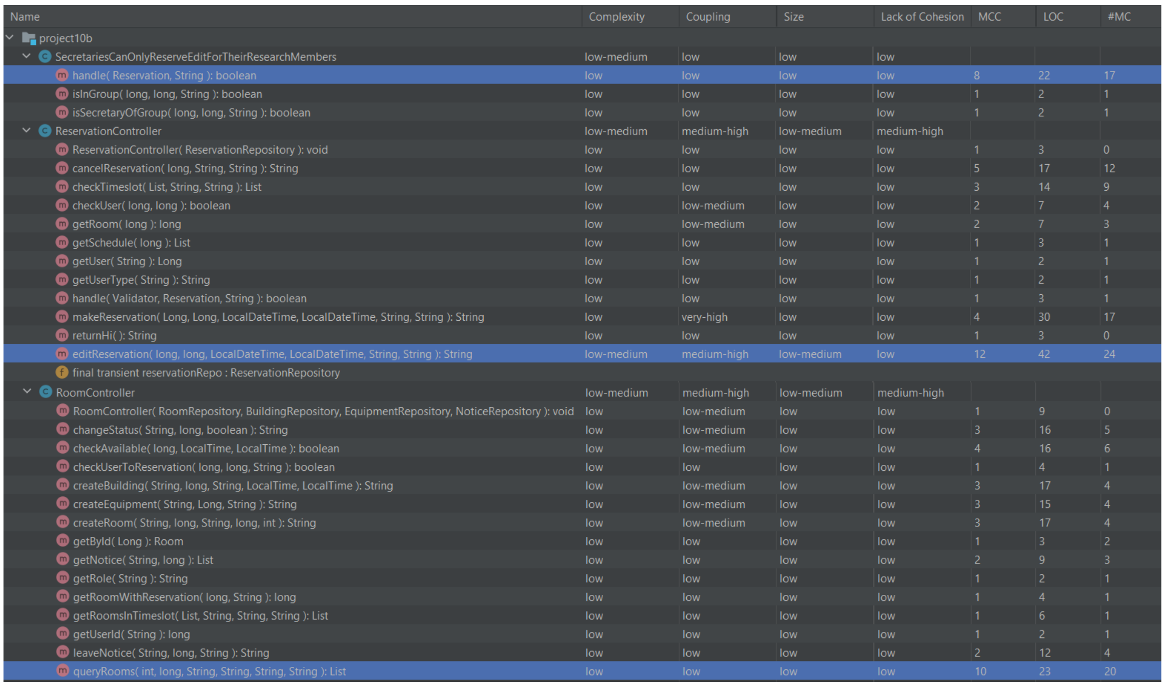
Number of method calls (#MC): 20, as debugging a method with too many separate method calls would result in difficulty to find where the bug is exactly. Too few method calls would also result in difficulties when we also set the Method lines of code to 25, as not all methods could be implemented with these specifics. Ultimately, setting it at a number similar to the lines of code per method seemed like a sane idea, and this is something that is commonly agreed on as well.

Class metric thresholds
Lines of code (LOC): 100. Some more complicated projects might need more lines of code for some Controllers that deal with the many requests that exist, but since this project is "small" and there are not that many requests to any microservice, this seems like a good idea.

Lack of cohesion of methods (LCOM): 0.7, the codeMR documentation with the explanation of the metric says higher than 1 is alarming, but since the classes that had 1+ LCOM were very small and simple classes we decided to lower the threshold to actually identify the classes that needed improvement in cohesion.

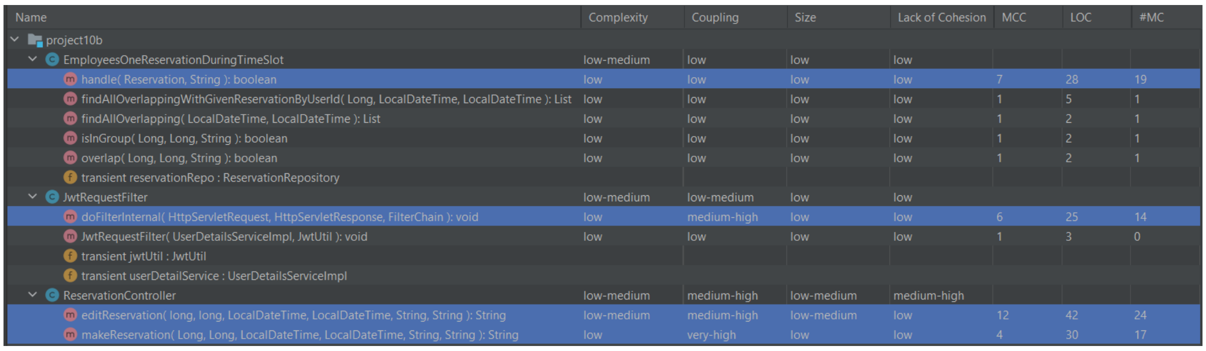
Lack of cohesion among methods (LCAM): 0.7, the codeMR documentation says the field's 'Lack of Cohesion' appears as 'high' when LCAM is greater than 0.8. However, since our project is small and most classes weren't problematic, we decided to lower the threshold to better identify the classes that need refactoring.

Methods that exceed the MCC threshold



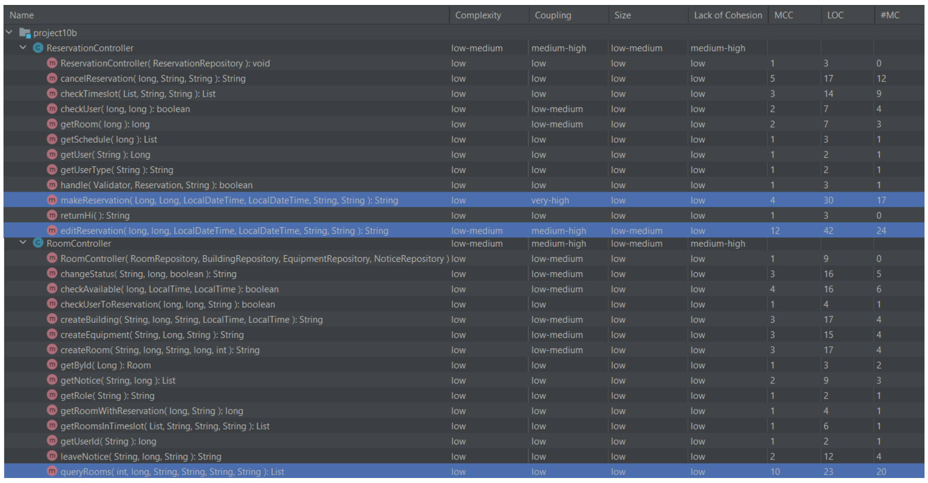
Name	Complexity	Coupling	Size	Lack of Cohesion	MCC	LOC	#MC
project10b							
SecretariesCanOnlyReserveEditForTheirResearchMembers	low-medium	low	low	low			
handle(Reservation, String): boolean	low	low	low	low	8	22	17
isInGroup(long, long, String): boolean	low	low	low	low	1	2	1
isSecretaryOfGroup(long, long, String): boolean	low	low	low	low	1	2	1
ReservationController	low-medium	medium-high	low-medium	medium-high			
ReservationController(ReservationRepository): void	low	low	low	low	1	3	0
cancelReservation(long, String, String): String	low	low	low	low	5	17	12
checkTimeslot(List, String, String): List	low	low	low	low	3	14	9
checkUser(long, long): boolean	low	low-medium	low	low	2	7	4
getRoom(long): long	low	low-medium	low	low	2	7	3
getSchedule(long): List	low	low	low	low	1	3	1
getUser(String): Long	low	low	low	low	1	2	1
getUserType(String): String	low	low	low	low	1	2	1
handle(Validator, Reservation, String): boolean	low	low	low	low	1	3	1
makeReservation(Long, Long, LocalDateTime, LocalDateTime, String, String): String	low	very-high	low	low	4	30	17
returnHi( ): String	low	low	low	low	1	3	0
editReservation(long, long, LocalDateTime, LocalDateTime, String, String): String	low-medium	medium-high	low-medium	low	12	42	24
final transient reservationRepo : ReservationRepository							
RoomController	low-medium	medium-high	low-medium	medium-high			
RoomController(RoomRepository, BuildingRepository, EquipmentRepository, NoticeRepository): void	low	low-medium	low	low	1	9	0
changeStatus(String, long, boolean): String	low	low-medium	low	low	3	16	5
checkAvailable(long, LocalTime, LocalTime): boolean	low	low-medium	low	low	4	16	6
checkUserToReservation(long, long, String): boolean	low	low	low	low	1	4	1
createBuilding(String, long, String, LocalTime, LocalTime): String	low	low-medium	low	low	3	17	4
createEquipment(String, Long, String): String	low	low-medium	low	low	3	15	4
createRoom(String, long, String, long, int): String	low	low-medium	low	low	3	17	4
getById(Long): Room	low	low	low	low	1	3	2
getNotice(String, long): List	low	low	low	low	2	9	3
getRole(String): String	low	low	low	low	1	2	1
getRoomWithReservation(long, String): long	low	low	low	low	1	4	1
getRoomsInTimeslot(List, String, String, String): List	low	low	low	low	1	6	1
getUserId(String): long	low	low	low	low	1	2	1
leaveNotice(String, long, String): String	low	low	low	low	2	12	4
queryRooms(int, long, String, String, String, String): List	low	low	low	low	10	23	20

Methods that exceed the LOC threshold



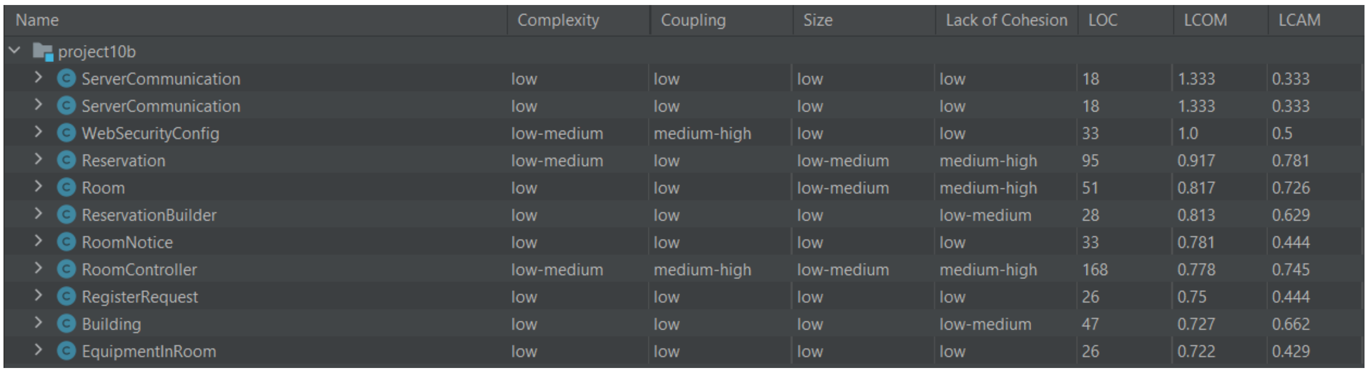
Name	Complexity	Coupling	Size	Lack of Cohesion	MCC	LOC	#MC
project10b							
EmployeesOneReservationDuringTimeSlot	low-medium	low	low	low			
handle(Reservation, String): boolean	low	low	low	low	7	28	19
findAllOverlappingWithGivenReservationByUserId(Long, LocalDateTime, LocalDateTime): List	low	low	low	low	1	5	1
findAllOverlapping(LocalDateTime, LocalDateTime): List	low	low	low	low	1	2	1
isInGroup(Long, Long, String): boolean	low	low	low	low	1	2	1
overlap(Long, Long, String): boolean	low	low	low	low	1	2	1
transient reservationRepo : ReservationRepository							
JwtRequestFilter	low-medium	low-medium	low	low			
doFilterInternal(HttpServletRequest, HttpServletResponse, FilterChain): void	low	medium-high	low	low	6	25	14
JwtRequestFilter(UserDetailsServiceImpl, JwtUtil): void	low	low	low	low	1	3	0
transient jwtUtil : JwtUtil							
transient userDetailService : UserDetailsServiceImpl							
ReservationController	low-medium	medium-high	low-medium	medium-high			
editReservation(long, long, LocalDateTime, LocalDateTime, String, String): String	low-medium	medium-high	low-medium	low	12	42	24
makeReservation(Long, Long, LocalDateTime, LocalDateTime, String, String): String	low	very-high	low	low	4	30	17

Methods that exceed the #MC threshold



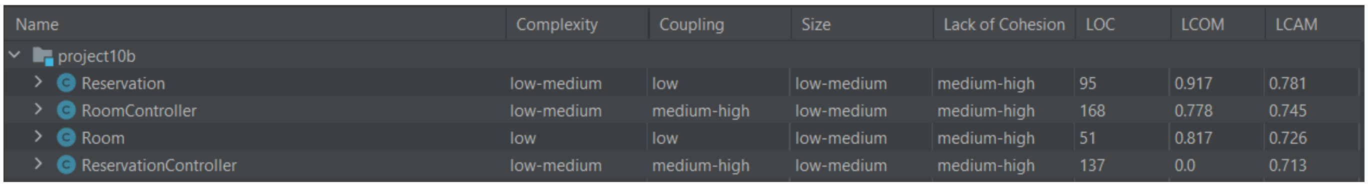
Name	Complexity	Coupling	Size	Lack of Cohesion	MCC	LOC	#MC
project10b							
ReservationController	low-medium	medium-high	low-medium	medium-high			
ReservationController(ReservationRepository): void	low	low	low	low	1	3	0
cancelReservation(long, String, String): String	low	low	low	low	5	17	12
checkTimeslot(List, String, String): List	low	low	low	low	3	14	9
checkUser(long, long): boolean	low	low-medium	low	low	2	7	4
getRoom(long): long	low	low-medium	low	low	2	7	3
getSchedule(long): List	low	low	low	low	1	3	1
getUser(String): Long	low	low	low	low	1	2	1
getUserType(String): String	low	low	low	low	1	2	1
handle(Validator, Reservation, String): boolean	low	low	low	low	1	3	1
makeReservation(Long, Long, LocalDateTime, LocalDateTime, String, String): String	low	very-high	low	low	4	30	17
returnHi( ): String	low	low	low	low	1	3	0
editReservation(long, long, LocalDateTime, LocalDateTime, String, String): String	low-medium	medium-high	low-medium	low	12	42	24
RoomController	low-medium	medium-high	low-medium	medium-high			
RoomController(RoomRepository, BuildingRepository, EquipmentRepository, NoticeRepository)	low	low-medium	low	low	1	9	0
changeStatus(String, long, boolean): String	low	low-medium	low	low	3	16	5
checkAvailable(long, LocalTime, LocalTime): boolean	low	low-medium	low	low	4	16	6
checkUserToReservation(long, long, String): boolean	low	low	low	low	1	4	1
createBuilding(String, long, String, LocalTime, LocalTime): String	low	low-medium	low	low	3	17	4
createEquipment(String, Long, String): String	low	low-medium	low	low	3	15	4
createRoom(String, long, String, long, int): String	low	low-medium	low	low	3	17	4
getById(Long): Room	low	low	low	low	1	3	2
getNotice(String, long): List	low	low	low	low	2	9	3
getRole(String): String	low	low	low	low	1	2	1
getRoomWithReservation(long, String): long	low	low	low	low	1	4	1
getRoomsInTimeslot(List, String, String, String): List	low	low	low	low	1	6	1
getUserId(String): long	low	low	low	low	1	2	1
leaveNotice(String, long, String): String	low	low	low	low	2	12	4
queryRooms(int, long, String, String, String, String): List	low	low	low	low	10	23	20

Classes that exceed the LCOM threshold



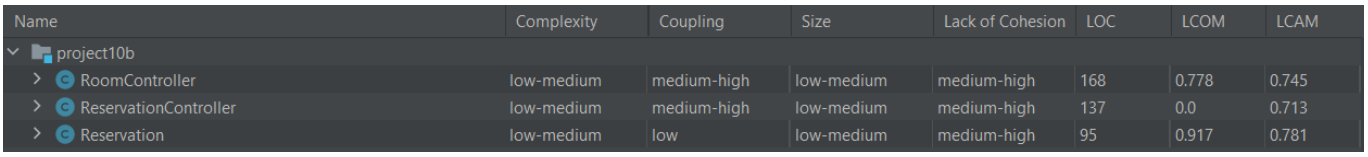
Name	Complexity	Coupling	Size	Lack of Cohesion	LOC	LCOM	LCAM
project10b							
ServerCommunication	low	low	low	low	18	1.333	0.333
ServerCommunication	low	low	low	low	18	1.333	0.333
WebSecurityConfig	low-medium	medium-high	low	low	33	1.0	0.5
Reservation	low-medium	low	low-medium	medium-high	95	0.917	0.781
Room	low	low	low-medium	medium-high	51	0.817	0.726
ReservationBuilder	low	low	low	low-medium	28	0.813	0.629
RoomNotice	low	low	low	low	33	0.781	0.444
RoomController	low-medium	medium-high	low-medium	medium-high	168	0.778	0.745
RegisterRequest	low	low	low	low	26	0.75	0.444
Building	low	low	low	low-medium	47	0.727	0.662
EquipmentInRoom	low	low	low	low	26	0.722	0.429

Classes that exceed the LCAM threshold



Name	Complexity	Coupling	Size	Lack of Cohesion	LOC	LCOM	LCAM
project10b							
Reservation	low-medium	low	low-medium	medium-high	95	0.917	0.781
RoomController	low-medium	medium-high	low-medium	medium-high	168	0.778	0.745
Room	low	low	low-medium	medium-high	51	0.817	0.726
ReservationController	low-medium	medium-high	low-medium	medium-high	137	0.0	0.713

Classes that exceed the LOC threshold



Name	Complexity	Coupling	Size	Lack of Cohesion	LOC	LCOM	LCAM
project10b							
RoomController	low-medium	medium-high	low-medium	medium-high	168	0.778	0.745
ReservationController	low-medium	medium-high	low-medium	medium-high	137	0.0	0.713
Reservation	low-medium	low	low-medium	medium-high	95	0.917	0.781

Among the code with metrics that exceeded our thresholds, we picked 5 methods and 5 classes based on how much they were improvable and how much they needed refactoring. Since we had planned the architecture from early on we did not have as many classes to refactor, so we chose some class refactoring operations we already used while developing.

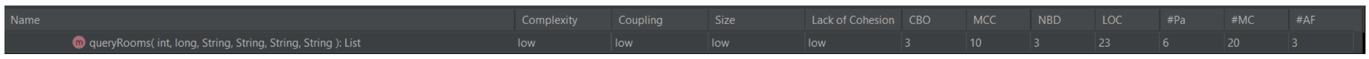
Task 2: Code Refactoring

Method Refactoring

Method 1 (RoomController.queryRooms)

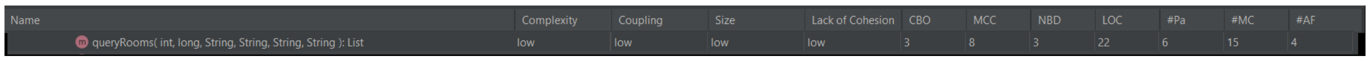
Code Smell: McCabe Cyclomatic Complexity (MCC) and the number of method calls (#MC)

Metric before refactoring: MCC 10, #MC 20



Name	Complexity	Coupling	Size	Lack of Cohesion	CBO	MCC	NBD	LOC	#Pa	#MC	#AF
queryRooms(int, long, String, String, String, String): List	low	low	low	low	3	10	3	23	6	20	3

Metric after refactoring: MCC 8, #MC 15



Name	Complexity	Coupling	Size	Lack of Cohesion	CBO	MCC	NBD	LOC	#Pa	#MC	#AF
queryRooms(int, long, String, String, String, String): List	low	low	low	low	3	8	3	22	6	15	4

Description of fix: We focused on improving the McCabe Cyclomatic Complexity (MCC) and the number of method calls (#MC).

We fixed this method with the extract method refactoring operation applied on the filtering on equipment. This used to be done like the other filtering with if statements and it also gathered a list of valid roomIds first. The refactored version however puts almost all of the checks and method calls used here outside of queryRooms. The findAllByEquipmentName was practically completely changed to now handle all the method calls that were used in queryRooms to map the result of the old version of this method to an attribute itself. Extracting this to a method and out of the other filtering code also allowed us to improve the MCC by instead of filtering on equipment making the list we used to filter on an already filtered list using the new findAllByEquipmentName method if the method should filter on equipment.

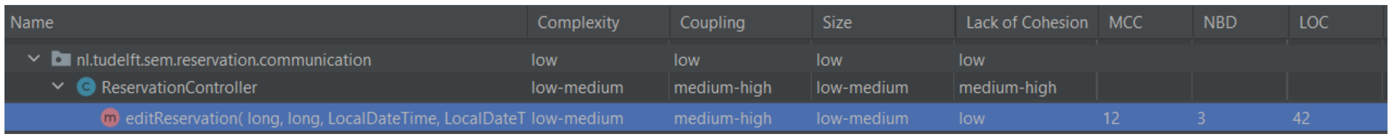
Another slight improvement to the MCC was done by removing an if statement that due to the or statement partnered if clause and the nature of the value they handled shouldn't ever be able to change the outcome of the or statement.

The MCC score is now on our threshold but the method can't be improved upon further without splitting it up in different methods which it already is where possible and any further splits would need more loops over the same objects which would be a lot slower therefore we left it at 8.

Method 2 (ReservationController.editReservation)

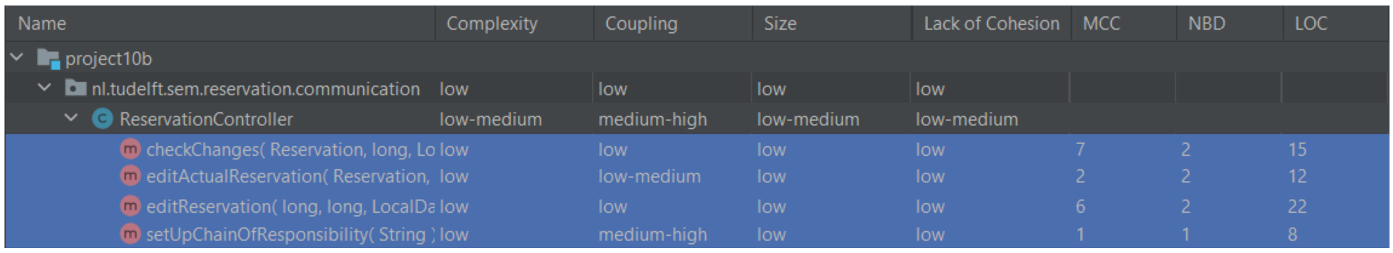
Code smell: Long method

Metric before refactoring: 12(MCC), 42(LOC)



Name	Complexity	Coupling	Size	Lack of Cohesion	MCC	NBD	LOC
nl.tudelft.sem.reservation.communication	low	low	low	low			
ReservationController	low-medium	medium-high	low-medium	medium-high			
editReservation(long, long, LocalDateTime, LocalDateT	low-medium	medium-high	low-medium	low	12	3	42

Metric after refactoring: 6(MCC), 22(LOC)



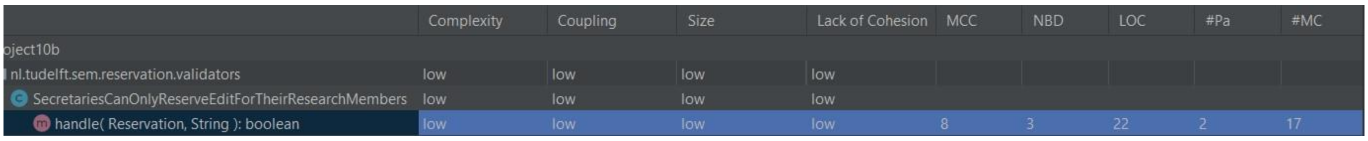
Name	Complexity	Coupling	Size	Lack of Cohesion	MCC	NBD	LOC
project10b							
nl.tudelft.sem.reservation.communication	low	low	low	low			
ReservationController	low-medium	medium-high	low-medium	low-medium			
checkChanges(Reservation, long, Lo	low	low	low	low	7	2	15
editActualReservation(Reservation,	low	low-medium	low	low	2	2	12
editReservation(long, long, LocalDa	low	low	low	low	6	2	22
setUpChainOfResponsibility(String)	low	medium-high	low	low	1	1	8

Description of fix: We focused on reducing the 'McCabe Cyclomatic Complexity' (MCC) and 'Lines of Code' (LOC). We fixed the method by the extract method refactoring operation. The editReservation() was a long and complex method, having 42 lines, but we identified the big chunks of it and split it into 4 separate methods. Now the chain of validators are set at the setUpChainOfResponsibility() method, the changes to be made are checked at checkChanges(), and the actual reservation in the database is edited in the editActualReservation() method, while before all of this was simply handled in the editReservation() method. By doing this we decreased the cyclomatic complexity as the if clauses were separated into different methods, and also reduced the maximum line of code in one method.

Method 3 (SecretariesCanOnlyReserveEditForTheirResearchMembers.handle)

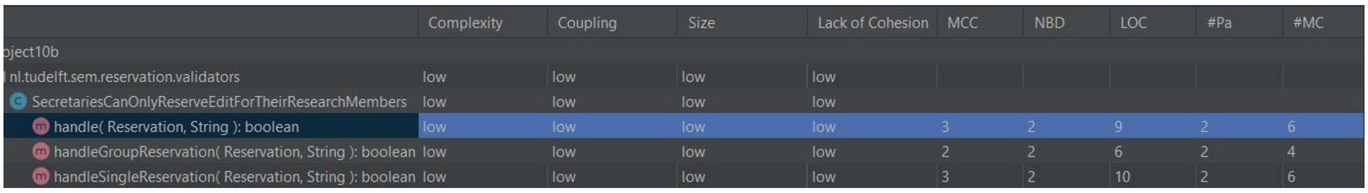
Code smell: Long method

Metric before refactoring:



	Complexity	Coupling	Size	Lack of Cohesion	MCC	NBD	LOC	#Pa	#MC
oject10b									
nl.tudelft.sem.reservation.validators	low	low	low	low					
SecretariesCanOnlyReserveEditForTheirResearchMembers	low	low	low	low					
handle(Reservation, String): boolean	low	low	low	low	8	3	22	2	17

Metric (after):



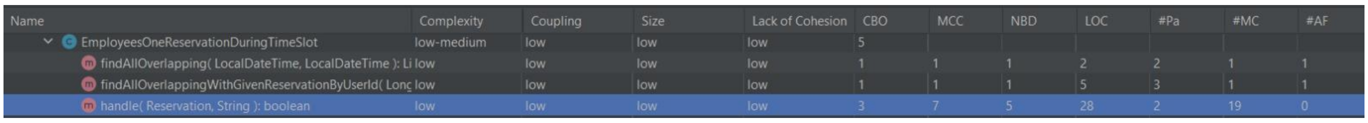
	Complexity	Coupling	Size	Lack of Cohesion	MCC	NBD	LOC	#Pa	#MC
oject10b									
nl.tudelft.sem.reservation.validators	low	low	low	low					
SecretariesCanOnlyReserveEditForTheirResearchMembers	low	low	low	low					
handle(Reservation, String): boolean	low	low	low	low	3	2	9	2	6
handleGroupReservation(Reservation, String): boolean	low	low	low	low	2	2	6	2	4
handleSingleReservation(Reservation, String): boolean	low	low	low	low	3	2	10	2	6

Description of fix: We focused on reducing the McCabe Cyclomatic Complexity (MCC) and Lines Of Code (LOC). We fixed the method using the extract method refactoring operation. handle() had completely different behavior for each type of reservation, separated by if-blocks. These separate pieces of code have been divided into dedicated methods, lowering the MCC and LOC of the handle method. We also optimized the code a little. It used to throw an error if the reservation has no valid type, but since the builder ensures that is impossible, this has been removed. Now, if the reservation is not of type SINGLE or GROUP, it is assumed to either be of type SELF or ADMIN, and thus the validator passes. This further reduced MCC and LOC.

Method EmployeesOneReservationDuringTimeSlot.handle()

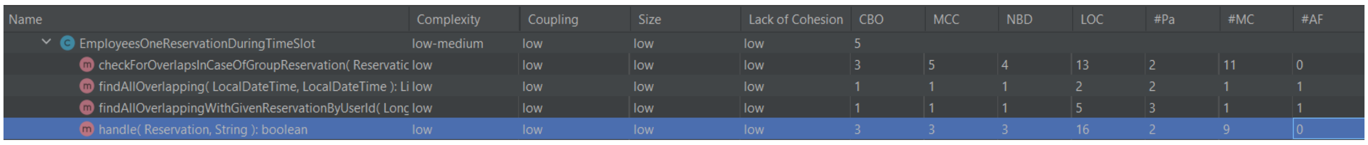
Code smell: Long method

Metric before refactoring: 7 MMC, 28 LOC, 19 #MC



Name	Complexity	Coupling	Size	Lack of Cohesion	CBO	MCC	NBD	LOC	#Pa	#MC	#AF
EmployeesOneReservationDuringTimeSlot	low-medium	low	low	low	5						
findAllOverlapping(LocalDateTime, LocalDateTime): Li	low	low	low	low	1	1	1	2	2	1	1
findAllOverlappingWithGivenReservationByUserId(Lonc	low	low	low	low	1	1	1	5	3	1	1
handle(Reservation, String): boolean	low	low	low	low	3	7	5	28	2	19	0

Metric after refactoring: 3 MCC, 16 LOC, 9 #MC



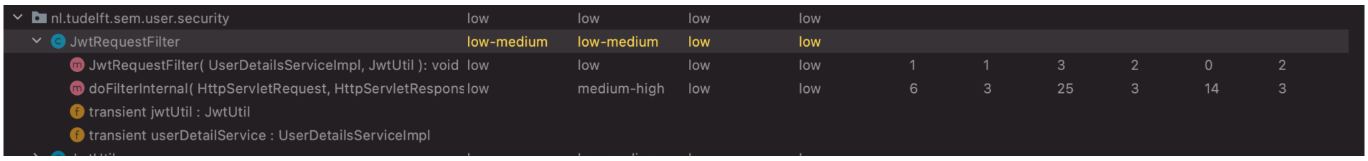
Name	Complexity	Coupling	Size	Lack of Cohesion	CBO	MCC	NBD	LOC	#Pa	#MC	#AF
EmployeesOneReservationDuringTimeSlot	low-medium	low	low	low	5						
checkForOverlapsInCaseOfGroupReservation(Reservatic	low	low	low	low	3	5	4	13	2	11	0
findAllOverlapping(LocalDateTime, LocalDateTime): Li	low	low	low	low	1	1	1	2	2	1	1
findAllOverlappingWithGivenReservationByUserId(Lonc	low	low	low	low	1	1	1	5	3	1	1
handle(Reservation, String): boolean	low	low	low	low	3	3	3	16	2	9	0

Description of fix: We focused on reducing the McCabe Cyclomatic Complexity (MCC) and Lines Of Code (LOC), since the method was long, hard to understand and to have an overview of what checks are happening where. We fixed the method using the extract method refactoring operation, applied on the logic of the code related to the checks for overlapping reservations when the reservation the user wants to make is for a research group. We extracted that logic and put it in a dedicated method called checkForOverlapsInCaseOfGroupReservation(...), as it is responsible for a separate functionality and would make the code more understandable. By doing that the handle() method became easier to grasp and the MCC was reduced significantly, as well as the LOC and even the #MC.

Method 5 (JwtRequestFilter.doFilterInternal)

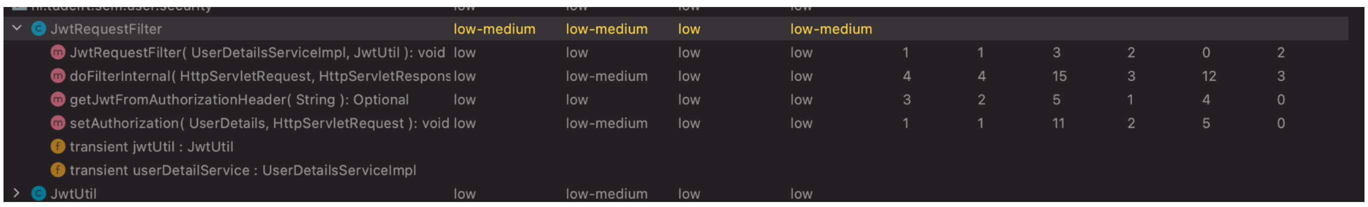
Code smell: Long Method

Metric before refactoring: 6 MMC and 25 LOC



nl.tudelft.sem.user.security	low	low	low	low						
JwtRequestFilter	low-medium	low-medium	low	low						
JwtRequestFilter(UserDetailsServiceImpl, JwtUtil): void	low	low	low	low	1	1	3	2	0	2
doFilterInternal(HttpServletRequest, HttpServletRespons	low	medium-high	low	low	6	3	25	3	14	3
transient jwtUtil : JwtUtil										
transient userDetailService : UserDetailsServiceImpl										

Metric after refactoring: 4 MMC and 15 LOC



JwtRequestFilter	low-medium	low-medium	low	low-medium						
JwtRequestFilter(UserDetailsServiceImpl, JwtUtil): void	low	low	low	low	1	1	3	2	0	2
doFilterInternal(HttpServletRequest, HttpServletRespons	low	low-medium	low	low	4	4	15	3	12	3
getJwtFromAuthorizationHeader(String): Optional	low	low	low	low	3	2	5	1	4	0
setAuthorization(UserDetails, HttpServletRequest): void	low	low-medium	low	low	1	1	11	2	5	0
transient jwtUtil : JwtUtil										
transient userDetailService : UserDetailsServiceImpl										
JwtUtil	low	low-medium	low	low						

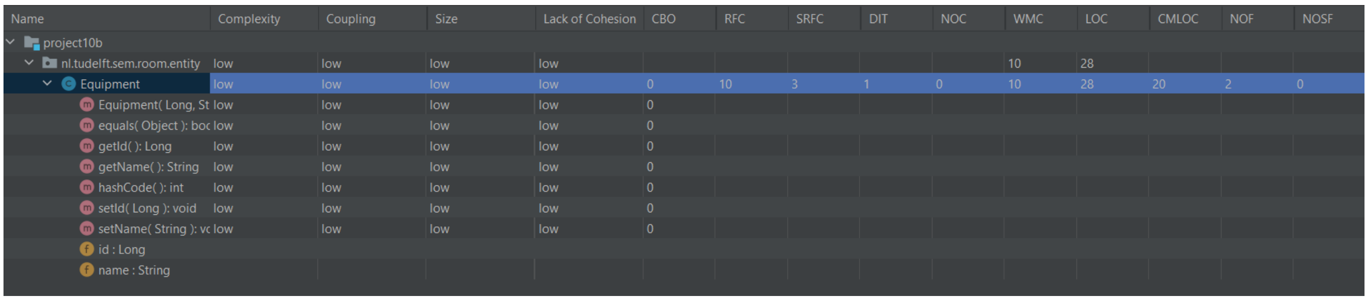
Description of fix: To reduce both the McCabe Cyclomatic Complexity (MCC) and Lines Of Code(LOC) of this method, a focus was put on applying Extract Method Refactoring. It was clear that some of the parts of the method had very distinct functionalities, thus these were separated into their own methods, namely 'getJwtFromAuthorizationToken' and 'setAuthorization'. Not only is the code now 10 lines shorter and with lower complexity, but it is simpler to understand, as the first thing the code does now is getting the jwt token from the header, and then only in a certain condition will a call to 'setAuthorization' be made. Both of these methods could now be expanded individually, if for example a new header had to be added when authorizing a user.

Class Refactoring

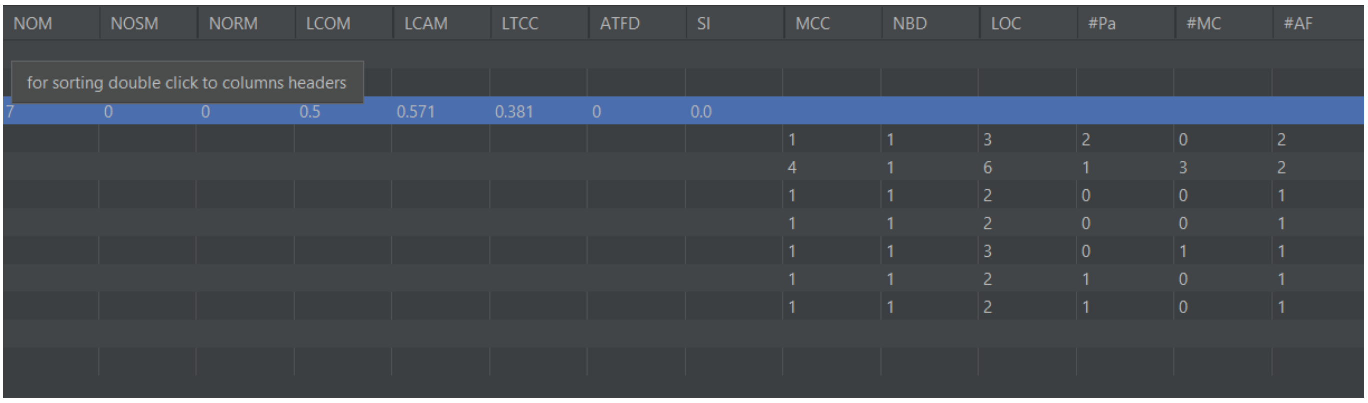
Class 1 (Equipment & EquipmentInRoom)

Code smell: Dispensable

Metric before refactoring: (first the Equipment metric and second the EquipmentInRoom metric)

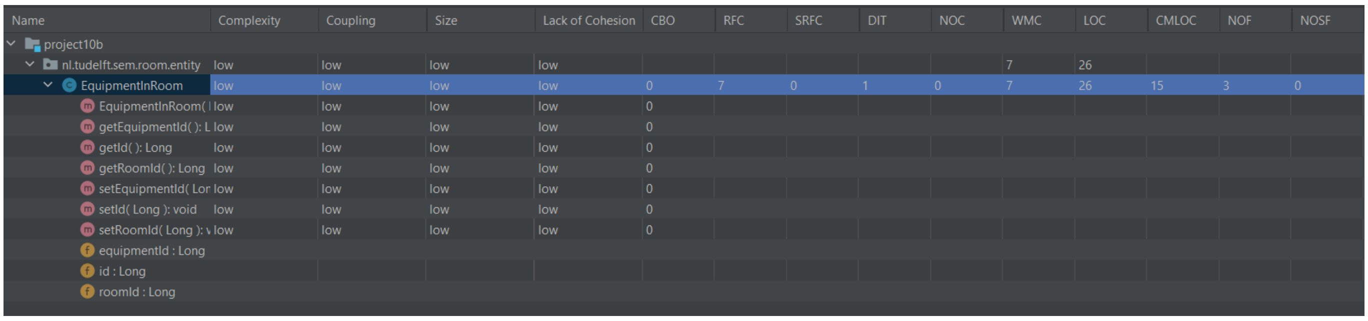


Name	Complexity	Coupling	Size	Lack of Cohesion	CBO	RFC	SRFC	DIT	NOC	WMC	LOC	CMLOC	NOF	NOSF
project10b														
nl.tudelft.sem.room.entity	low	low	low	low						10	28			
Equipment	low	low	low	low	0	10	3	1	0	10	28	20	2	0
Equipment(Long, St	low	low	low	low	0									
equals(Object): boo	low	low	low	low	0									
getId( ): Long	low	low	low	low	0									
getName( ): String	low	low	low	low	0									
hashCode( ): int	low	low	low	low	0									
setId(Long): void	low	low	low	low	0									
setName(String): vc	low	low	low	low	0									
id : Long														
name : String														

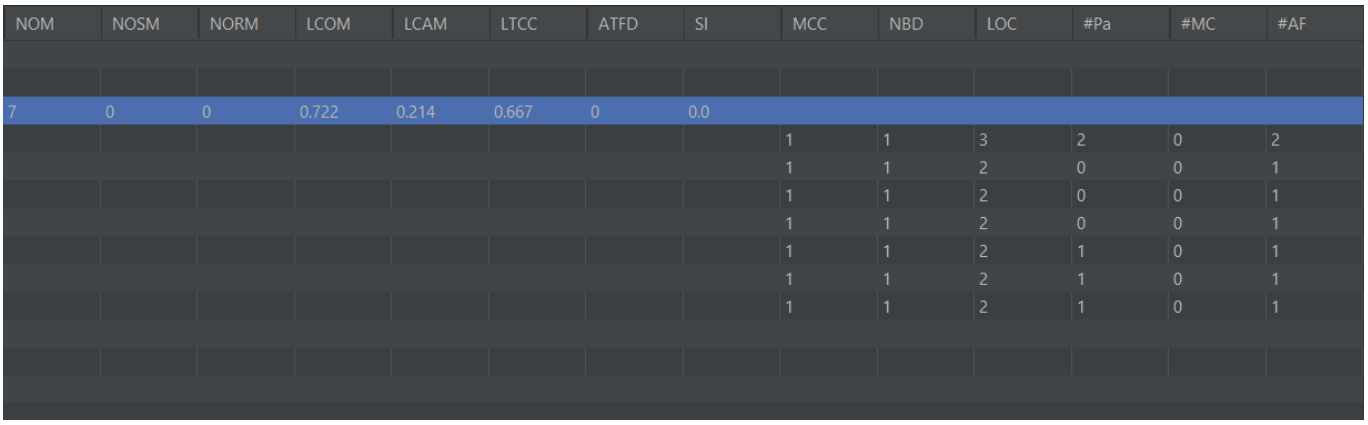


for sorting double click to columns headers

NOM	NOSM	NORM	LCOM	LCAM	LTCC	ATFD	SI	MCC	NBD	LOC	#Pa	#MC	#AF
7	0	0	0.5	0.571	0.381	0	0.0						
								1	1	3	2	0	2
								4	1	6	1	3	2
								1	1	2	0	0	1
								1	1	2	0	0	1
								1	1	3	0	1	1
								1	1	2	1	0	1
								1	1	2	1	0	1

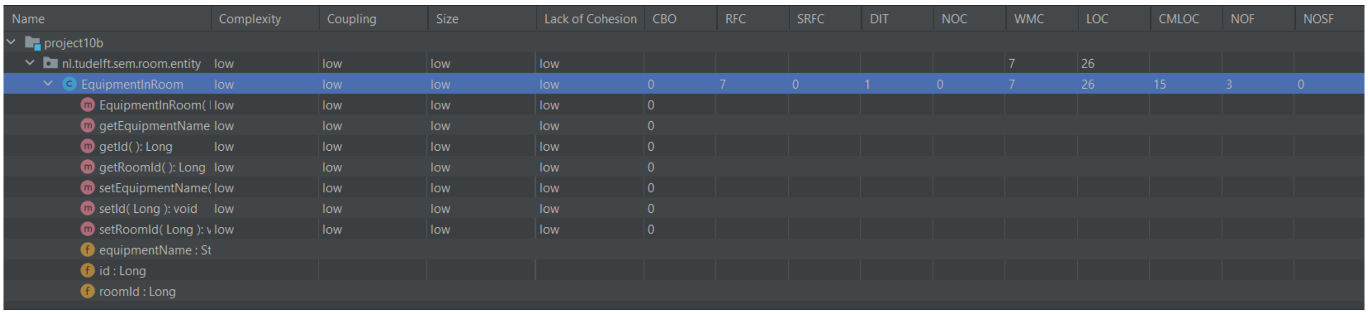


Name	Complexity	Coupling	Size	Lack of Cohesion	CBO	RFC	SRFC	DIT	NOC	WMC	LOC	CMLOC	NOF	NOSF
project10b														
nl.tudelft.sem.room.entity	low	low	low	low						7	26			
EquipmentInRoom	low	low	low	low	0	7	0	1	0	7	26	15	3	0
EquipmentInRoom(I	low	low	low	low	0									
getEquipmentId( ): L	low	low	low	low	0									
getId( ): Long	low	low	low	low	0									
getRoomId( ): Long	low	low	low	low	0									
setEquipmentId(Lor	low	low	low	low	0									
setId(Long): void	low	low	low	low	0									
setRoomId(Long): v	low	low	low	low	0									
equipmentId : Long														
id : Long														
roomId : Long														

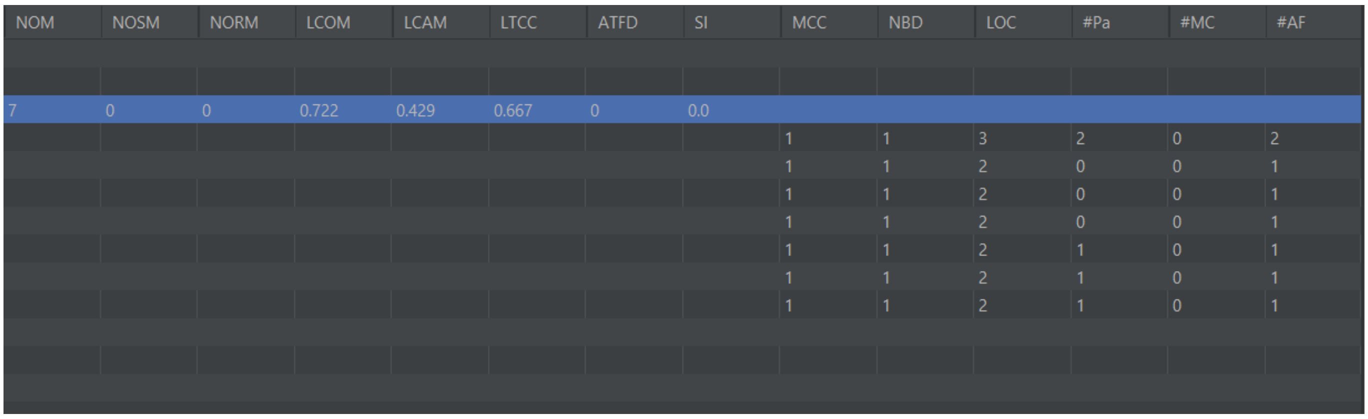


NOM	NOSM	NORM	LCOM	LCAM	LTCC	ATFD	SI	MCC	NBD	LOC	#Pa	#MC	#AF
7	0	0	0.722	0.214	0.667	0	0.0						
								1	1	3	2	0	2
								1	1	2	0	0	1
								1	1	2	0	0	1
								1	1	2	0	0	1
								1	1	2	1	0	1
								1	1	2	1	0	1
								1	1	2	1	0	1

Metric after refactoring: (EquipmentInRoom metric)



Name	Complexity	Coupling	Size	Lack of Cohesion	CBO	RFC	SRFC	DIT	NOC	WMC	LOC	CMLOC	NOF	NOSF
project10b														
nl.tudelft.sem.room.entity	low	low	low	low						7	26			
EquipmentInRoom	low	low	low	low	0	7	0	1	0	7	26	15	3	0
EquipmentInRoom(I	low	low	low	low	0									
getEquipmentName	low	low	low	low	0									
getId( ): Long	low	low	low	low	0									
getRoomId( ): Long	low	low	low	low	0									
setEquipmentName(	low	low	low	low	0									
setId(Long): void	low	low	low	low	0									
setRoomId(Long): v	low	low	low	low	0									
equipmentName : St														
id : Long														
roomId : Long														



NOM	NOSM	NORM	LCOM	LCAM	LTCC	ATFD	SI	MCC	NBD	LOC	#Pa	#MC	#AF
7	0	0	0.722	0.429	0.667	0	0.0						
								1	1	3	2	0	2
								1	1	2	0	0	1
								1	1	2	0	0	1
								1	1	2	0	0	1
								1	1	2	1	0	1
								1	1	2	1	0	1
								1	1	2	1	0	1

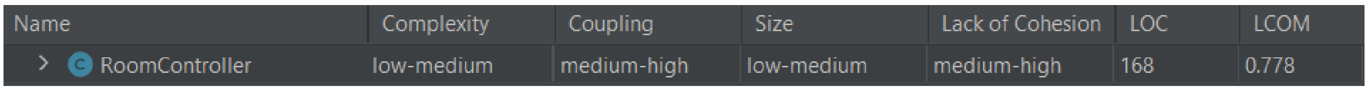
Description of fix: The equipment class was removed because it was dispensable since it only had an id and name field and some basic object methods. The only place equipment was used back then was in EquipmentInRoom since these classes worked together. EquipmentInRoom used to have an id for itself and one for the room and the equipment it was linking but now it just uses the name of the equipment instead of the id. So what we eventually did was kind of like the move method operation but instead for the entire class functionalities.

As seen from the metrics we were able to keep the same metrics for the EquipmentInRoom while having put all the functionality of Equipment in there and Equipment be removed.

Class 2 (RoomController)

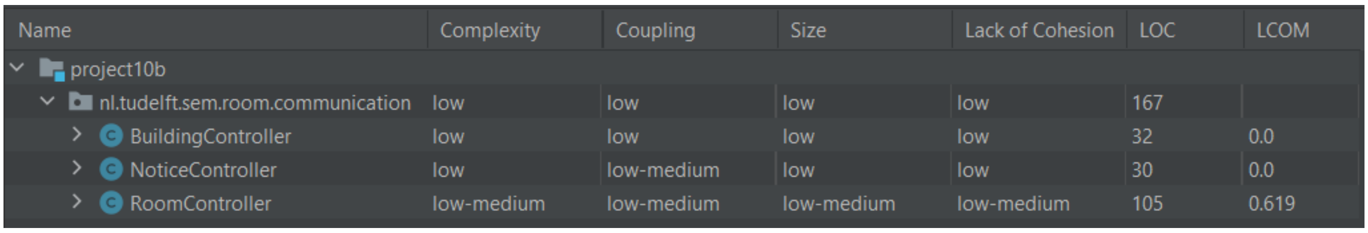
Code smell: Blob Class

Metric before refactoring: 0.778(LCOM)



Name	Complexity	Coupling	Size	Lack of Cohesion	LOC	LCOM
RoomController	low-medium	medium-high	low-medium	medium-high	168	0.778

Metric after refactoring: 0.619(LCOM)



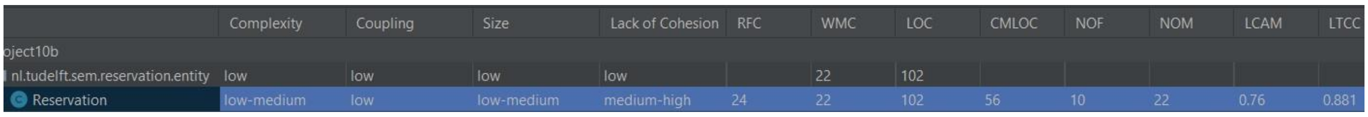
Name	Complexity	Coupling	Size	Lack of Cohesion	LOC	LCOM
project10b						
nl.tudelft.sem.room.communication	low	low	low	low	167	
BuildingController	low	low	low	low	32	0.0
NoticeController	low	low-medium	low	low	30	0.0
RoomController	low-medium	low-medium	low-medium	low-medium	105	0.619

Description of fix: We focused on reducing ‘Lack of Cohesion of Methods’(LCOM) by the extract class refactoring operation. Since the RoomController class was responsible for every HTTP request to the room service, we split them into 3 classes, each dealing with one or two entities. As a result, the repository attributes that were used a few times were moved to separate controllers and improved the cohesion of classes (reduced LCOM to pass the 0.7 threshold).

Class 3 (Reservation -> Builder)

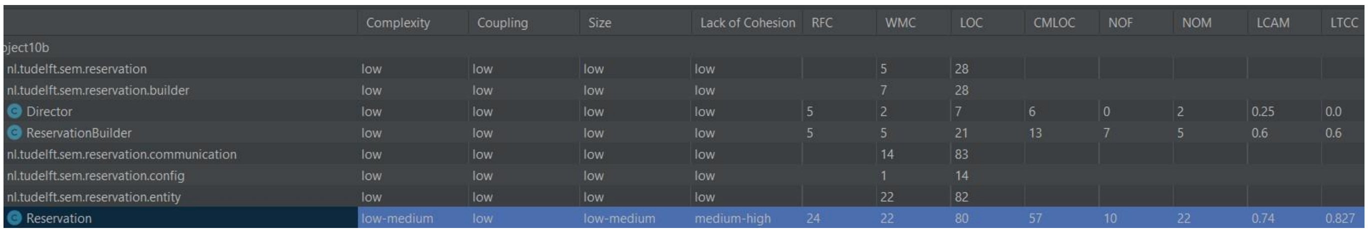
Code smell: Blob Class

Metric before refactoring:



	Complexity	Coupling	Size	Lack of Cohesion	RFC	WMC	LOC	CMLOC	NOF	NOM	LCAM	LTCC
oject10b												
nl.tudelft.sem.reservation.entity	low	low	low	low		22	102					
Reservation	low-medium	low	low-medium	medium-high	24	22	102	56	10	22	0.76	0.881

Metric after refactoring:



	Complexity	Coupling	Size	Lack of Cohesion	RFC	WMC	LOC	CMLOC	NOF	NOM	LCAM	LTCC
oject10b												
nl.tudelft.sem.reservation	low	low	low	low		5	28					
nl.tudelft.sem.reservation.builder	low	low	low	low		7	28					
Director	low	low	low	low	5	2	7	6	0	2	0.25	0.0
ReservationBuilder	low	low	low	low	5	5	21	13	7	5	0.6	0.6
nl.tudelft.sem.reservation.communication	low	low	low	low		14	83					
nl.tudelft.sem.reservation.config	low	low	low	low		1	14					
nl.tudelft.sem.reservation.entity	low	low	low	low		22	82					
Reservation	low-medium	low	low-medium	medium-high	24	22	80	57	10	22	0.74	0.827

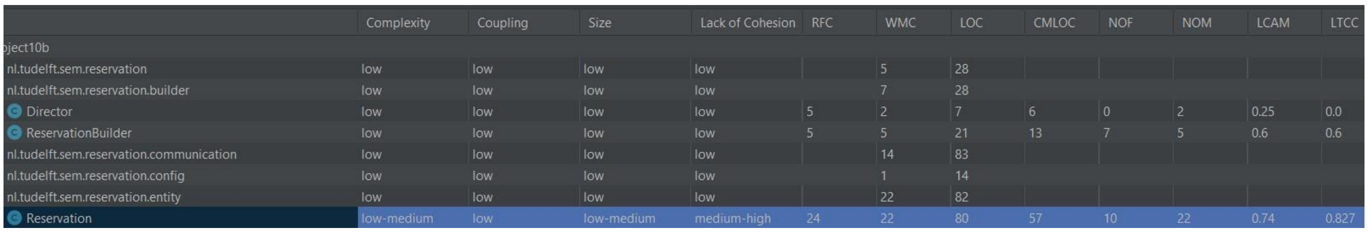
Description of fix:

We focussed on reducing Lines Of Code (LOC) using the extract class refactoring operation. The Reservation class used to contain a builder class, with methods that took a reservation field as an argument and returned a copy of the builder with those fields added to it. This design choice resulted in a very large reservation class, and as such the builder was extracted. This design is also closer to the way builder patterns were described in class. A Builder interface was added later.

Class 4 (Reservation)

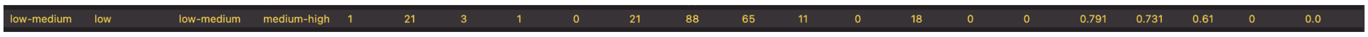
Code smell: Dispensable

Metric before refactoring:0.917 (LCOM)



	Complexity	Coupling	Size	Lack of Cohesion	RFC	WMC	LOC	CMLOC	NOF	NOM	LCAM	LTCC
oject10b												
nl.tudelft.sem.reservation	low	low	low	low		5	28					
nl.tudelft.sem.reservation.builder	low	low	low	low		7	28					
Director	low	low	low	low	5	2	7	6	0	2	0.25	0.0
ReservationBuilder	low	low	low	low	5	5	21	13	7	5	0.6	0.6
nl.tudelft.sem.reservation.communication	low	low	low	low		14	83					
nl.tudelft.sem.reservation.config	low	low	low	low		1	14					
nl.tudelft.sem.reservation.entity	low	low	low	low		22	82					
Reservation	low-medium	low	low-medium	medium-high	24	22	80	57	10	22	0.74	0.827

Metric after refactoring: 0.791 (LCOM)



low-medium	low	low-medium	medium-high	1	21	3	1	0	21	88	65	11	0	18	0	0	0.791	0.731	0.61	0	0.0

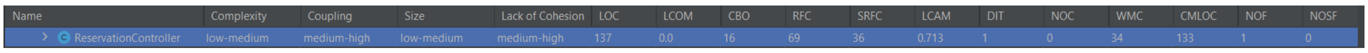
Description of fix:

A focus was made on improving the ‘Lack of Cohesion of Methods (LCOM) in the Reservation entity. Upon investigating the class, it was clear that some of the setters were never used at all, and that some methods, such as ‘changeLocation’ or changeTime’ updated more than one field by calling the setters. Since this method was within the class, using the setters was unnecessary as you could still access them. Due to this, these methods were updated to no longer use the setters, and all unused setters were also removed from the class, resulting in an improved metric for Lack of Cohesion of Methods. Regardless, this metric could not be improved to the threshold (0.7) as it mostly contains getters/setters which of course only depend on one attribute each.

Class 5 (ReservationController)

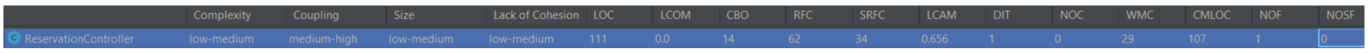
Code smell: Blob Class

Metric before refactoring: 137 LOC and 0.713 LCAM



Name	Complexity	Coupling	Size	Lack of Cohesion	LOC	LCOM	CBO	RFC	SRFC	LCAM	DIT	NOC	WMC	CMLOC	NOF	NOSF
ReservationController	low-medium	medium-high	low-medium	medium-high	137	0.0	16	69	36	0.713	1	0	34	133	1	0

Metric after refactoring: 111 LOC and 0.656 LCAM



	Complexity	Coupling	Size	Lack of Cohesion	LOC	LCOM	CBO	RFC	SRFC	LCAM	DIT	NOC	WMC	CMLOC	NOF	NOSF
ReservationController	low-medium	medium-high	low-medium	low-medium	111	0.0	14	62	34	0.656	1	0	29	107	1	0

Description of fix: The ReservationController class has become very large, so we split up the logic there by creating two additional classes - UserController and RoomController, where we extracted the related to user and room code correspondingly. This split, as well as the refactoring of the editReservation() method in ReservationController reduces LCAM (passing the 0.7 threshold) and LOC metrics and the class overall becomes smaller, more manageable and with reduced coupling. That would avoid any issues related to the blob/large class code smell in the future.