# Code Review

* [Review the artifacts](#30j0zll)
* [GOAL](#1fob9te)
* [Code Analysis](#3znysh7)
  + [Findbugs](#2et92p0)
* [Design](#tyjcwt)
* [CheckList](#3dy6vkm)
* [CODE Review Activity](#1t3h5sf)
* [History](#4d34og8)

# **Review the artifacts**

* Define relevant metrics

•Defect density: # defects / LOC  ---------------> 10/138 (7.2%)  
•Inspection rate: # LOC / Time unit     -----------------------> 3937 / 8h

* Collect data
* Document anomalies/defects accordingly

                                       -->reservation fee defect , redeem defect , parking spot overlab defect ...etc

                                           static analysis defect

* Share the output of the reviews with Mentor and Prof. Genari.

# **GOAL**

**•Reduce the number of defects in software earlier in the development process**

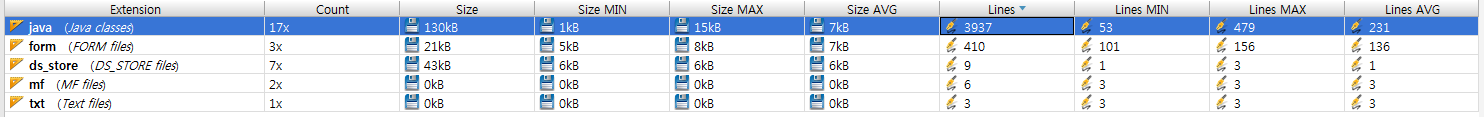
**•Uncover defects that would be difficult or impossible to discover by means of testing**

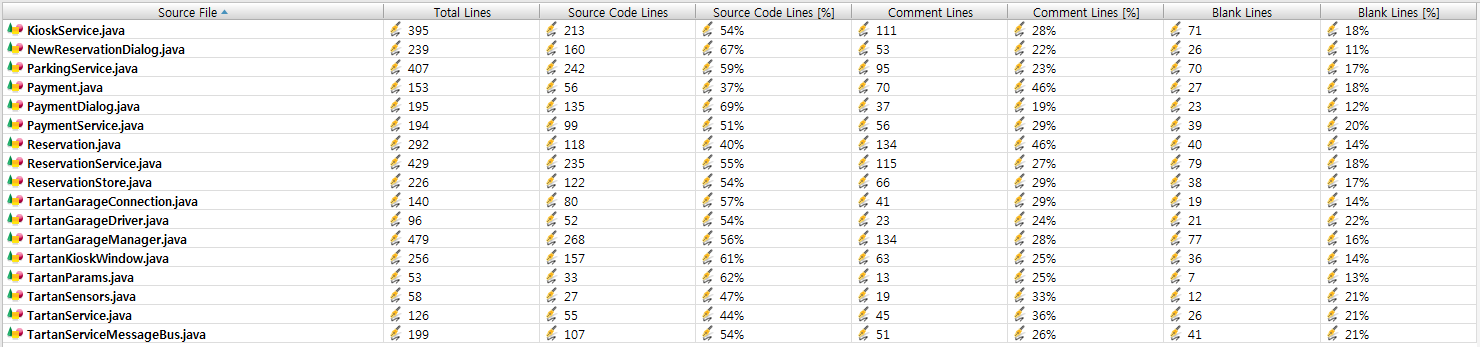
**•Improve learning and communication within the software team**

# **Code Analysis**

**using Statics(intellij plugin)**

**LOC : 3937**



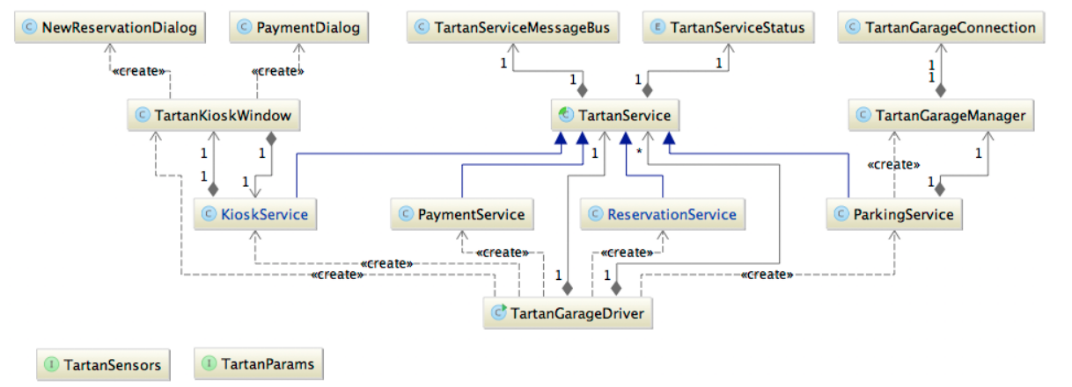


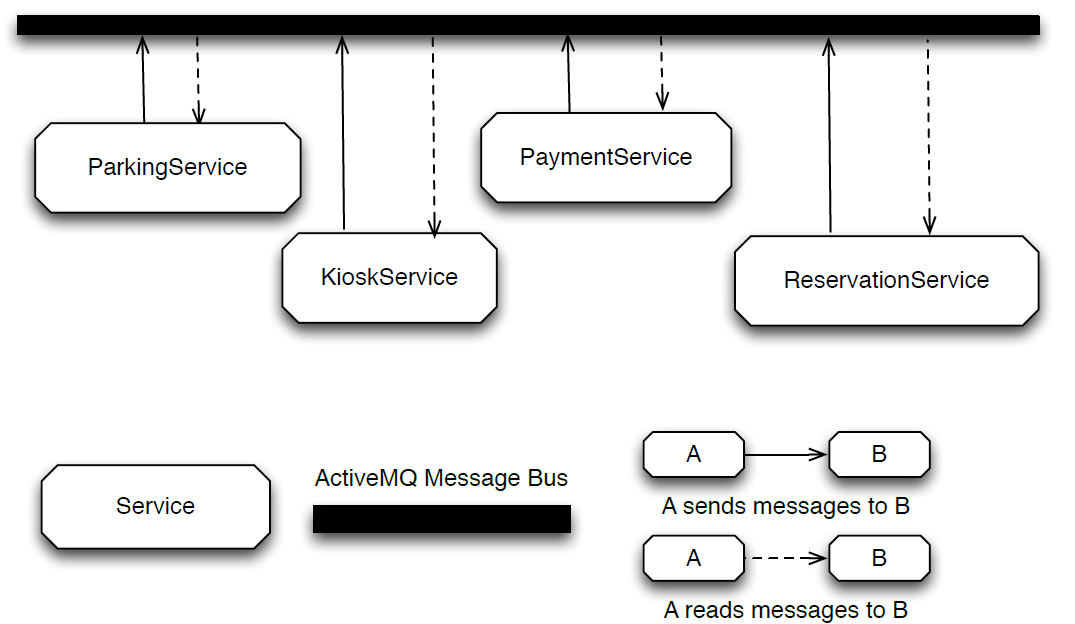
### Findbugs

It is important to find defects in the code without executing the code. We can detect faults in advance through static analysis. We can also find defects that are difficult to detect with dynamic testing. I think static analysis is important because we can also find dependencies and inconsistencies in the software model.

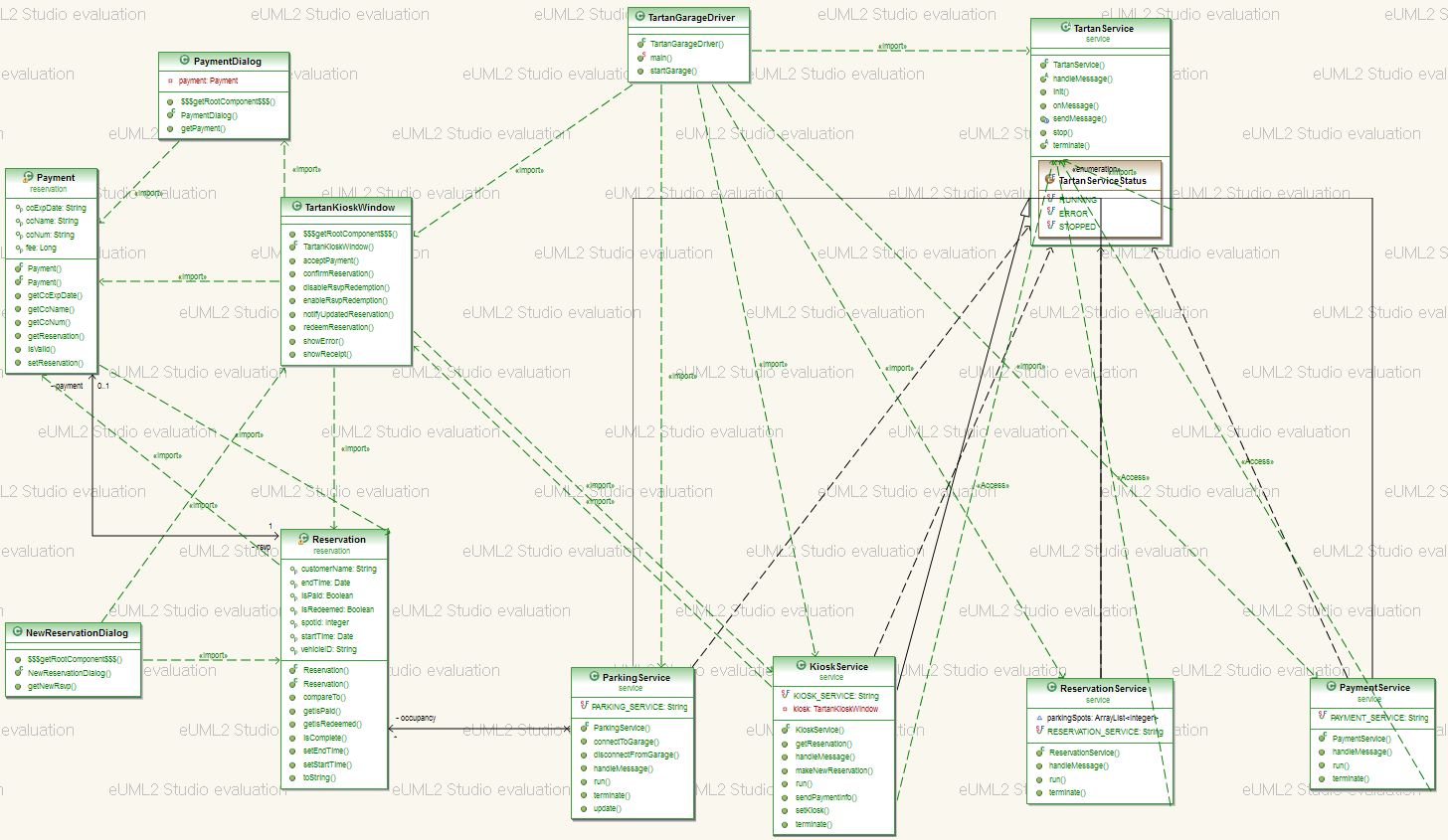
==Static Analysis result==  
TartanGarage.jar:215 Suspicious comparison of Integer references in edu.cmu.tartan.service.ParkingService.handleGarageEntry(Reservation) [Scariest(1), High confidence]  
TartanGarage.jar:303 Possible null pointer dereference of selectedRsvp in edu.cmu.tartan.service.KioskService.handleRedeemReservation(HashMap) [Scary(8), Normal confidence]  
TartanGarage.jar:397 Possible null pointer dereference of results in edu.cmu.tartan.service.ReservationService.handleRedeemReservation(HashMap) [Scary(8), Normal confidence]  
TartanGarage.jar:66 Incorrect lazy initialization and update of static field edu.cmu.tartan.hardware.TartanGarageConnection.connection in edu.cmu.tartan.hardware.TartanGarageConnection.getConnection(String) [Scary(8), Normal confidence]

# **Design**





<TarTan Project Code Diagram Analysis using eUML2>



# CheckList

|  |  |
| --- | --- |
| **category** | **contents** |
| C01 | User can make reservate parkinglot |
| C02 | Reservations can only be redeemed once. |
| C03 | A reservation must be unique within the system by customer name, vehicle ID (i.e. license plate), or time. |
| C04 | Reservations must always be paid for before entry to the garage is granted. |
| C05 | Reservation fees and penalties must be accurate. |
| C06 | Reservations for a parking spot must not overlap. The system should never reserve the same |
| C07 | parking spot for more than one car for the same time period. That is, each parking spot must have at most one reservation per time period. |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| Complete | Verify that the code covers all of the design.  +above checklist |
| Includes | Verify that the includes are complete. |
| Initialization | Check variable and parameter initialization. •at program initiation •at start of every loop •at class/function/procedure entry |
| Names | Check name spelling and use. •Is it consistent? •Is it within the declared scope? •Do all structures and classes use ‘.’ reference? |
| OutputFormat | Check the output format. •Line stepping is proper. •Spacing is proper. |
| ()Pairs | Ensure that () are proper and matched. |
| LogicOperators | •Verify the proper use of ==, =, ||, and so on. •Check every logic function for (). |
| Line-by-linecheck | Check every line of code for •instruction syntax •proper punctuation |
| Standards | Ensure that the code conforms to the coding standards. |
| FileOpenandClose | Verify that all files are •properly declared •opened •closed |

# CODE Review Activity

|  |  |  |
| --- | --- | --- |
| **Category** | **Code** | **Result** |
| Checklist: Complete  Category: C01 | **NewReservationDialog**  Expand source  private void onOK() {  try {  String customerName = nameTextField.getText();  String licensePlate = licensePlateTextField.getText();  String year = String.valueOf(Calendar.getInstance().get(Calendar.YEAR));  SimpleDateFormat parser = new SimpleDateFormat("EEE, MMM dd hh a yyyy");  String start = String.valueOf(startDayComboBox.getSelectedItem()) + " " + String.valueOf(startTimeComboBox.getSelectedItem() + " " + year);  System.out.println(start);  String end = String.valueOf(startDayComboBox.getSelectedItem()) + " " + String.valueOf(endTimeComboBox.getSelectedItem() + " " + year);  Date startDate = parser.parse(start);  Date endDate = parser.parse(end);  if (!customerName.isEmpty() && !licensePlate.isEmpty()) {  rsvp = new Reservation();  rsvp.setCustomerName(customerName);  rsvp.setVehicleID(licensePlate);  rsvp.setStartTime(startDate);  rsvp.setEndTime(endDate);  }  } catch (Exception e) {  e.printStackTrace();  rsvp = null;  }  dispose();  } | java.text.ParseException: Unparseable date: "목, 6월 29 12 AM 2017" at java.text.DateFormat.parse(DateFormat.java:366) at edu.cmu.tartan.NewReservationDialog.onOK(NewReservationDialog.java:131) at edu.cmu.tartan.NewReservationDialog.access$000(NewReservationDialog.java:23) at edu.cmu.tartan.NewReservationDialog$1.actionPerformed(NewReservationDialog.java:52) at javax.swing.AbstractButton.fireActionPerformed(AbstractButton.java:2022) |
| Checklist: Complete  Category: C02    code inspection  Declaration redundancy | **Reservation**  Expand source  public void setIsRedeemed(Boolean r) {  isRedeemed = r;  } | * Actual value of parameter ''r'' is always ''true'' * --> unsatisfied with testcase02 |
| Checklist: LogicOperators | **KioskService**  Expand source  private void handleParkingExit(HashMap<String, Object> message) {  // Boolean state = false;  // if (message.containsKey(TartanParams.PAYLOAD)) {  // HashMap<String, Object> body = (HashMap<String, Object>) message.get(TartanParams.PAYLOAD);  // if (body.containsKey(TartanParams.EXIT\_STATE)) {  // state = (Boolean) body.get(TartanParams.EXIT\_STATE);  // }  // }  String vid = JOptionPane.showInputDialog(kiosk, "Enter vehicle ID to exit", "Exit",JOptionPane.QUESTION\_MESSAGE);  // if (state == false) {  // vid =  // }  if (vid != null || (vid != null && ("".equals(vid) == false))) {  HashMap<String, Object> msg = new HashMap<String, Object>();  msg.put(TartanParams.COMMAND, TartanParams.MSG\_EXIT\_GARAGE);  msg.put(TartanParams.PAYLOAD, vid);  sendMessage(ParkingService.PARKING\_SERVICE, msg);  } else {  JOptionPane.showMessageDialog(kiosk,  "You must enter a valid vehicle ID", "Invalid Vehicle", JOptionPane.ERROR\_MESSAGE);  }  } | * Unnecessary expression is included |
| Checklist: Complete  Category: C03 | **Category**  Expand source  @Override  public int compareTo(Reservation o) {  if (startTime.compareTo(o.getStartTime()) == 0)  if (endTime.compareTo(o.getEndTime()) == 0)  if (customerName.compareTo(o.getCustomerName()) == 0)  if (vehicleID.compareTo(o.getVehicleID()) == 0)  return 0;  return -1;  } | unsatisfied with testcase02  CustormerName or vehicleID can be duplicated |
| Funtion  (PSP) | **ReservationService**  Expand source  private Integer getParkingSpot(Reservation newRsvp) {  ...  // There are no spots available for this time  else if (occupiedSpots.size() > parkingSpots.size()) { /\* FIXME: Maybe logical error, Should use >= for fully occupied slots \*/  return TartanParams.SPOT\_UNAVAILABLE;  }  else {  Collections.sort(occupiedSpots);  spot = occupiedSpots.get(occupiedSpots.size() - 1) + 1; // get the next spot  }  return spot;  } |  |
| Checking  (PSP) | **ParkingService**  Expand source  private void handleExecuteGarageExit(HashMap<String, Object> message) {  Reservation rsvp = (Reservation) message.get(TartanParams.PAYLOAD);  String vid = rsvp.getVehicleID();  Payment payment = rsvp.getPayment();  if (payment.getFee() != null) {  allowGarageExit();  synchronized (occupancy) {  ListIterator<Reservation> iter = occupancy.listIterator();  while (iter.hasNext()) {  if (iter.next().getVehicleID() == vid) { /\* FIXME: vid should compare by using equals() method \*/  iter.remove();  break;  }  }  }  HashMap<String, Object> receipt = new HashMap<String, Object>();  receipt.put(TartanParams.COMMAND, TartanParams.MSG\_EXIT\_COMPLETE);  receipt.put(TartanParams.PAYLOAD, rsvp);  sendMessage(KioskService.KIOSK\_SERVICE, receipt);  }  } |  |
| Funtion  (PSP) | **CategoryKioskService**  Expand source  private void handleRedeemReservation(HashMap<String, Object> request) {  .....  if (!results.isEmpty()) {  Vector<Reservation> validReservations = new Vector<Reservation>();  Date now = Calendar.getInstance().getTime();  // build a list of spots occupied at this time  for (Reservation r : results) {  Date st = r.getStartTime();  Date et = r.getEndTime();  if (now.after(st) && now.before(et)) {  if (r.getIsRedeemed() == false)  System.err.println("Reservation not yet redeemed!");  validReservations.add(r); /\* FIXME: maybe error, only reservation which is not yet redeemed should be added to validReserv \*/  }  ....  } |  |
| Funtion  (PSP) | **PaymentService**  Expand source  private Long computeTotalFee(Reservation rsvp) {  Long rate = setRate();/\* FIXME: Maybe error, Dynamic Rate should be applied based on time range in the reservation \*/  } |  |
| Funtion  (PSP) | **ReservationService**  Expand source   public Vector<Reservation> lookupByVehicle(String name) {  Vector<Reservation> results = new Vector<Reservation>();  for ( Reservation r : reservations) {  if (r.getCustomerName().equals(name)) { /\* FIXME: Maybe Copy & Paste bug, should check vehicle \*/  results.add(r);  }  }  return results;  } |  |
| N/A  (PSP) | **ReservationService**  Expand source  private void handleCreateNewReservation(HashMap<String, Object> request) {  ...  HashMap<String, Object> body = new HashMap<String, Object>(); /\* FIXME: Unused variable \*/  ...  } |  |
|  |  |  |

# History

1. **[2017-06-26] Quality Planning & Role Define**
   1. Attendant: All of Team Member
   2. Activity: Quality Planning, Quality Assurance & Control Activity Decision, Role Define for Quality Planning
      1. Quality Planning Setup Member: Wonil Hwang, Sunghyuk Nah, Chongjae Yoo
      2. Code Review: Kyungman Yoo, Jaeseung Bae, Sungbong Kim
      3. Administration Console Requirements: Boram Kwon
   3. Time: 9:00PM ~ 11:00PM
2. **[2017-06-27] Code Review Planning & Methodology Setup**
   1. Attendant: All of Team Member
   2. Activity: Review Planning, Goal Setup, Review Method Setup, Tool Choice, Output Metric Define
   3. Time: 6:00PM ~ 10:00PM
3. **[2017-06-29] Code Review Activity**
   1. Attendant: All of Team Member
   2. Activity: Code Review
   3. Time: 6:00PM ~ 9:00PM
4. **[2017-07-04] Code Review Activity**
   1. Attendant: All of Team Member
   2. Activity: Code Review
   3. Time: 6:00PM ~ 8:00PM