**Test Cases v0.1**

**ParkWire**

****

***Ακαδημαϊκό Έτος 2021 - 2022***

**Αβραμόπουλος Μιχαήλ - 1067451 - Έτος Δ**

**(Editor)**

[**up1067451@upnet.gr**](mailto:up10@upnet.gr)

**Δεληγιάννη Μυρτώ - 1067389 - Έτος Δ**

**(Editor)**

[**up1067389@upnet.gr**](mailto:up1067389@upnet.gr)

**Κοντογιάννης Γεώργιος - 1070908 - Έτος Δ**

**(Editor)**

[**up1070908@upnet.gr**](mailto:up1070908@upnet.gr)

**Νικολούδης Παναγιώτης - 1067076 - Έτος Δ**

**(Editor)**

[**up1067076@upnet.gr**](mailto:up10@upnet.gr)

**Πανάικας Σωτήριος - 1067412 - Έτος Δ**

**(Editor)**

[**up1067412@upnet.gr**](mailto:up10@upnet.gr)

[**Εισαγωγή**](#_bs6dg2xxijvr) **4**

[**Constructor Parked**](#_lz76bnqnu8us) **5**

[**Μέθοδος Parked it!**](#_fzpn49efflbr) **6**

[**Μέθοδος searchPaid**](#_ib1wb49wdtdx) **7**

[**Μέθοδος setTimeEstimate**](#_jioaa98au8sh) **9**

[**Μέθοδος addNewBusiness**](#_fcrbwr99niwx) **10**

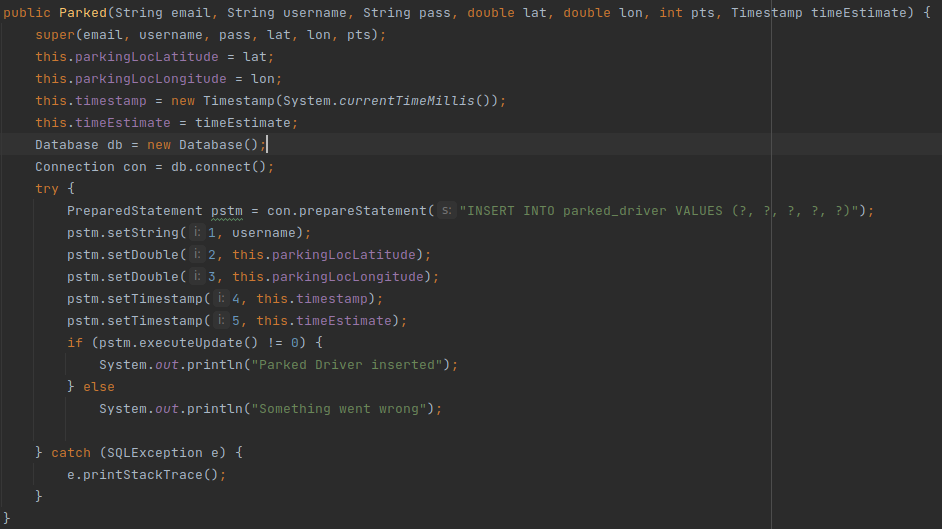
[**Μέθοδος editBusiness**](#_b4xknojx5k53) **11**

# Εισαγωγή

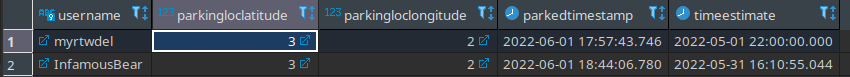
Ως μεθοδολογία για την υλοποίηση των Test Cases παρουσιάζουμε μία σωστή υλοποίηση κώδικα για ορισμένες λειτουργικές μεθόδους του προγράμματος με τα επιθυμητά και αναμενόμενα αποτελέσματα όπως επίσης και μία λανθασμένη η οποία καταλήγει σε error.

# Constructor Parked

Ο constructor της κλάσης Parked κατά την οποία εισάγεται στην βάση ο χρήστης seeking driver ως parked driver.

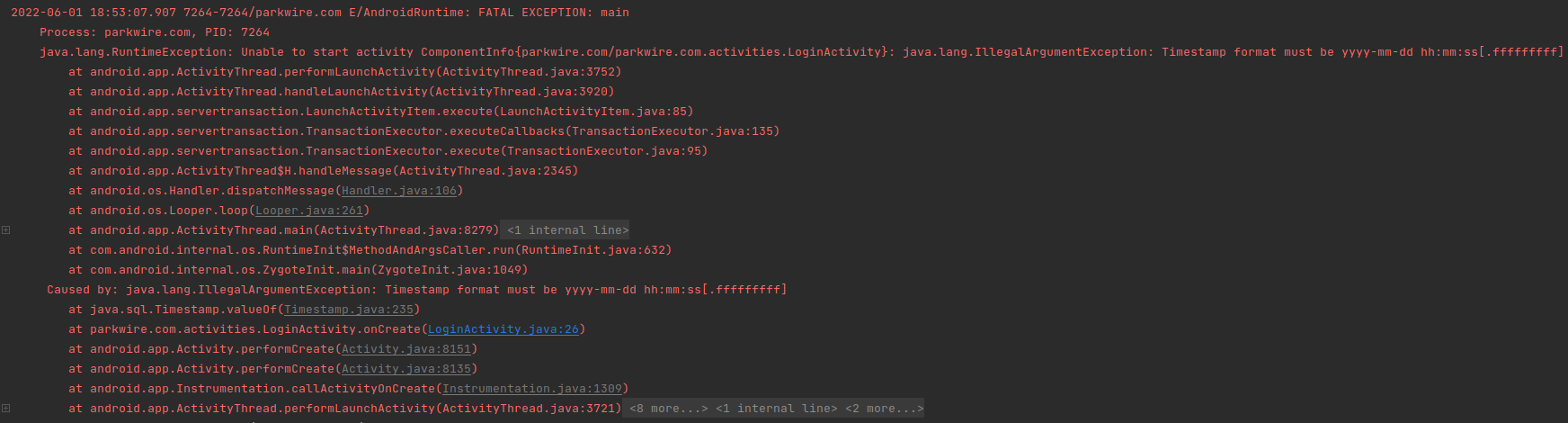






Περίπτωση που το timestamp δεν είναι σε σωστό formatting:



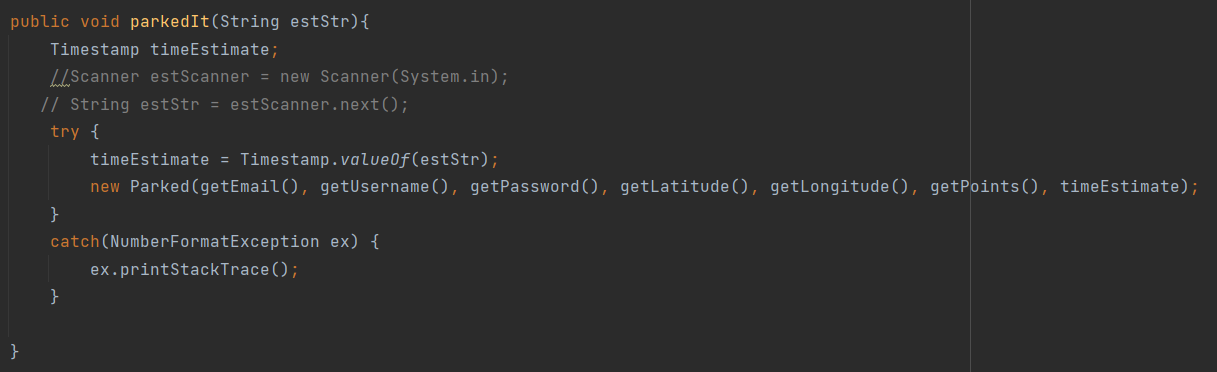


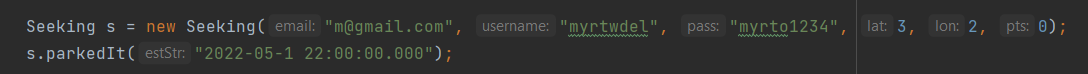
# Μέθοδος Parked it!

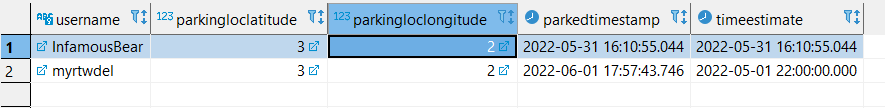
Η parkedIt() είναι μέθοδος της κλάσης Seeking κατά την οποία ο Seeking Driver εισάγει στο σύστημα ότι πάρκαρε επιτυχώς. Στη συνέχεια, το σύστημα δημιουργεί ένα νέο αντικείμενο Parked και ενημερώνει την βάση δεδομένων, δηλαδή ο συγκεκριμένος Driver εισάγεται στον πίνακα Parked. Το αναμενόμενο αποτέλεσμα από τον κώδικα Java είναι η επιτυχής εγγραφή του χρήστη στη βάση δεδομένων.

Στην εικόνα 1 βλέπουμε την μέθοδο parkedIt(), η οποία χρησιμοποιεί τον Constructor Parked().

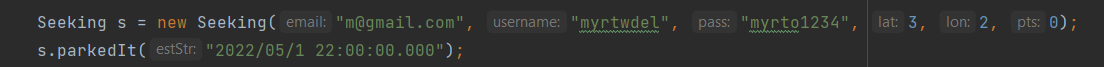
Στην εικόνα 2, φαίνεται ο τρόπος με τον οποίο τρέχουμε τη μέθοδο αυτή. Χρειαζόμαστε ένα αντικείμενο τύπου Seeking, οπότε δημιουργούμε το s και έπειτα τρέχουμε τη μέθοδο. Το αποτέλεσμα, δηλαδή η δεύτερη εγγραφή στον πίνακα parked\_driver της βάσης δεδομένων φαίνεται στην εικόνα 3 και είναι το αναμενόμενο.







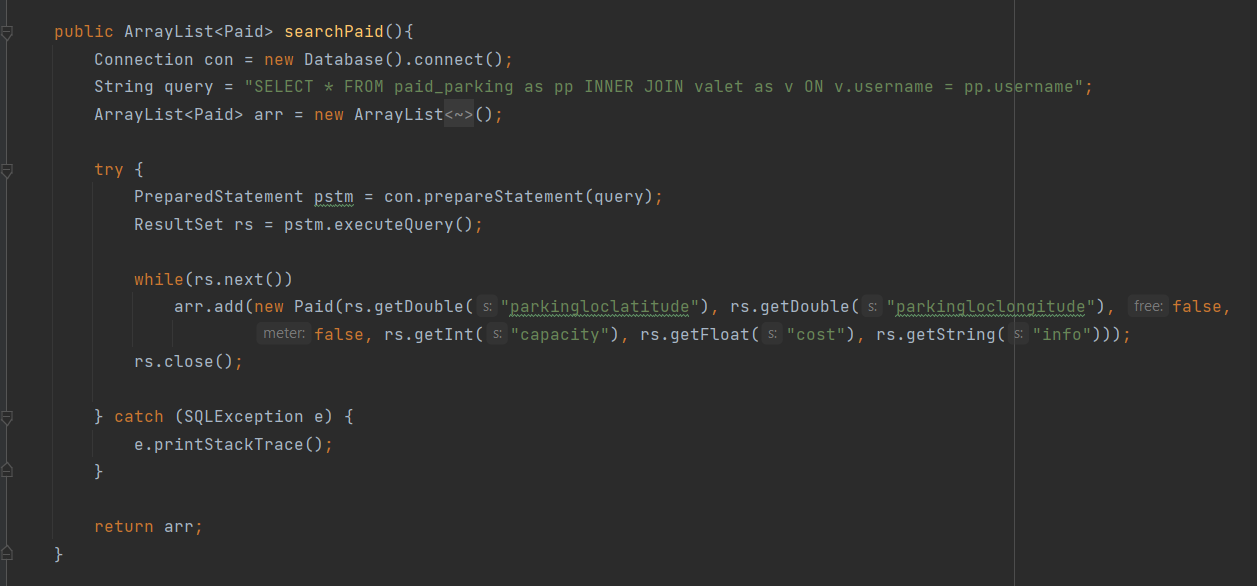
Στην περίπτωση που το timestamp δεν έχει το σωστό format:

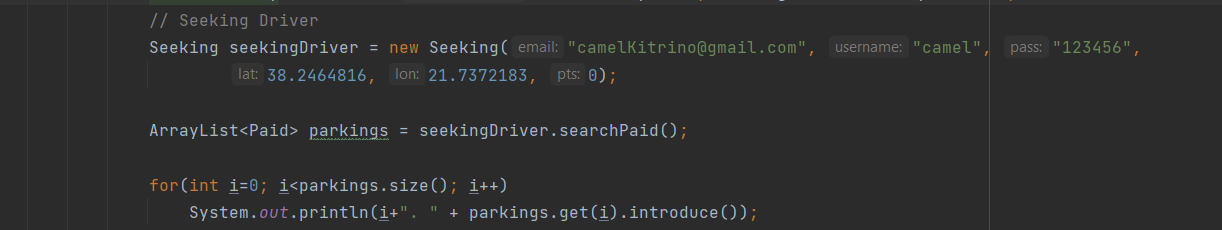


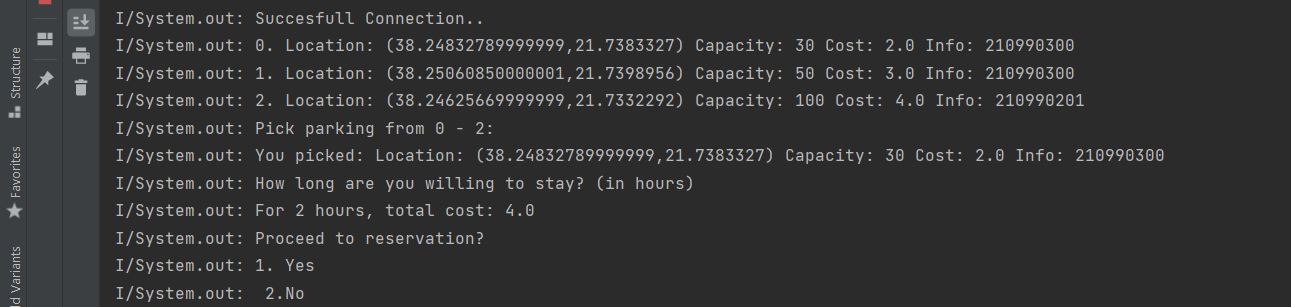


# Μέθοδος searchPaid

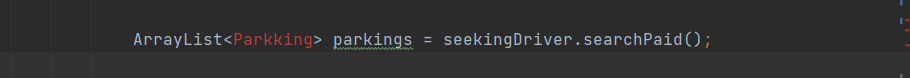
Η searchPaid είναι μέθοδος του Seeking Driver και αποτελεί μια υπομονάδα για τη διαδικασία εύρεσης Ιδιωτικών Parking (use cases 4). Η λειτουργία αυτή και κατ’ επέκταση αυτή η μέθοδος εκτελείται μετά τη πυροδότηση του κουμπιού “Run” της Εικόνας 3 των mock-ups. Πρακτικά, αναλαμβάνει να “φέρει” από το σύστημα τις ιδιωτικές θέσεις parking και τις επιστρέφει σα μια δυναμική λίστα αντικειμένων Paid. Στό παράδειγμα εκτέλεσης, ο seeking driver “βλέπει” τις θέσεις με τη βοήθεια της μεθόδου introduce (Dummy code).

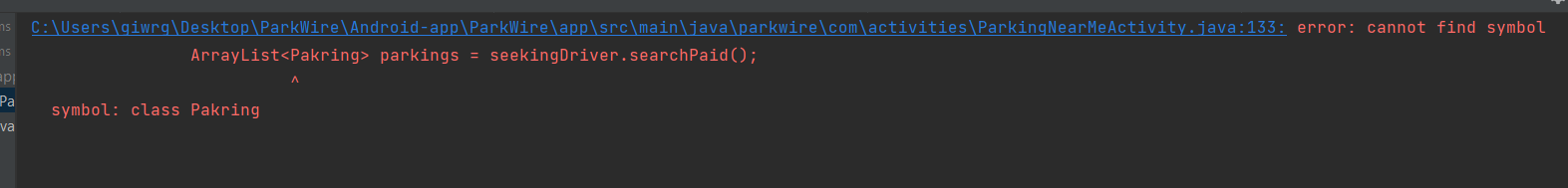






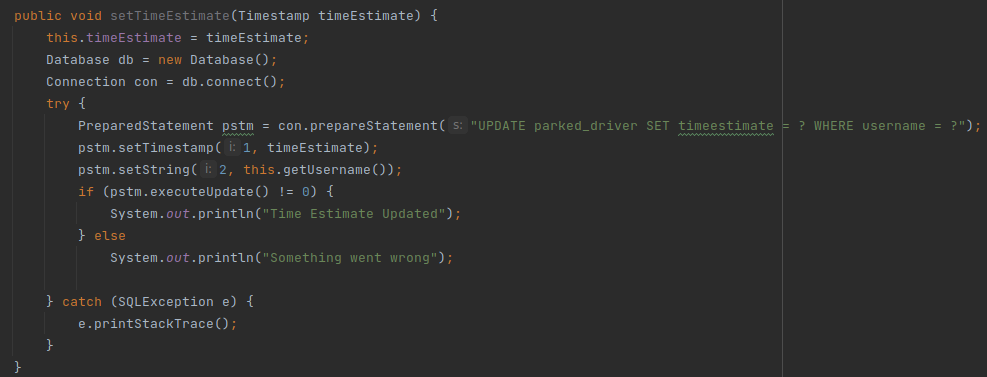
Λάθος επιστραφώμενος τύπος:

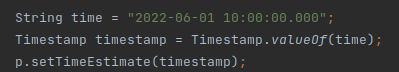


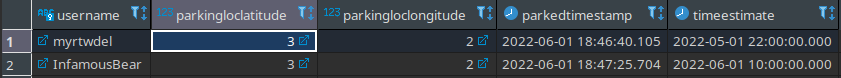


# Μέθοδος setTimeEstimate

Η setTimeEstimate είναι μέθοδος της κλάσης Parked κατά την οποία το σύστημα ζητάει από τον Parked Driver, χρησιμοποιείται και για να δώσει νέο estimate.

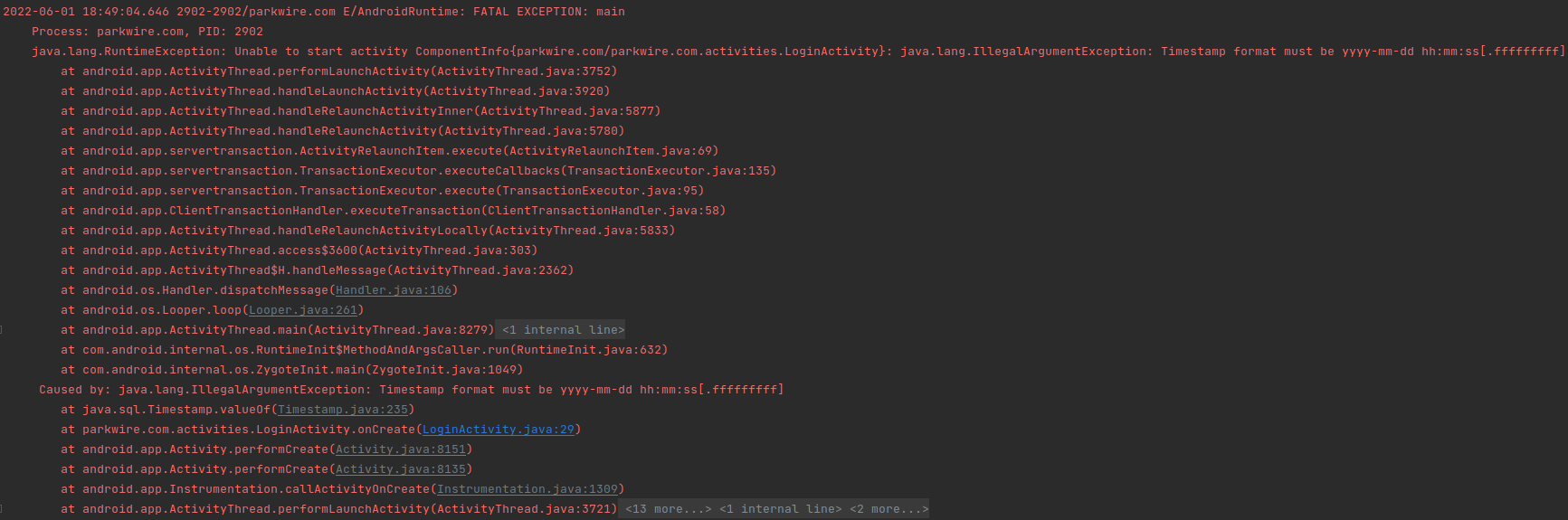






Περίπτωση που το timestamp δεν είναι σε σωστό formatting:

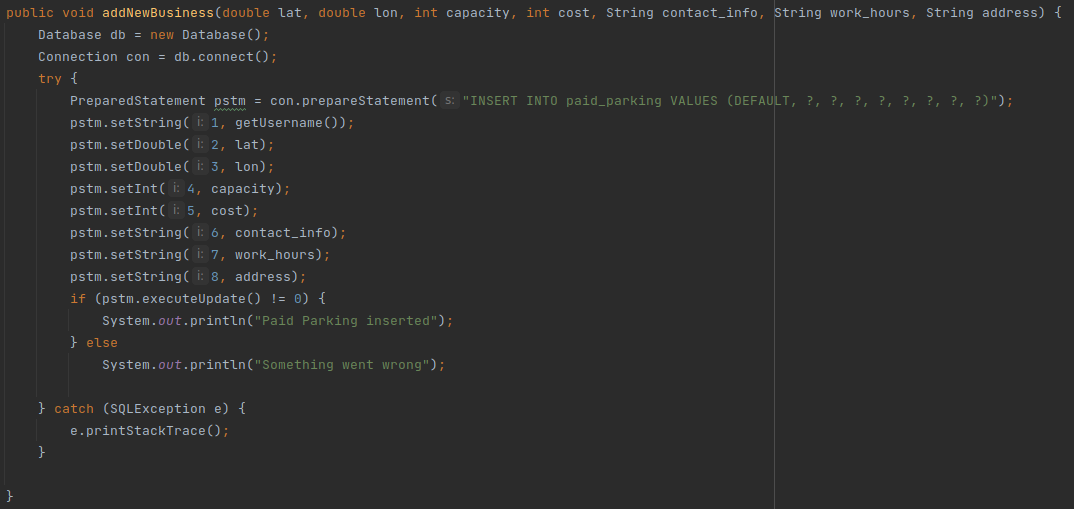




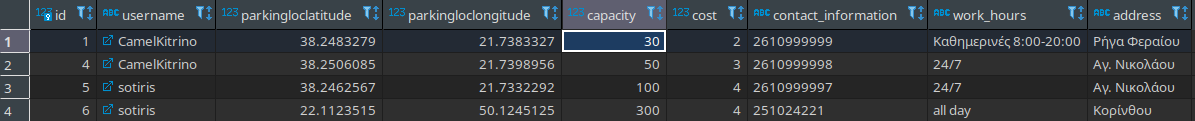
Το σύστημα επιστρέφει πρόβλημα με το format.

# Μέθοδος addNewBusiness

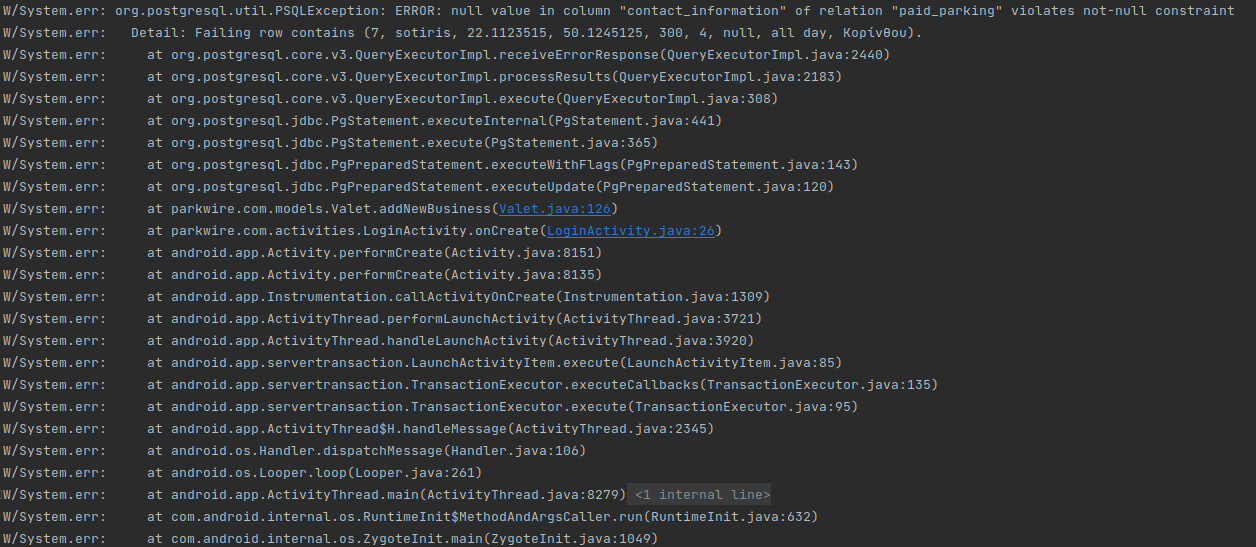
Η addNewBusiness είναι μέθοδος της κλάσης Valet κατά την οποία εισάγεται από τον χρήστη που έχει την ιδιότητα του Valet μια νέα επιχείρηση.







Περίπτωση που κάποια τιμή είναι null:



# Μέθοδος editBusiness

Η editBusiness είναι μέθοδος της κλάσης Valet κατά την οποία ενημερώνεται από τον χρήστη που έχει την ιδιότητα του Valet μια επιχείρηση του.

| public void editBusiness(double lat, double lon, int capacity, int cost, String contact\_info, String work\_hours, String address, int id) {  Database db = new Database();  Connection con = db.connect();  if (lat != 0 && lon != 0) {  try {  PreparedStatement pstm = con.prepareStatement("UPDATE paid\_parking SET parkingloclatitude = ?, parkingloclongitude = ? WHERE username = ? AND id = ?");  pstm.setDouble(1, lat);  pstm.setDouble(2, lon);  pstm.setString(3, this.getUsername());  pstm.setInt(4, id);  if (pstm.executeUpdate() != 0) {  System.*out*.println("Paid Parking Location updated");  } else  System.*out*.println("Something went wrong");  } catch (SQLException e) {  e.printStackTrace();  }  }  if (capacity != 0) {  try {  PreparedStatement pstm = con.prepareStatement("UPDATE paid\_parking SET capacity = ? WHERE username = ? AND id = ?");  pstm.setInt(1, capacity);  pstm.setString(2, this.getUsername());  pstm.setInt(3, id);  if (pstm.executeUpdate() != 0) {  System.*out*.println("Paid Parking capacity updated");  } else  System.*out*.println("Something went wrong");  } catch (SQLException e) {  e.printStackTrace();  }  }  if (cost != 0) {  try {  PreparedStatement pstm = con.prepareStatement("UPDATE paid\_parking SET cost = ? WHERE username = ? AND id = ?");  pstm.setInt(1, cost);  pstm.setString(2, this.getUsername());  pstm.setInt(3, id);  if (pstm.executeUpdate() != 0) {  System.*out*.println("Paid Parking cost updated");  } else  System.*out*.println("Something went wrong");  } catch (SQLException e) {  e.printStackTrace();  }  }  if (contact\_info != null) {  try {  PreparedStatement pstm = con.prepareStatement("UPDATE paid\_parking SET contact\_info = ? WHERE username = ? AND id = ?");  pstm.setString(1, contact\_info);  pstm.setString(2, this.getUsername());  pstm.setInt(3, id);  if (pstm.executeUpdate() != 0) {  System.*out*.println("Paid Parking contact\_info updated");  } else  System.*out*.println("Something went wrong");  } catch (SQLException e) {  e.printStackTrace();  }  }  if (work\_hours != null) {  try {  PreparedStatement pstm = con.prepareStatement("UPDATE paid\_parking SET work\_hours = ? WHERE username = ? AND id = ?");  pstm.setString(1, work\_hours);  pstm.setString(2, this.getUsername());  pstm.setInt(3, id);  if (pstm.executeUpdate() != 0) {  System.*out*.println("Paid Parking work\_hours updated");  } else  System.*out*.println("Something went wrong");  } catch (SQLException e) {  e.printStackTrace();  }  }  if (address != null) {  try {  PreparedStatement pstm = con.prepareStatement("UPDATE paid\_parking SET address = ? WHERE username = ? AND id = ?");  pstm.setString(1, address);  pstm.setString(2, this.getUsername());  pstm.setInt(3, id);  if (pstm.executeUpdate() != 0) {  System.*out*.println("Paid Parking address updated");  } else  System.*out*.println("Something went wrong");  } catch (SQLException e) {  e.printStackTrace();  }  }  } |
| --- |



