ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΑΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Μ117 - Προηγμένες Μέθοδοι Προγραμματισμού

Μπεγέτης Νικόλαος - Μ1345 Καθηγητής: Ιωάννης Σμαραγδάκης

Άσκηση 1: Design Patterns

Στον κώδικα που παραδίδεται στο zip file με όνομα ask1_M1345.zip επιλέξαμε να υλοποιήσουμε τα πρότυπα σχεδίασης:

- ✓ Observer Patern
- ✓ Bridge Pattern

Τα παραπάνω πρότυπα υλοποιήθηκαν από διαφορετικό σύνολο κλάσεων.
Το Observer πρότυπο σχεδίασης το χρησιμοποιήσαμε για να δημιουργήσουμε μία προσομοίωση που αφορά τον έλεγχο για αλλαγές στον πόσιμο νερό από σένσορα και την αυτόματη ενημέρωση 2 applications (θεωρητικά ενός web app και ενός mobile app), τα οποία σχεδιάστηκαν με το observer pattern πρότυπο.Διαλέξαμε το observer pattern πρότυπο γιατί με αυτό μπορούμε να παρακολουθούμε τις μεταβολές των τιμών και να ενημερώνουμε ταυτόχρονα διάφορες εφαρμογές που χρειάζονται αυτό το συγχρονισμό.

Τα πλεονεκτήματα του παραπάνω σχεδιασμού είναι ότι μπορούμε να προσθέσουμε πολλούς observers ανα πάσα στιγμή χωρίς να επηρεάζουμε τους σένσορες και να προκαλούμε αλλαγές. Στην περίπτωσή μας οι observers είναι το web App και το Mobile App, που χρειάζονται τον αυτόματο συγχρονισμό χωρίς όμως να εξαρτώνται από το αντικείμενο το οποίο αλλάζει τιμές και χρειάζεται να συγχρονίζεται (observable αντικείμενο). Το observable αντικείμενο είναι ο sensor που ελέγχει για μεταβολές στα συστατικά του νερού.
Το παραπάνω όμως πρότυπο μπορεί να προκαλέσει συμφόρηση στην περίπτωση που οι αλλαγές είναι πολλές, γρήγορες και μεγάλες σε όγκο δεδομένων. Σε αυτή την περίπτωση και αν οι observers είναι πολλοί η διάδοση των δεδομένων μπορεί να καθυστερήσει και να προκληθούν προβλήματα.

Το Bridge πρότυπο σχεδίασης το χρησιμοποιήσαμε για να δημιουργήσουμε μία προσομοίωση υλοποίησης φορητών υπολογιστικών συσκευών (tablet, mobile) τα οποία μπορεί να έχουν διάφορα λειτουργικά συστήματα και όμοιες λειτουργίες σε αυτά τα συστήματα τις οποίες πρέπει όλες οι συσκευές να υλοποιούν. Τέτοιες λειτουργίες είναι: να ανάβουν(turn on power) και να σβήνουν(turn off power), να μπορούν να πραγματοποιούν κλήσεις, να στέλνουν μηνύματα, να κατεβάζουν εφαρμογές, να ανοίγουν και να κλείνουν εφαρμογές και να τραβάνε φωτογραφίες.
Συνεπώς, τα ζητούμενα είναι 1) να μπορούν να υλοποιούνται όλες οι παραπάνω λειτουργίες από όλους τους συνδυασμούς συσκευών-λειτουργικών συστημάτων, 2) να είναι δυνατή η εύκολη επεκτασιμότητα νέων λειτουργιών ταυτόχρονα αλλάζοντας ένα μόνο κομμάτι κώδικα για όλους τους συνδυασμούς και όχι ενα κομμάτι κώδικα σε κάθε

συνδυασμό, 3) να είναι δυνατό να εγκατασταθούν διαφορετικά λειτουργικά συστήματα σε κάθε συσκευή εύκολα και απλά, να συνδυάζονται δηλαδή οι δύο ιεραρχίες. Για να συμβεί αυτό το πιο συμβατό πρότυπο σχεδίασης είναι το Bridge πρότυπο, γι αυτό και το επιλέξαμε, για να υπάρχει μία «γέφυρα» ανάμεσα στις υλοποιήσεις που αφορούν οι μεν τις φορητές συσκευές και οι δε τα λειτουργικά συστήματα.
Οι φορητές συσκευές που επιλέξαμε είναι οι ΝΟΚΙΑ, LG και i-phone, ενώ τα λειτουργικά συστήματα που επιλέξαμε είναι τα Windows, Android και iOS. Στην υλοποίησή μας κατασκευάζουμε ενδεικτικά 2 συσκευές ΝΟΚΙΑ με λειτουργικά Windows και Android, 2 συσκευές LG με λειτουργικά Android και Windows, και 2 συσκευές i-phone με λειτουργικά iOS και Windows.
Τα πλεονεκτήματα του παραπάνω πρότυπο σχεδίασης είναι 1) ότι με αυτό το πρότυπο επιτυγχάνεται η αποσύνδεση της *αφαίρεσης* στην υλοποιήση. 2)Ότι αν χρειαστεί μπορεί να μεταβληθεί η ιεραρχία της υλοποίησης σε οποιαδήποτε πλευρά της «γέφυρας», είτε στις συσκευές είτε στα λειτουργικά συστήματα, χωρίς να αλληλοεπηρεάζονται. 3) Ότι όπως εξηγούμε και στο 2), το ίδιο ισχύει και για την ιεραρχία της αφαίρεσης. 4) Ότι με αυτό το πρότυπο έχουμε μινιμαλισμό στον κώδικα , μιας και χρειάζεται να φτιάξουμε λιγότερες υποκλάσεις (6 σε αυτή την προσομοίωση, 3 για τις συσκευές και 3 για τα λειτουργικά), αντί για περισσότερες(9, δηλαδή κάθε λειτουργικό για κάθε μία από τις συσκευές, το οποίο βέβαια θα έδινε αυτονομία κάθε λειτουργικό σε κάθε συσκευή ξεχωριστά). 5) Ως συνέπεια του 4) ο κώδικας όντας μινιμαλιστικός είναι και πιο ξεκάθαρος στην κατανόηση.
Το παραπάνω πρότυπο όμως, μπορεί να μας οδηγήσει πιο εύκολα σε αδιέξοδα κατά το debugging καθώς η πολυπλοκότητα λόγω «γέφυρας» αυξάνεται αφού πλέον δεν υπάρχει αυτονομία για κάθε συσκευή και άρα είναι πιο δύσκολη η ανίχνευση για bugs.

Αξίζει να σημειωθεί ότι και στις 2 περιπτώσεις των 2 προτύπων σχεδίασης το *overriding* που προσφέρει η Java όταν κάνει extend καθώς και τα interfaces βοηθάνε πολύ στην υλοποίησή τους. Επίσης, το Bridge pattern θα μπορούσαμε να το υλοποίσουμε με σχετικά λίγες αλλαγές και σε C++ κάνοντας χρήση Templates, ενώ το Observer Pattern, θα ήταν ευκολότερο να το υλοποιήσουμε και σε C# χρησιμοποιώντας delegates.

Στο παραδοτέο περιλαμβάνεται το Eclipse project, καθώς όλη η υλοποίηση έγινε στο Eclipse Luna.

Τετάρτη, 4 Νοεμβρίου 2015