<Destination Sol>

3η Εργασία στην Προηγμένη Τεχνολογία Λογισμικού

Εντοπισμός Code Smells , Refactorings , Git

<Σαρόγλου Απόστολος>

4/08/22,Θεσσαλονίκη

Με την μελέτη εξέλιξη της ποιότητας του λογισμικού, χρησιμοποιώντας τις μετρικές πολυπλοκότητας, της σύζευξης, της συνοχής, της κληρονομικότητας, και του μεγέθους, όπου έγιναν στο Έγγραφο Ελέγχου Ποιότητας ακολούθησε η μελέτη των προβληματικών κλάσεων. Έγινε αναγνώριση Προβλημάτων Ποιότητας του λογισμικού που μελετήθηκε και προέκυψαν τα αποτελέσματα για συγκεκριμένες κλάσεις όπου ο χαρακτηρισμός των τιμών των μετρικών τους είναι HIGH έχουν δηλαδή υψηλές τιμές αναφοράς. Οι κλάσεις αυτές μελετήθηκαν περεταίρω καθώς επίσης έγινε προσπάθεια με στόχο την βελτίωση των τιμών των μετρικών τους σε επακόλουθα στάδια και αναλύθηκαν ώστε να ληφθούν οι νέες τιμές για κάθε αναδόμηση που έγινε.

Αρχικά έγινε εντοπισμός των οσμών κώδικα (code smells) για την τελευταία έκδοση λογισμικού ανοικτού κώδικα σε Java που επιλέχθηκε στη 2η εργασία, με τη χρήση του εργαλείου **SmellDetectorMerger**. Με τα αποτελέσματα που φαίνονται στον παρακάτω πίνακα, ακολούθησαν και οι επόμενες αλλαγές στον κώδικα.

Smell Type	Affected Element	Detected by
Long Parameter List	SolGun.shoot()	Organic
Feature Envy	SolGun.update()	Organic
Long Method	SolGun.update()	Organic
Dispersed Coupling	SolGun.update()	Organic
Long Parameter List	SolGun.update()	Organic
Complex Class	SolInputManager	Organic
God Class	SolInputManager	PMD
Class Data Should Be Private	SolInputManager	Organic
Feature Envy	SolInputManager.draw()	Organic
Dispersed Coupling	SolInputManager.draw()	Organic
Long Method	SolInputManager.update()	Organic
Intensive Coupling	SolInputManager.update()	Organic
Feature Envy	SolInputManager.updateCursor()	Organic
Long Parameter List	SolInputProcessor.touchDown()	Organic
Long Parameter List	SolInputProcessor.touchUp()	Organic
Lazy Class	SolItem	Organic
Class Data Should Be Private	SolItemType	Organic
Lazy Class	SolItemType	Organic
Class Data Should Be Private	SolItemTypes	Organic
Class Data Should Be Private	SolLayouts	Organic
Lazy Class	SolLayouts	Organic
God Class	SolMath	PMD
Long Parameter List	SolMath.toRel()	Organic
Long Parameter List	SolMath.toWorld()	Organic
Class Data Should Be Private	SolNames	Organic

Smell Type	Affected Element	Detected by
Data Class	ShowInventory	Organic
Refused Parent Bequest	ShowInventory	Organic
Class Data Should Be Private	ShowInventory	Organic
Feature Envy	ShowInventory.updateCustom()	Organic
Long Method	ShowInventory.updateCustom()	Organic
Intensive Coupling	ShowInventory.updateCustom()	Organic
Long Parameter List	Sky.handleContact()	Organic
Long Parameter List	Sky.receiveDmg()	Organic
Feature Envy	Sky.update()	Organic
Long Method	Sky.update()	Organic
Class Data Should Be Private	SkyConfig	Organic
Data Class	SloMo	Organic
Class Data Should Be Private	SloMo	Organic
Class Data Should Be Private	SloMoConfig	Organic
Long Parameter List	SloMoConfig.SloMoConfig()	Organic
Feature Envy	SmallObjAvoider.avoid()	Organic
Long Method	SmallObjAvoider.avoid()	Organic
Long Parameter List	SmallObjAvoider.avoid()	Organic
God Class	SolApplication	PMD
Feature Envy	SolApplication.create()	Organic
Long Method	SolApplication.create()	Organic
Feature Envy	SolApplication.draw()	Organic
Feature Envy	SolCam.applyInput()	Organic
Feature Envy	SolCam.drawDebug()	Organic
Feature Envy	SolCam.getDesiredViewDistance()	Organic

Το πρώτο στιγμιότυπο είναι η κλάση 16 (SolInputManager) και το δεύτερο η κλάση 29 (SolApplication).

Η Τρίτη κλάση shipbuilder.java (84) επίσης επιλέγεται απλά δεν φαίνεται στο στιγμιότυπο.

Αλλαγή σε κλάση 16 (

Στην κλάση εντοπίστηκε χρήση πολλαπλών if statements τα οποία δυσκολεύουν την ανάγνωση του κώδικα από τον προγραμματιστή.

```
public void update(SolApplication solApplication) {
    boolean mobile = solApplication.isMobile();
    SolGame game = solApplication.getGame();

    // This keeps the mouse within the window, but only when playing the game with the mouse.
    // All other times the mouse can freely leave and return.
    if (!mobile && solApplication.getOptions().controlType == GameOptions.ControlType.MIXED && game != null && getTopScre
        if (!Gdx.input.isCursorCatched() && !osIsLinux) {
            Gdx.input.setCursorCatched(true);
        }
        maybeFixMousePos();
    } else {
        if (Gdx.input.isCursorCatched()) {
            Gdx.input.setCursorCatched(false);
        }
    }

    updatePointers();

    boolean consumed = false;
    mouseOnUi = false;
    boolean clickOutsideReacted = false;
    for (SolUiScreen screen : screens) {
        boolean consumedNow = false;
        List<SolUiControl> controls = screen.getControls();
        for (SolUiControl control : controls) {
            control.update(inputPointers, currCursor != null, !consumed, this, solApplication);
            if (control.isOn() || control.isJustOff()) {
```

Αυτό επιτυγχάνεται με την δημιουργία μιας νέας private μεθόδου η οποία καλείται εντός της υπάρχουσας προκαλώντας το επιθυμητό αποτέλεσμα.

```
        scrolledUp = null;
        touchDragged = false;
    }
    private checkInputMeth() {

        boolean mobile = solApplication.isMobile();
        SolGame game = solApplication.getGame();

        if (!mobile && solApplication.getOptions().controlType == GameOptions.ControlType.MIX
            if (!Gdx.input.isCursorCatched() && !osIsLinux) {
                Gdx.input.setCursorCatched(true);
            }
            maybeFixMousePos();
        } else {
            if (Gdx.input.isCursorCatched()) {
                Gdx.input.setCursorCatched(false);
            }
        }
        return game;
    }
```

Αλλαγή σε κλάση 29 (SolApplication)

Ομοίως στην κλάση SolApplication στην μέθοδο draw() όπου υπάρχουν πολλές if δυσκολεύουν την κατανόηση του κώδικα χωρίς να χρειάζεται.

```
            solGame.update();
        }

        SolMath.checkVectorsTaken(null);
    }

    private void draw() {
        Gdx.gl.glClear(GL20.GL_COLOR_BUFFER_BIT);
        commonDrawer.begin();
        if (solGame != null) {
            solGame.draw();
        }
        uiDrawer.updateMtx();
        inputManager.draw(uiDrawer, this);
        if (solGame != null) {
            solGame.drawDebugUi(uiDrawer);
            factionDisplay.drawFactionNames(solGame, uiDrawer, inputManager, solGame.getObjectManager());
        }
        if (fatalErrorMsg != null) {
            uiDrawer.draw(uiDrawer.whiteTexture, displayDimensions.getRatio(), .5f, 0, 0, 0, .25f, 0, SolColo
            uiDrawer.drawString(fatalErrorMsg, displayDimensions.getRatio(), .5f, FontSize.MENU, true, SolCol
            uiDrawer.drawString(fatalErrorTrace, .2f * displayDimensions.getRatio(), .6f, FontSize.DEBUG, fal
        }
        DebugCollector.draw(uiDrawer);
        if (solGame == null) {
            uiDrawer.drawString("v" + Const.VERSION, 0.01f, .974f, FontSize.DEBUG, UiDrawer.TextAlignment.LEF
        }
        commonDrawer.end();
    }
```

Έτσι, αντιμετωπίζουμε το ίδιο πρόβλημα δημιουργώντας μεθόδους, οι οποίες καλούνται εντός της draw που σχεδιάζουν ξεχωριστά τα αντικείμενα του παιχνιδιού. Ταυτόχρονα επιτυγχάνεται πιο εύκολη σχεδίαση μελλοντικών αντικειμένων με τον ίδιο τρόπο.

Η μέθοδος draw() διασπάστηκε στις παρακάτω μεθόδους.

[soldraw(), soldrawDebugUi(), soldrawUi(), soldrawString()]

```
private void draw() {
    Gdx.gl.glClear(GL20.GL_COLOR_BUFFER_BIT);
    commonDrawer.begin();
    soldraw();
    uiDrawer.updateMtx();
    inputManager.draw(uiDrawer, this);
    soldrawDebugUi();
    soldrawUi();
    DebugCollector.draw(uiDrawer);
    soldrawString();
    commonDrawer.end();
}
public void soldraw() {
    if (solGame != null) {
        solGame.draw();
    }
}
public void soldrawDebugUi() {
    if (solGame != null) {
        solGame.drawDebugUi(uiDrawer);
        factionDisplay.drawFactionNames(solGame, uiDrawer, inputMana
    }
}
public void soldrawUi() {
    if (fatalErrorMsg != null) {
        uiDrawer.draw(uiDrawer.whiteTexture, displayDimensions.getR
        uiDrawer.drawString(fatalErrorMsg, displayDimensions.getRat
        uiDrawer.drawString(fatalErrorTrace, .2f * displayDimension
    }
}
public void soldrawString() {
```

Αλλαγή σε κλάση 84 (ShipBuiler)

```
if (pilot.isPlayer()) {
    for (List<SolItem> group : itemContainer) {
        for (SolItem i : group) {
            if (i instanceof Shield) {
                if (i.isEquipped() > 0) {
                    shield = (Shield) i;
                    continue;
                }
            }
            if (i instanceof Armor) {
                if (i.isEquipped() > 0) {
                    armor = (Armor) i;
                    continue;
                }
            }
            if (i instanceof Gun) {
                Gun g = (Gun) i;
                if (i.isEquipped() > 0) {
                    int slot = i.isEquipped();
                    if (g1 == null && hullConfig.getGunSlot(0).allowsRotation() != g.config.fixed
                        g1 = g;
                        continue;
                    }
                    if (hullConfig.getNrOfGunSlots() > 1 && g2 == null && hullConfig.getGunSlot(1)
                        g2 = g;
                    }
                    if (g1 != g && g2 != g) {
                        i.setEquipped(0); // The gun couldn't fit in either slot
                    }
                }
            }
```

Στην κλάση shipBuilder η μέθοδος περιλάμβανε ξεχωριστές περιπτώσεις διαχωρισμού τις επιλογής τον αντικειμένων του αντικειμένου player. Στις δυο περιπτώσεις γινόταν ξεχωριστά επαναληπτικές μέθοδοι οι οποιες συνέβαιναν δυο φορες χωρίς λόγο προκαλόντας πολλές επαναλήψεις για κάθε αντικείμενο χωρίς να υπάρχει κάποιος λόγος. Με την αλλαγή στον κώδικα οι επαναλήψεις λιγοστεφσαν καθώς ο διαχωρισμός γίνεται εντός της for. Βελτιώνει την ταχύτητα του κώδικα, το complexity καθώς ο κώδικας γίνεται πιο κατανοητός και το cohesion.

```
Armor armor = null;

for (List<SolItem> group : itemContainer) {
    for (SolItem i : group) {
        // For the player use new logic that better respects what was explicitly equipped
        if (pilot.isPlayer()) {
            if (i instanceof Shield) {
                if (i.isEquipped() > 0) {
                    shield = (Shield) i;
                    continue;
                }
            }
            if (i instanceof Armor) {
                if (i.isEquipped() > 0) {
                    armor = (Armor) i;
                    continue;
                }
            }
            if (i instanceof Gun) {
                Gun g = (Gun) i;
                if (i.isEquipped() > 0) {
                    int slot = i.isEquipped();
                    if (g1 == null && hullConfig.getGunSlot(0).allowsRotation() != g.conf
                        g1 = g;
                        continue;
                    }
                    if (hullConfig.getNrOfGunSlots() > 1 && g2 == null && hullConfig.getG
                        g2 = g;
                    }
                    if (g1 != g && g2 != g) {
                        i.setEquipped(0); // The gun couldn't fit in either slot
```

```
                }
            }
        } else {
            if (i instanceof Shield) {
                shield = (Shield) i;
                continue;
            }
            if (i instanceof Armor) {
                armor = (Armor) i;
                continue;
            }
            if (i instanceof Gun) {
                Gun g = (Gun) i;
                if (g1 == null && hullConfig.getGunSlot(0).allowsRotation() != g.config.fixed) {
                    g1 = g;
                    continue;
                }
                if (hullConfig.getNrOfGunSlots() > 1 && g2 == null && hullConfig.getGunSlot(1).all
                    g2 = g;
                }
            }
        }
    }
}
```

Τελικά προκύπτει ο πίνακας αλλαγών στις μετρικές όπως φαίνεται παρακάτω. Η έκδοση 2.0.0 v1 είναι τα αποτελέσματα μετά την αλλαγή της κλάσης SollinputManager, ομοίως η 2.0.0 v2 μετά την αλλαγή της SolApplication και τέλος η v3 της shipbuilder. Σημαντικές φαίνονται να είναι οι αλλαγές για τη LCOM, υπάρχει κατακόρυφη και σταθερή μείωση από την v1 έκδοση και μετά.

	2.0.0 original				2.0.0 v1				2.0.0 v2				2.0.0 v3		
	SUM	**AVG**	**MAX**		SUM	AVERAGE	MAX		SUM	AVERAGE	MAX		SUM	AVERAGE	MAX
DIT	405	1,035806	2	**DIT**	433	1,043373	2	**DIT**	433	1,043373	2	**DIT**	433	1,043373	2
CBO	3173	8,11509	220	**CBO**	2025	4,879518	196	**CBO**	2026	4,881928	196	**CBO**	2026	4,881928	196
LCOM	68159	174,3197	48092	**LCOM**	11846	30,60982	6863	**LCOM**	11846	30,60982	6863	**LCOM**	11846	30,60982	6863
WMC*	560,5028	1,658292	10	**WMC***	323,243	0,814214	6	**WMC***	323,243	0,814214	6	**WMC***	323,243	0,814214	6
SIZE1	24149	61,76215	544	**SIZE1**	9822	23,66747	544	**SIZE1**	9822	23,66747	544	**SIZE1**	9816	23,65301	544

Το git που βρίσκονται οι παραπάνω αλλαγές είναι στο παρακάτω link:

https://github.com/apostolisSrg/Software-Analysis-DestinationSol