תשובות לשאלות במטלה 2

**קבוצה מס' 3**

**תאריך הגשה: 22.12.14**

**מגישים:**

|  |  |  |
| --- | --- | --- |
| **שם הסטודנט** | **כתובת אימייל** | **מס' ת.ז.** |
| אורי זיו | oriziv@elcam.co.il | 027360486 |
| עוז מוכתר | oz1804@gmail.com | 036567659 |
| אולה חרמונן | harmonenola@gmail.com | 326927431 |
| ילנה לדיז'נסקי | al8389@gmail.com | 309031284 |

1. **א.** במידול המערכת שלנו השתמשנו בעקרון התכן Divide and conquer- הפרד ומשול.לפי עקרון זה ממדלים את המערכת כך שתהיה מחולקת למספר חלקים, דבר שמקל על המימוש, ועל ביצוע שינויים עתידיים. העקרון בא לידי ביטוי בחלוקת דיאגרמת ה-class לפי מודל שלושת השכבות. חילקנו את הקלאסים השונים לאחד משלושת ה-stereotypes הבאים:

* יישות (ברמת ה-(DB.
* ממשק (מקשר בין ה-actor למערכת, כלומר בין הקליינט לסרבר).
* בקרה(מקשר בין ממשק ליישות ואחראי על ביצוע תהליכים).

לדוגמה: מידלנו מוצר במערכת CCRM בצורה הבאה:

* יישות product: מכילה את פרטי המוצר. שדות כגון שם, מס"ד וכו'.
* בקרה productManagement: מכיל את כל המתודות השונות הקשורות בניהול המוצר.
* ממשק ProductManagementScreen: מכיל את כפתורי ה-GUI שבעזרתן המשתמש מבצע פעולות במערכת.

בנוסף לכך, כאשר בנינו את דיאגרמת ה-package , חילקנו אותה כך שתהיה הפרדה ברורה בין ה- client ל-.server

**ב.** עקרון של oop שלא בא לידי ביטוי במידול הפרויקט שלנו הוא overriding-דריסה. ב-UML לא ניתן להביא לידי ביטוי עיקרון זה היות ו-UML נוגע לכל היותר בחתימה של המתודות (שלא משתנה כאשר אנו דורסים אותן).

1. במידול המערכת שלנו באים לידי ביטוי Design patterns הבאים:

* : General Hierarchy התבנית הזו באה לידי ביטוי בשני מקומות במערכת: במסך general screenשממנו יורשים שאר המסכים, וביישות Person שממנה יורשים עובד ולקוח.
* OCSF: התבנית הזו באה לידי בתקשורת בין client ו- server ע"י שליחת הודעות כמו הודעות מידע והודעות שגיאה והתראה.

1. **א.1.** היתרונות של מודל ה-UML כעזר לתהליך התכנון הם:

* הוא מאפשר למתכנת לתת ביטוי ויזואלי למחלקות השונות שמרכיבות את המערכת, ובכך לחלק את המערכת למודולים השונים שלה בצורה גרפית.
* מגדיר שפה אוניברסלית עם נורמות ברורות.
* ניתן ליצור באמצעותו דיאגרמות קלות להבנה, אותן ניתן לשתף עם חברי צוות שעובדים איתו, מנהלים ואפילו חברות שקונות את התוכנה.
* נותן את האפשרות לצפות קשיים בשלבים המאוחרים יותר של תהליך הפיתוח וכך לקבל החלטות מראש. מה שעשוי להוזיל את עלויות הפיתוח וזמן הפיתוח.
* מודל זה מאפשר גם לשתף רעיונות מופשטים כמו תבניות תכן שמסייעות אף הן למדל מערכת יותר יעילה ונכונה, וגם לבצע שימוש חוזר ברעיונות קיימים שהוכחו כיעילים.
* באמצעות שימוש בדיאגרמות sequence ,activityניתן להראות את רצף האירועים על ציר הזמן, כלומר איזה פעולות מתרחשותבאיזה שלב של התהליך.
* במודל זה קיימת אופציה לשימוש בהורשה, מה שמונע שכפול מידע.

בזמן מידול המערכת שמנו לב כי בכל המסכים שלנו צריכה להיות אופציה של שליחת הודעת מערכת, לכן החלטנו ליצור מסך ראשי שבו תהיה קיימת האופציה הזאת, וממנו יורשים שאר מסכי המערכת. ע"י כך מנענו את שכפול המתודה displayNotification בכל המסכים, מה שגרם לדיאגרמה להיות ברורה יותר ונוחה לקריאה.

**א.2.** הקושי במידול ב-UML שהוא אינו מאפשר למדל חלון (ממשק משתמש) בצורה ויזואלית עם תיבות טקסט וכפתורים. קושי זה יכול לגרום לנו לאי אחידות בסידור כפתורים, חלונות ומסכים בזמן המימוש. פתרון אפשרי לכך הוא להשתמש בתוכנת PowerPoint על מנת ליצור תמונה ויזואלית של מסכים וכפתורים, וכך כל כותבי הקוד יוכלו לראות את הפורמט, להציע הצעות לייעול ואחר כך לממש את הממשק הגרפי בהתאם.

**ב.** מידול הפרויקט שלנו בעזרת מודלUML נתן מענה חלקי לצורכי התכן של הפרויקט.

דיאגרמותUML נותנות מענה לחלק התיאורי של מערכת, כלומר ניתן לראות איזה ישויות ישנן במערכת ואיך מתבצעת התקשורת ביניהן, וניתן לראות את סדר הפעולות בביצוע תהליכים שונים במערכת. אך לעומת זאת, מידול בעזרת מודל UML לא נותן לנו כלים כדי שנוכל לדעת איך המערכת תיראה בפועל על מסך המשתמש. לדוגמה המסך CustomerManageScreen קיים במערכת שלנו ובו מתוארות כל המתודות ש-class מבצע ואילו כפתורים יהיו על המסך. אבל מידע זה הוא חלקי בלבד, ואינו נותן לנו לראות את הסדר של הכפתורים על המסך, את צבעם ואת צבע המסך עצמו. כלומר לא ניתן לדעת מהו העיצוב הגרפי שיהיה על המסך.