**Group Number 1 / Date: 25.4.16**

|  |  |  |
| --- | --- | --- |
| shayscal90@gmail.com | 302632105 | Shay Shahar |
| yakir1114@gmail.com | 312143969 | Yakir Karandian |
| rizraz2001@gmail.com | 305003659 | Raz Cohen |
| [asaff19899@gmail.com](mailto:asaff19899@gmail.com) | 200940997 | Assaf Tzarfati |

**Assignment 2 Answers:**

1. כאשר ניגשנו לתהליך יצירת התיק הרפואי במערכת עמדה בפנינו דילמה כיצד לממש את הדבר. מבחינת הדרישות, תיק רפואי ראשוני נוצר אוטומטית במערכת כאשר המוקדן מזין תעודת זהות של לקוח שאינו קיים. כאשר ניגשנו לתכנון התהליך, חשבנו אם יש הגיון ביצירה אוטומטית של תיק רפואי כיוון שאם נעבוד ע"פ הגישה של יצירה אוטומטית ייתכנו טעויות הקלדה, ובעקבות כך ייווצרו תיקים רפואיים ללקוחות פיקטיביים. פתרנו את הבעיה ע"י כך שהחלטנו שבמקרה והלקוח שהוזן אינו קיים, תוצג הודעה מתאימה למוקדן שיוכל במידת הצורך ליצור לקוח ע"י מילוי פרטיו הדרושים. בעת יצירת לקוח, המערכת יוצרת אוטומטית תיק רפואי. העיקרון שבו השתמשנו לפתרון הבעיה הוא "הכלה" (Composition). כאשר ניצור אובייקט מסוג Client , במהלך היצירה שלו ייווצר אובייקט מסוג MedicalFile אשר מקושר אליו באופן ישיר, כיוון שבין לקוח לתיק רפואי קיים יחס של 1 ל – 1.

2-

א.

אחד העקרונות של Object Oriented Analysis & Design שבו השתמשנו הוא

ירושה (Generalization): *"ב*[*תכנות מונחה-עצמים*](https://he.wikipedia.org/wiki/%D7%AA%D7%9B%D7%A0%D7%95%D7%AA_%D7%9E%D7%95%D7%A0%D7%97%D7%94-%D7%A2%D7%A6%D7%9E%D7%99%D7%9D)*, ירושה היא דרך לבסס יחס "סוג-של" (is-a) בין* [*עצמים*](https://he.wikipedia.org/wiki/%D7%90%D7%95%D7%91%D7%99%D7%99%D7%A7%D7%98_(%D7%9E%D7%93%D7%A2%D7%99_%D7%94%D7%9E%D7%97%D7%A9%D7%91))*..."* (מתוך ויקיפדיה). בעזרת עקרון זה, ניתן להשתמש בתכונות של מחלקה אחת במחלקה אחרת ולהוסיף לה תכונות או מתודות אחרות אשר יהיו ייחודיות למחלקה היורשת.

עיקרון זה בא לידי ביטוי בכמה מקומות ב-Class Diagram:

מחלקת GeneralUI שמורישה לשאר מחלקות ה-GUI הנמצאות בתרשים המחלקות.

בנוסף מחלקת AbstractReport מורישה למחלקת WeeklyReport למשל ו-Monthly Report יורשת מ-WeeklyReport כיוון שאלו תכונות שצריכות להופיע גם במחלקה היורשת כיוון שכל דו"ח מערכתי מכיל נתונים בסיסיים ונתונים נוספים הייחודיים לו. בנוסף יש את המחלקה Person שכל שחקן במערכת שהוא אדם יורש ממנה. מחלקה זו מייצגת את תכונות האדם הבסיסיות שיש לשמור לכל אדם במערכת.

בנוסף, קיימת הורשה ממחלקת- AbstractClient וממחלקת- AbstractServer

למחלקות ClientController, ServerController בהתאמה. כיון שמחלקות אבסטרקטיות יש לממש לפניי היצירה של המופעים שלהן, עלינו לבצע את ההורשה ולהשתמש בתכונות של המחלקות ולהגדיר את הנתונים הייחודים עבור התקשורת בפרויקט שלנו.

ב.

פולימורפיזם: *"* [*פולימורפיזם*](https://he.wikipedia.org/wiki/he:%D7%A4%D7%95%D7%9C%D7%99%D7%9E%D7%95%D7%A8%D7%A4%D7%99%D7%96%D7%9D_(%D7%AA%D7%95%D7%9B%D7%A0%D7%94)) *(רב צורתיות) היא תכונה שמאפשרת לנו להתייחס לאובייקטים בצורה גמישה, המתאימה לצרכינו, בדרך כלל על ידי שימוש בתכונה ההורשה."* (מתוך ויקיפדיה). זו תכונה של [שפות תכנות](http://he.wikipedia.org/wiki/%D7%A9%D7%A4%D7%AA_%D7%AA%D7%9B%D7%A0%D7%95%D7%AA) מונחות עצמים המאפשרת לטפל ב[ערכים](http://he.wikipedia.org/wiki/%D7%A2%D7%A8%D7%9A_(%D7%9E%D7%93%D7%A2%D7%99_%D7%94%D7%9E%D7%97%D7%A9%D7%91)) מ[טיפוסים](http://he.wikipedia.org/wiki/%D7%98%D7%99%D7%A4%D7%95%D7%A1_%D7%A0%D7%AA%D7%95%D7%A0%D7%99%D7%9D) שונים בעזרת [ממשק](http://he.wikipedia.org/w/index.php?title=%D7%9E%D7%9E%D7%A9%D7%A7_%D7%AA%D7%95%D7%9B%D7%A0%D7%94&action=edit&redlink=1) אחיד שפת תכנות התומכת בפולימורפיזם מאפשרת למתכנת יכולת לממש אלגוריתמים ומבני נתונים שונים בצורה גנרית ולאחר מכן לממש אותם באופן ייחודי בהתאם לאובייקט היורש.

במערכת שהתבקשנו לממש אין אובייקטים מטיפוסים שונים שיש לטפל בהם ע"י ממשק אחיד לכן עקרון זה אינו בא לידי ביטוי.

1. 1.

מספר יתרונות של מודל UML כעזר לתהליך התכנון:

באופן כללי, תרשימי ה-UML הם תרשימים ידידותיים וברורים למשתמש הקורא אותם.

ה-Use Case מאפשר להציג את הפעולות האפשריות של המשתמש מול המערכת,

ה-Class Diagram מאפשר למתכנת לתת ביטוי ויזואלי למחלקות השונות שמרכיבות את המערכת ואת הקשרים ביניהן ובכך לחלק את המערכת למודולים השונים שלה בצורה גראפית, דבר אשר חוסך זמן במהלך מימוש הקוד.

תרשימי ה-Activity וה-Sequence נותנים את האפשרות לצפות בתהליכים שונים המתקיימים במערכת, באינטראקציה של המשתמש מול המערכת ותגובתה ל-"כפתורים" השונים ב-UI.

תרשים ה-Sequence מתאר בצורה מעמיקה כיצד התרחיש המדובר מתחיל, באיזה מתודות משתמש וכיצד הוא מסתיים, דבר המקל על כתיבת הקוד והחלק הלוגי של התוכנית.

בנוסף, ה-Package Diagram מאפשר לנו למדל את המערכת בצורה נכונה ומקל עלינו בתהליך מימוש התוכנה.

מודל ה-UML נותן את האפשרות לצפות קשיים בשלבים המאוחרים יותר של תהליך הפיתוח וכך לקבל החלטות מראש. מה שעשוי להוזיל את עלויות וזמן הפיתוח.

מודל זה מאפשר גם לשתף רעיונות מופשטים כמו תבניות תכן שמסייעות אף הן למדל מערכת יותר יעילה ונכונה. כך גם לבצע שימוש חוזר ברעיונות קיימים שהוכחו כיעילים.

למשל במערכת -GHealth תיארנו ע"י הכלה חזקה (Composition) את היחס בין לקוח והתיק הרפואי שלו. כל לקוח חדש שנוצר במערכת נוצר עבורו תיק רפואי. תיק רפואי לא יכול להתקיים ללא לקוח ולקוח מבחינת המערכת חייב להחזיק תיק רפואי אשר מכיל את המידע וההיסטוריה הרפואית שלו.

1. 2.

חלק מהקשיים בהם נתקלנו בעקבות חסרונות ה-UML:

יצירת התרשימים הרבים הנדרשים אורכים זמן רב, בעקבות כך חילקנו את המשימות בין חברי הצוות. במהלך יצירת התרשימים לא היה ניתן לתאר ולהבין בצורה מוחלטת את מאגרי המידע. ניסינו בצורה המרבית לנסות לתאר ולהבין את המימוש שיגיע בשלב המאוחר יותר על ידי הידע שצברנו עד כה.

מידול ה-UML אינו מאפשר למדל בצורה ויזואלית את הממשקים. עם זאת הייתה הסכמה בין חברי הצוות לגבי הממשק הגרפי שנייצר בשלב הבא בפרויקט. לפי זה התבצעה חלוקת המחלקות של ה-boundary. בנוסף בממשק הגראפי אפשר לבצע מספר פעולות במקביל, דבר שטיפה התקשינו לתאר במודלים השונים.

ב.

בהתאם לניסיון שרכשנו במהלך העבודה במטלה זו, שימוש במתודולוגיית UML אינה נותנת מענה מלא לצרכי תהליך ה-Design שביצענו בפרויקט שלנו. אף תרשים לא נותן מענה לממשק הגרפי. אין בעצם אפשרות להציג את מבנה החלון שיופיע למשתמש.

בנוסף לא קיים תיאור למבנה מסד הנתונים במערכת, דבר שיכל להקל ביצירת המחלקות הדרושות לפעולת המערכת.

בשלב זה התבצע סיכום בין חברי הצוות על נראות הממשקים ואין לזה בסיס למימוש לאחר מכן.