תכנות מכוון עצמים - הקונספט

גיל כהן

כל הזכויות שמורות ל flashoo.co.il :: קמפוס הפלאש הישראלי

הקדמה

"מכוון עצמים" הוא מושג שנזרק לעיתים קרובות על ידי מתכנתים, אבל לא רק על ידם, "מכוון עצמים" לא קשור רק לתכנות אלא גם לעולם ארגון המידע וניהול מסדי נתונים, מערכות הפעלה, ממשקי משתמש מכווני עצמים וכו'. אנו נתמקד רק בשטח התכנות, עם זאת חשוב לזכור שהמתודולוגיה הזו מרחפת מעל שטחים נוספים.

שלושת האלמנטים הבסיסיים עליהם נשענת כל סביבת פיתוח מכוונת עצמים הם: מחלקות, הורשה וריבוי צורות. על אלמנטים אלו נרחיב במאמר, את המאמר נסכם בסיווג שפות מכוונות עצמים והקשר של פלאש לכל העניין.

מחלקות

אלמנט המפתח בתכנות מכוון עצמים הוא המחלקה (class). מחלקה היא **תיאור של קבוצת עצמים**, כאשר כל עצם נבדל מחברו על ידי **מצב**, כל עצם יכול לבצע פעולות מסוימות אשר מוגדרות במחלקה.

ניתן להתייחס למחלקה כאל סקיצה המתארת מבנה כלשהו, נניח בן אדם. המחלקה מתארת אלו תכונות קיימות לכל בני האדם ואילו פעולות אדם יכול לבצע. בין התכונות ניתן למצוא: גובה, משקל, צבע שיער, גיל, מין וכו'. הפעולות שאדם יכול לבצע הן רבות, אם לציין כמה מהן: הליכה, אכילה, דיבור, כתיבת מאמרים וכו'.

תפקיד המחלקה הוא לתאר אילו תכונות ופעולות קיימות לכל בני האדם. הזכרתי קודם לכן כי מחלקה היא **תיאור של קבוצת עצמים** וזה בדיוק מה שהגדרנו כאן, תיארנו קבוצת עצמים, קבוצת כל בני האדם. ציינתי גם כי כל עצם ועצם נבדל על ידי **מצב**, איך זה מתבטא בהקשר של מחלקת בני האדם שיצרנו? לכל אדם קיימות התכונות גובה, משקל, צבע שיער וכו', אבל לכל אדם ואדם יכולים להיות ערכים שונים בתכונות הללו, הווה אומר, גובהו של אדם אחד הוא 180 ס"מ ואילו של אחר 208 ס"מ. התכונה קיימת אצל כל בני האדם, אבל הערך שלה יכול להשתנות בין אדם לאדם. את ערכם של כלל התכונות מכנים בשם **מצב**.

הנקודה הזו כל כך אימננטית להבנת תכנות מכוון עצמים שאני רוצה להדגיש אותה שוב ולהתמקד במושגים **מחלקה** ו**עצם**. מחלקה היא הסכימה אשר מתארת קבוצה של עצמים, אין זה תפקיד המחלקה לתאר עצם או לבצע פעולות, תפקידה הוא לספק את כל המאפיינים הדרושים בכדי שאפשר יהיה לתאר עצם, ולהגדיר אילו פעולות העצם יכול לבצע וכיצד לבצע אותם.

נהוג להמשיל את העניין גם לקו יצור של מכוניות. בקו הייצור ישנה סכימה, בלו-פרינט של כלל המכוניות, הסכימה מתארת את המכוניות השונות אבל ודאי תסכימו שהסכימה לא יכולה לנסוע, וגובה המכונית המתואר בסכימה הוא לא גובה הסכימה שאם היא נמצאת על דף, גובהה כנראה שואף לאפס. הסכימה מטרתה היא לתאר כל עצם ועצם (מכונית ומכונית). כל מכונית יכולה להיות שונה מחברתה על ידי תכונות שונות (מצב) אבל הסכימה היא אחת.

כל הזכויות שמורות ל flashoo.co.il :: קמפוס הפלאש הישראלי

כיצד מכונית יכולה להיות שונה מחברתה אם כולן נוצרו על ידי אותה סכימה? ובכן, מכונית אחת יכולה להיות בצבע אדום והשנייה בכחול, מכונית אחת יכולה לנוע ברגע נתון במהירות מסוימת, הדבר לא ישפיע על שאר המכוניות ואלו ינועו בקצבים שונים. המשפט האחרון נשמע כה מובן מאליו וזאת בכדי שנוכל לחזור חזרה לרעיון המחלקה-עצם ולהבין שמחלקה מתארת קבוצת עצמים, כל עצם יכול להתבדל משאר העצמים למרות שנוצרו מאותה המחלקה.

רעיון המחלקות הומצא בכדי לקיים שני רעיונות בסיסיים בתכנות: מדולריות והיכולת להגדיר הפשטות.

המחלקות כאלמנט מפשט

שפות תכנות מציעות מספר מבני נתונים בסיסיים למדי, ביניהם: מספר, מחרוזת, טיפוס בוליאני וכו'. אמנם זו התקדמות ביחס לאפס ואחד אבל הפשטה קשה מאוד ליישם רק בעזרת מספרים ומחרוזות. איך אפשר לתאר בן אדם רק עם מספרים ומחרוזות? אפשר להגדיר משתנה אחד בשם age שיהווה את גיל האדם, משתנה בשם name שיכיל את שמו, משתנה נוסף בשם isMale שערכו יהיה אמת אם האדם המדובר הוא זכר ושקר אם ההיפך הוא הנכון. אבל מה יקרה אם נרצה ליצור עוד אדם? נצטרך להגדיר שוב את ערימת המשתנים שלנו, אם נתחכם נוכל ליצור מערך של משתנים מכל סוג וכל אדם יאופיין על ידי אינדקס מסוים בכל המערכים, הדרך הזו הייתה יכולה להצליח אבל נדמה כי מצאנו דרך טובה בהרבה שלא מפירה את עיקרון המודולאריות עליה נדון בהמשך.

בעזרת הגדרת מחלקות אנו בעצם מרחיבים את שפת התכנות עליה אנו עובדים, אנו יוצרים טיפוסים חדשים, דוגמת בני אדם, טיפוסים אלו מאפשרים לנו רמת הפשטה גבוהה הרבה יותר מאשר טיפוס נתונים מסוג מספר או מחרוזת. יצירה של עצם חדש כמוהו כיצירה של משתנה חדש. טכניקה זו עדיפה בהרבה, במקום להגדיר בכל פעם מחדש מה זה אדם, נגדיר זאת פעם אחת ואז נשתמש בהגדרה זו על ידי כך שניצור עצמים שונים מאותו הסוג, ניצור בני אדם שונים, גאוני.

המחלקות כאלמנט מודולארי

בעולם האלקטרוניקה נהוג לפתח שבבים, הרעיון מאחורי שבבים הוא שניתן לשלב כלל שבב בכל פרויקט ולדעת בדיוק מה השבב עושה ומה תפקידו וזאת מבלי לדעת כיצד הוא פועל מבפנים. לעיקרון הזה קוראים עיקרון הסתרת המידע והוא נועד בכדי לקיים עיקרון חשוב מאוד בפיתוח - שימוש חוזר (reusability).

הרעיון תחת שימוש חוזר (בקוד במקרה שלנו) הוא שניתן לבנות מודל (= קטע קוד) אשר מבצע פעולה מסוימת, ולהשתמש באותו המודל בפרויקטים שונים מבלי לבנות את אותו המודל בכל פעם מחדש. שימוש חוזר חוסך כמובן זמן פיתוח אבל זה לא נגמר כאן, מודל אשר השתמשו בו במקומות אחרים כנראה נקי יותר מבאגים שהרי הוא הורץ כבר במקומות אחרים, התחזוקה של הקוד קלה ומצומצמת לאותו המודל ולא מתפרסת על קטעי קוד רבים, שוב, גאוני.

כל הזכויות שמורות ל flashoo.co.il :: קמפוס הפלאש הישראלי

איך זה מתקשר למחלקות? הרעיון מאחורי מחלקות הוא לבצע את כל הלוגיקה באופן מוסתר מאלו אשר ישתמשו במחלקה, אלו רק צריכים לדעת כיצד להשתמש במחלקה ולא כיצד היא פועלת מבפנים, אם נחזור למחלקת בני האדם שלנו, נוכל להגדיר למחלקה פעולת אכילה, אלו שישתמשו במחלקה לא יצטרכו לדעת כיצד בדיוק פועלת הפעולה בכדי לבצע אותה, הם יצטרכו לדעת רק כיצד לבצע את הפעולה. נהוג להמשיל את העניין לקופסא שחורה, לאף אחד לא איכפת כיצד היא פועלת, רק כיצד להפעיל אותה. לי ברגעים אלו לא איכפת כיצד Word פועל, כל שאיכפת לי הוא שהוא פועל ואני יודע כיצד להפעיל אותו, כל הקצאות הזיכרון, ניהול התצוגה, שמירת הקבצים וכו' היא לא מענייני, אני רוצה שבלחיצה על Ctrl+S הקובץ ישמר, איך? זה העבודה של מיקרוסופט.

בכל מחלקה יש להגדיר אילו חלקים בקוד מהווים את הממשק אשר יוצג בפני המשתמש במחלקה ואילו חלקים יוסתרו, זאת בכדי לשמור על רעיון הקופסא השחורה, בשפות תכנות רבות ניתן לקבוע דרגת הרשאה מתוך שלוש קיימות: ציבורי, מוגן ופרטי. אל קוד אשר מוגדר כציבורי משתמש במחלקה יוכל לגשת מכל מקום, קוד אשר מוגדר כפרטי לא יהיה ניתן לגישה מחוץ לתחום המחלקה ולרוב ישמש את הלוגיקה הפנימית של המחלקה. קוד המוגדר כמוגן יהיה ניתן לגישה על ידי תת מחלקות (עליהן נדון בסעיף הדן בהורשה).

הורשה (inheritance)

הרעיון מאחורי ההורשה גם הוא נוגע לשימוש חוזר בקוד (reusability), הרעיון בהורשה הוא ליצור מחלקה על בסיס מחלקה אחרת ולא מבראשית. איך הורשה פועלת? הדרך הכי טובה להבין זאת היא ראשית להסתכל על דוגמא, אם נרצה לבנות משחק כדורגל, נצטרך לתאר את השחקנים השונים, מכיוון שכל שחקן הוא דבר ראשון בן אדם (ויש כאלו ששוכחים את זה לקראת המונדיאל), נוכל להגדיר מחלקה שתתאר שחקן על בסיס מחלקת בני האדם ולא מאפס. הרי לכל שחקן יש גובה, משקל, צבע עיניים וכו', בנוסף ישנן תכונות שנרצה לייחד רק לשחקני כדורגל כמו קבוצה בה משחק השחקן, גולים לעונה, שכר וכו'. במקום להמציא את הגלגל מחדש נוכל להתבסס על מחלקת בני האדם, **לרשת** אותה ולהוסיף תכונות ופעולות נוספות אל תוך מחלקת שחקני הכדורגל שלנו.

הרעיון מאחורי הורשה הוא ליצור תת קבוצות מקבוצה קיימת, בדוגמא שלנו הקבוצה הקיימת היא קבוצת בני האדם, תת קבוצה אחת של בני האדם היא שחקני כדורגל ובעזרת הורשה קל יותר ליצור תת קבוצה זו מפני שאין צורך לכתוב מחדש את אותם החלקים הזהים בין המחלקות.

כאשר מתבצעת הורשה קיים קשר בין שתי המחלקות, למחלקה היורשת קוראים תת-מחלקה או מחלקה נגזרת, ואילו למחלקה המקורית, ממנה יצרנו את המחלקה החדשה, נהוג לקרוא כיתת בסיס.

באחד המאמרים על הורשה [Eckel 89] נכתב "ממש כפי שילד יורש את תכונות הוריו, מחלקה נגזרת (הנקרא גם תת-מחלקה) יורשת את המאפיינים מכיתת בסיס (הנקראת גם כיתת-על). באמצעות הגזירה אנו אומרים: "המחלקה החדשה הזו היא המחלקה הישנה עם מספר שינויים" למשל, גורד שחקים הוא בניין עם קומות רבות."

כל הזכויות שמורות ל flashoo.co.il :: קמפוס הפלאש הישראלי

אם לסכם זאת במילים, הורשה ממחלקה אחת לאחרת יכולה לקחת מקום אם ורק אם מחלקה א' היא חלקית למחלקה ב', באנגלית ניתן לומר שהורשה לוקחת מקום אם מחלקה א' is a מחלקה ב'. לדוגמא, כל מלבן הוא (is a) מרובע או בדוגמא הקודמת שלנו כל שחקן כדורגל הוא (is a) בן אדם. הורשה לא יכולה להתקיים תמיד, לדוגמא כוס הוא לא צלחת ולא כל בני האדם הם שחקני כדורגל.

אילו יתרונות קיימות בהורשה? היתרון העיקרי הוא חסכון בכתיבת קוד קיים, שימוש חוזר בקוד משפיע ישירות על זמן הפיתוח, התחזוקה וניפוי הבאגים של פרויקטים. יתרון נוסף הוא שבכדי לרשת מחלקה קיימת אין צורך להבין כיצד היא פועלת מבפנים, כך מפתחים שונים יכולים להרחיב מחלקות של מפתחים אחרים מבלי להתעניין ולבזבז את זמנם על הבנת הקוד המפעיל את המחלקה. היתרון השלישי נעוץ ברעיון המכונה ריבוי צורות או פולימורפיזם עליו נרחיב מיד.

לסיום אציין כי הורשה לא מוגבלת רק על ידי מחלקת בסיס אחת, ניתן ליצור מחלקה חדשה על סמך יותר ממחלקת בסיס אחת, אני לא ארחיב בנושא, במיוחד לאור העובדה שנושא זה הוא שינוי במחלקות בקרב מפתחים רבים, יש הטוענים שהוא מיותר לחלוטין ואחרים מקצינים ואומרים כי הורשה מרובה סותרת את עקרונות תכנות מכוון העצמים.

ריבוי צורות (Polymorphism)

Polymorphism - פירושו ריבוי צורות. הכוונה באופן מופשט מה היא שעצם אחד יוכל ללבוש מספר צורות. אין ספק שזה המקום לדוגמא בכדי להבהיר את העניין.

הבה נגדיר מחלקה שתתאר את כל היונקים הקיימים ונגזור ממנה מחלקות שונות המהוות תת קבוצות של יונקים כמו בני אדם, דולפינים וסוסים. כל יונק אוכל, ולכן מן הראוי שכבר בתיאור המחלקה נגדיר את פעולת האכילה של היונקים. הבעיה היא שדולפין אוכל באופן שונה מאשר סוס או בן אדם, ניתן לומר שהוא מגיב באופן שונה לאותה הפונקציה, אותה הפעולה. בעזרת פולימורפיזם עצם מסוג יונק יוכל להתאים את פעולת האכילה המתאימה לו, העצם ידע שהוא דולפין, סוס או בן אדם וידע להגיב באופן מתאים לפונקצית האכילה.

אני מאמין כי הערכה אמיתית לריבוי צורות תבוא רק עם הבנה מעמיקה של רעיון ההורשה. את רעיון ההורשה לא ניתן כמובן להבין לעומק רק ממאמר זה הבא לתאר בקצרה את מתודולוגית תכנות מכוון עצמים, אבל חשוב להכיר את התכונה הזו ולדעת בקווים כללים מה מטרתה.

כל הזכויות שמורות ל flashoo.co.il :: קמפוס הפלאש הישראלי

סיווג שפות מכוונות עצמים

שפות מכוונות עצמים קיימות כיום בשפע, אבל לא כולן תומכות בתכנות מכוון עצמים באותה האדיקות, בכדי לתאר את רמת התמיכה בתכנות מכוון עצמים הוגדרו שלושה דרגות תמיכה:

שפות מבוססות עצמים (Object based) – אלו שפות התומכות בעצמים, כאשר הכוונה היא לשפות אשר מכילות מרכיבים המאפשרים לבצע פעולות ונמצאים במצב מסוים.

שפות מבוססות מחלקות (Class based) – אלו שפות המאפשרות ליצור מחלקות על ידי יצירת מאפיינים ושיטות לכל מחלקה. מהמחלקות הללו ניתן ליצור עצמים שונים. בעצם שפות אלו מאפשרת הרחבה של השפה על ידי בניית מחלקות.

שפות מכוונות עצמים (Object Oriented) – שפות אלו תומכות גם כן ביצירת מחלקות ועצמים מהמחלקות הללו. שפות אלו גם מציעות מנגנון הורשה ופולימורפיזם.

הקשר לפלאש

איך פלאש או יותר נכון ActionScript בגרסת ה MX מתקשרת לכל עניין תכנות מכוון העצמים? ובכן, ActionScript בנויה על סמך תקן ECMA (European Computer Manufacturers Association), התקן מהווה המלצה לשפה מכוונת עצמים ולכן ActionScript אכן מכוונת עצמים, אם כי לא באותה הרמה של שפות תכנות אחרות כמו C++ ו Java.

ActionScript תומכת במרכיבים רבים של תכנות מכוון עצמים אבל מדלגת על אחרים. ActionScript תומכת ביצירת מחלקות ועצמים מתוך המחלקות הללו, היא לא תומכת בהסתרת מידע על ידי הגדרת אלמנטים מסוימים כפרטיים או מוגנים, AS תומכת בהורשה אך לא בהורשה מרובה ולסיום, AS לא תומכת בפולימורפיזם.

השילוב של פלאש עם ActionScript הופך את תכנות מכוון האובייקטים למעניין. כל סמל אשר נוצר בפלאש ונשמר בספרייה מהווה עצם מסוג MovieClip. לא ניתן לומר כי אותו סמל מהווה תת מחלקה של MovieClip למרות שהגיוני לחשוב כך. אם סמל הוא מחלקה אז ניתן להתייחס למופעים השונים של הסמל על הבמה כאל עצמים הנוצרים מאותה המחלקה. בעצם שימוש במושג מופע מתייחס למושג עצם או אובייקט, בעוד סמל הוא לא אחר מאשר מחלקה.

ניתן גם לבנות אובייקטים משלך ב ActionScript, להוסיף להם שיטות (פעולות) ומאפיינים (תכונות), כמו בכל שפת תכנות מכוונת עצמים המכבדת את עצמה, את הדרך ליישום העניין לא נכסה במאמר זה אשר נוגע רק בצד התיאורטי של העניין. לסיום אציין כי רעיון הקומפוננטות, אשר מהווה את אחד השדרוגים של Flash MX על פני גרסאותיו הקודמות, הוא לא אחר מאשר בניית מחלקות נגזרות מהמחלקה Movieclip.

כל הזכויות שמורות ל flashoo.co.il :: קמפוס הפלאש הישראלי

סיכום

במאמר זה למדנו על שלושת האלמנטים הבסיסיים המרכיבים את תכנות מכוון העצמים: מחלקות, הורשה ופולימורפיזם. למדנו על היתרונות ביצירת מחלקות בהיבט של מודולאריות והפשטה, למדנו על הורשה והיתרונות הכורכים בה והזכרנו בקצרה את הרעיון העומד מאחורי ריבוי צורות (פולימורפיזם). בהמשך למדנו על הסיווג של שפות תכנות מכוונות עצמים כפונקציה של התמיכה שלהן במתודולוגיה ודנו בקשר שבין פלאש לתכנות מכוון עצמים.

כל הזכויות שמורות ל flashoo.co.il :: קמפוס הפלאש הישראלי