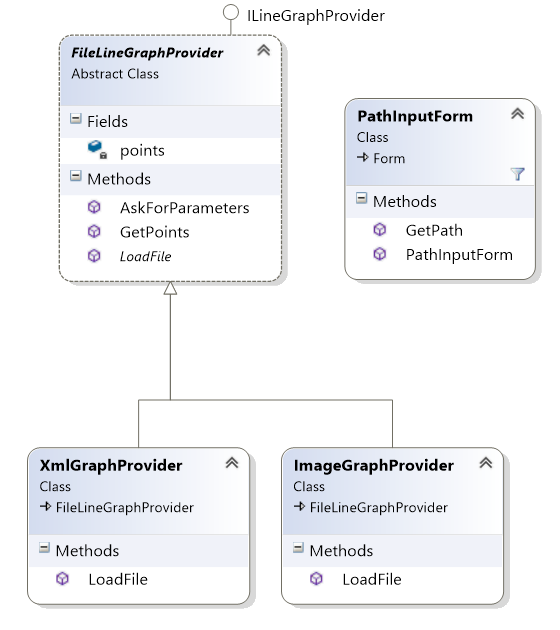
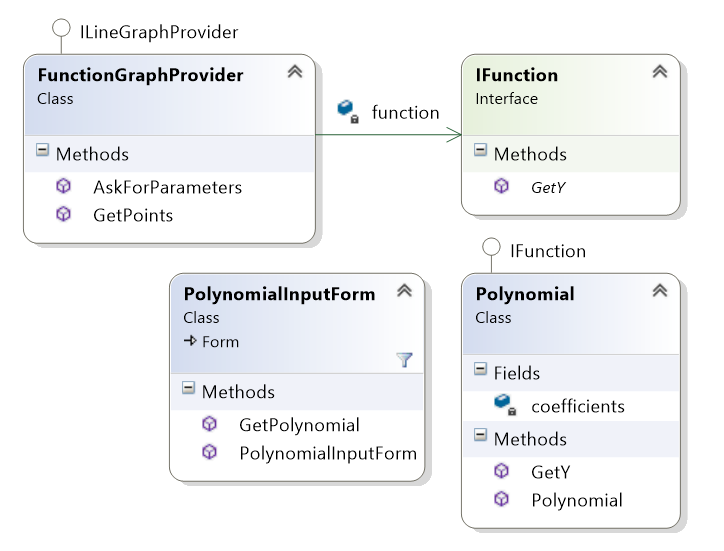
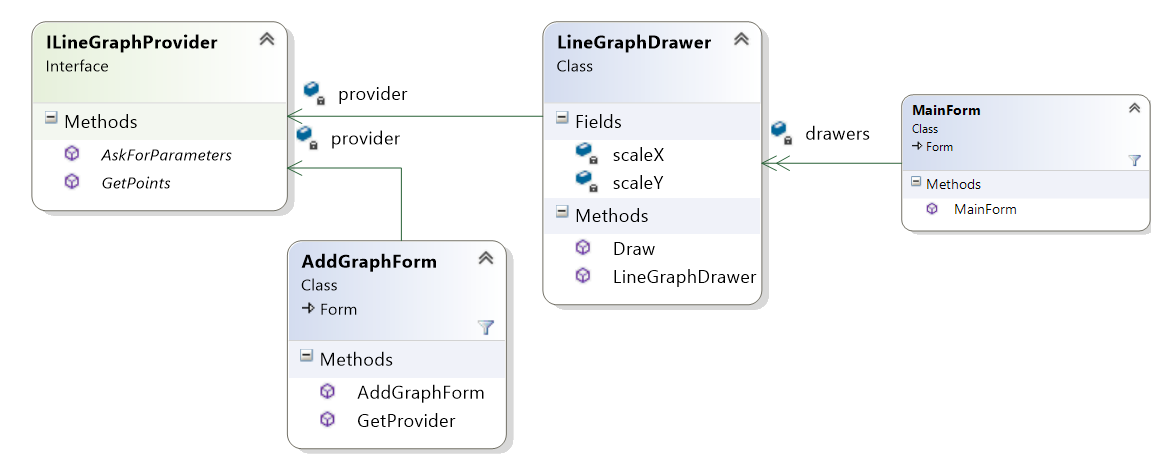
מטלת ביצוע 2 – ציור גרפים  
תכנון וחלוקת העבודה

# תרשים UML



# סיכום המחלקות

**ILineGraphProvider** – מגדיר את הממשק של מקור גרף קוי. מכיל פעולה שמבקשת מהמשתמש להכניס פרמטרים עבור המקור, ופעולה שמחזירה את הנקודות בתור **IEnumerable<PointF>** (**PointF** מוגדרת ב- System.Drawing). הפעולה גם מקבלת פרמטר מסוג float שהוא המרווח הקטן ביותר שניתן להבחין בו, כלומר הגודל של פיקסל אחד לאחר שקלול קנה המידה, וטווח של ציר ה-X.

על **IEnumerable**, ראה ב"שיקולי התכנון".

**FunctionGraphProvider** – מספקת גרף ע"י פונקציה (פולינום). הפרמטרים שהיא מבקשת מהמשתמש הם מקדמי הפולינום (באמצעות **PolynomialInputForm**). לשם כך, יש לה תכונה מסוג **IFunction** שכרגע המחלקה היחידה שמממשת אותו היא פולינום.

**FileLineGraphProvider** – מחלקה מופשטת עבור מקורות גרפים שמשתמשים בקובץ. הפרמטר שהיא מבקשת מהמשתמש הוא רק מיקום הקובץ (באמצעות **PathInputForm**), ולאחר שהיא מקבלת אותו היא קוראת לפעולה המופשטת שלה *LoadFile*, ושומרת את הנקודות שהתקבלו בתכונה, כדי להשתמש בהן אחר כך.

**ImageGraphProvider** – מספקת גרף ע"י תמונה. בפעולה LoadFile, היא מחזירה את המיקומים של כל הפיקסלים שאינם לבנים (אלה הנקודות של הגרף).

**XmlGraphProvider** – מספקת נקודות מקובץ XML, ע"פ פורמט XML מוגדר מראש.

**LineGraphDrawer** – מחלקה שתפקידה לצייר גרף ממקור אחד, ע"פ קנה מידה מוגדר (בנפרד לכל ציר). הפעולה המרכזית בה היא Draw, שמקבלת Graphics באמצעותו לצייר.

**MainForm** – המסך העיקרי של התכנית, שמאפשר למשתמש להוסיף ולראות גרפים (ע"י **AddGraphForm**), לשנות את קני המידה של הצירים עבור כל הגרפים ביחד, ולנוע על פני מערכת הצירים באמצעות גרירת העכבר.

**AddGraphForm** – טופס שמבקש מהמשתמש סוג של גרף וקני מידה, ומאתחל **LineGraphDrawer** על פיהם.

# שיקולי התכנון

* אחד הדברים הראשונים שהחלטנו עליהם הוא שטיפוס ההחזרה של ILineGraphProvider.GetPoints יהיה IEnumrable<PointF>, שהוא ממשק מובנה בשפה, שמייצג רצף או אוסף שניתן לסרוק אותו בצורה ליניארית. חוץ מהעובדה שהממשק מותאם בדיוק למשימה, השימוש בו הופך את ILineGraphProvider לגמיש יותר, בעזרת שימוש במילת yield.
* שלושת סוגי המקורות נבחרו מכיוון שהם נראו לנו מגוונים, מאתגרים מספיק, והם עונים על דרישות המטלה.
* היתרון של מחלקת **FileLineGraphProvider** הוא שהמחלקה מאגדת את הטיפול בפתיחת קבצים, ששתיים מהמחלקות האחרות משתמשות בו. אם זאת, המחלקה כפי שהיא מתוכננת כרגע לא מאפשרת למחלקות היורשות לבקש פרמטרים נוספים מהמשתמש (מלבד אלה שבקובץ), או להשתמש בפרמטר של המרווח של פיקסל אחד. החלטנו לא לספק לה את הפונקציונליות הזו כרגע, כי היא לא הכרחית לגרף שגם ככה מקבל את הנקודות שלו מקובץ מוגדר מראש שלא צפוי להשתנות לאחר שהוא כבר נקרא.
* הממשק **IFunction** נועד לדאוג שניתן יהיה להרחיב את השימוש של **FunctionGraphProvider** בעתיד, למרות שבתכנון הנוכחי יש לו רק מממש אחד (פולינום), שבעצם קשור באופן הדוק מדי למחלקה (בשלב הקלט). זאת משום שהכנה כזו תדרוש יותר מדי עבודה ביחס ליתרון שלה בעבודה זו.
* היתרון של התכנון שביצענו הוא שניתן בקלות לראות את מבנה התכנה ע"פ תרשים ה-UML, וההפרדה לממשקים ומחלקות בעלות אחריות מצומצמת מקלה על העבודה ועל האפשרות להרחיב את יכולות התכנה בעתיד.

# חלוקת העבודה

אמיר – XmlGraphProvider

יואב – FunctionGraphProvider

אביב – ImageGraphProvider

איל- MainForm ו-LineGraphDrawer

האחריות על המחלקות כוללת את טפסי הקלט הרלוונטיים.