דיאגרמה לתיאור Classes

הקדמה לדיאגרמת Classes

כללים במתן שמות ל-Classes שאנו מתארים

הרשאות גישה של מרכיבי ה-Class השונים

משתנים (Attributes) שמוגדרים ב-Class

משתנים סטטיים (Static Attributes)

מתודות (Operations) שמוגדרים ב-Class

תיאור תקלות שעלולות להתרחש (Operations Exceptions)

מתודות סטטיות (Static Operations)

מחלקה אבסטרקטית (Abstract Class)

קשר של Dependency בין Classes שונים

קשר של Association בין Classes שונים

קשר של Aggregation בין Classes שונים

קשר של Composition בין Classes שונים

תיאור Class שיורש מ-Class אחר

תיאור קשר בין שני Classes באמצעות Association Class

תיאור Interfaces ו-Classes שמיישמים אותם

תיאור Classes שפועלים כ-Templates

© 2009 Haim Michael. All Rights Reserved.

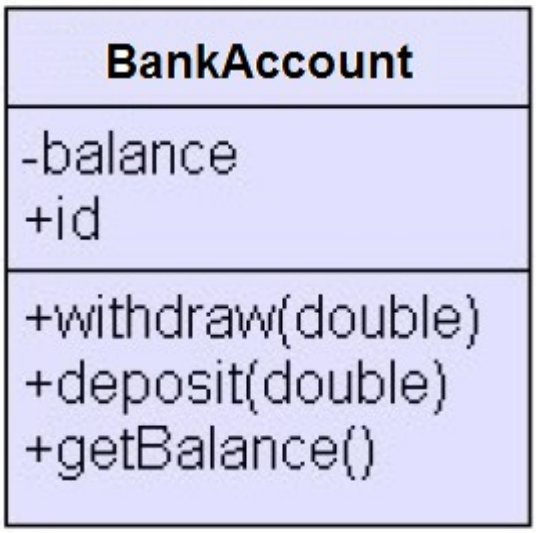
הקדמה לדיאגרמת Classes

באמצעות Class Diagram ניתן לתאר את המחלקות שאנו עומדים להגדיר, את הקשרים שיש ביניהן ואת המאפיינים (ה-attributes) והמתודות (ה-operations) שכל Class יכלול בהגדרתו.

באמצעות Class Diagram ניתן להציג את המערכת המתוכננת מנקודת ראות סטטית בלבד. ה-Class Diagram מציג את ה-Classes השונים שיוגדרו ואת הקשרים שיש ביניהם (קשרים שבדרך כלל באים לידי ביטוי בקשרים בין האובייקטים שנוצרים מה-Classes המתוארים). ה-Class Diagram לא מתייחס לעיתוי שבו המתודות (ה-operations) השונות תופעלנה והוא לא מתייחס לעיתוי שבו יווצרו האובייקטים מה-Classes המתוארים.

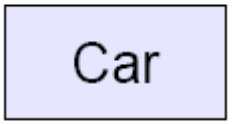
כל Class מתואר באמצעות מלבן אשר מחולק לשלושה חלקים. בחלק העליון מקובל לרשום את שם ה-Class. בחלק האמצעי מקובל לפרט את המשתנים שיוגדרו בו ובחלק התחתון מקובל לפרט את ה-operations (המתודות) ואת ה-constructors (הבנאיים).

ה-Class Diagram הבא מתאר את ההגדרה של ה-class ששמו BankAccount, אשר כולל הגדרה של שני משתנים (attributes) ושלוש מתודות (operations). את שם ה-class מקובל לרשום בגופן מודגש בגודל גדול יותר בהשוואה לגודל של הגופן שמשמש לכתיבת יתר הטקסטים שמופיעים בדיאגרמה.

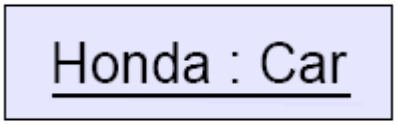


© 2009 Haim Michael. All Rights Reserved.

ניתן לתאר Class גם באמצעות מלבן אשר כולל את שמו בלבד, מבלי להתייחס למשתנים (ה-attributes) והמתודות (ה-operations) שמוגדרים בו. כך לדוגמא, הדיאגרמה הבאה מתארת את ה-class ששמו Car.



ניתן לתאר אובייקט אשר נוצר מ-class מסויים באמצעות מלבן אשר כולל בתוכו את שם האובייקט, הסימן ':' לימינו, מייד לאחריו שם ה-class שממנו האובייקט נוצר ומתחת לכל הטקסט קו תחתי רציף. הדיאגרמה הבאה מציגה תיאור של אובייקט ששמו Honda אשר נוצר מה-class ששמו Car.



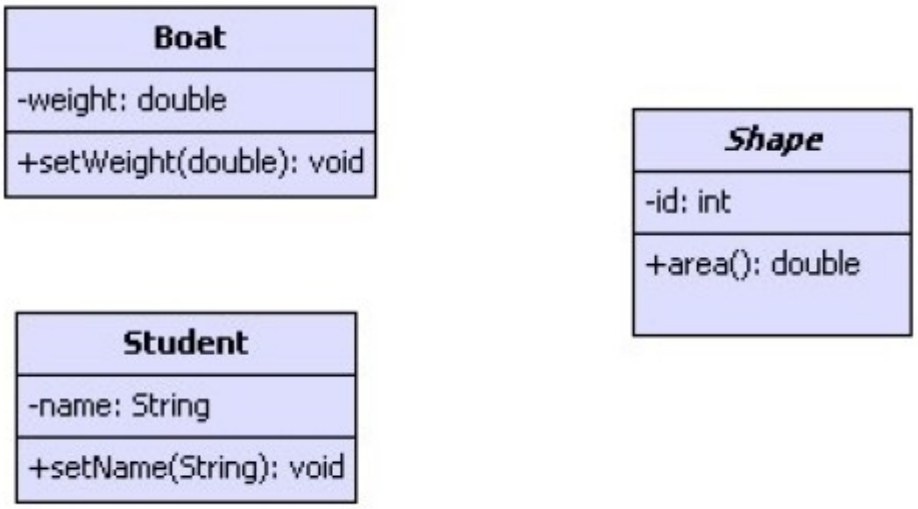
בדרך זו ניתן לייצור תרשימים אשר מתארים את האובייקטים שנוצרים מה-classes השונים ואת הקשרים ביניהם (Objects Diagram).

© 2009 Haim Michael. All Rights Reserved.

כללים במתן שמות ל-Classes

את שם ה-class מקובל לרשום באותיות מודגשות ובגופן שגודלו גדול יותר מגודלו של הגופן שמשמש לכתיבת שאר הטקסט בדיאגרמה. כמו כן, מקובל לרשום את שם ה-class (בדומה לכללי הכתיבה המקובלים ב-Java) באותיות קטנות כאשר האות הראשונה של השם (וכמו כן האות הראשונה של כל אחת מהמילים שמרכיבות את השם) היא אות גדולה. כאשר מתארים Abstract Class מקובל לרשום את שמו בגופן מסוג italic (מוטה הצידה).

הדיאגרמה הבאה כוללת תיאור של ה-classes הרגילים Boat ו- Student ותיאור של ה-abstract class ששמו Shape.



.

© 2009 Haim Michael. All Rights Reserved.

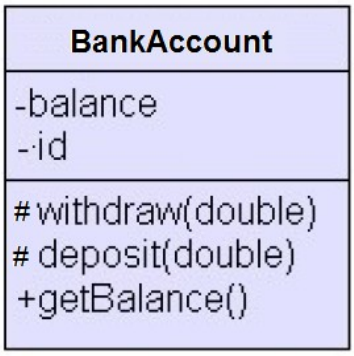
הרשאות גישה של מרכיבי ה-Class השונים

תרשימי UML מאפשרים מתן הרשאת גישה בנפרד לכל משתנה (attribute) ולכל מתודה (operation) שמוגדרים ב-class. הרשאות הגישה האפשריות הן:

הרשאת הגישה Privte אשר מתוארת באמצעות הסימן הגרפי -
הרשאת הגישה Public אשר מתוארת באמצעות הסימן הגרפי +
הרשאת הגישה Protected אשר מתוארת באמצעות הסימן הגרפי #
הרשאת הגישה Package אשר מתוארת באמצעות הסימן הגרפי ~

המשמעות שיש לכל אחת מהרשאות הגישה הללו דומה לזו שמוכרת משפת התיכנות Java. משתנה ו/או מתודה עם הרשאת הגישה private יהיו נגישים באופן ישיר רק אם הניסיון לפנות אליהם ייכתב בגבולות ה-class שבו הם מוגדרים. משתנה ו/או מתודה עם הרשאת הגישה public יהיו נגישים באופן ישיר מכל מקום. משתנה ו/או מתודה עם הרשאת הגישה package יהיו נגישים באופן ישיר רק עם הניסיון לפנות אליהם ייכתב בגבולות של class אשר שייך ל-package שאליו שייך ב-class שבו הם מוגדרים. משתנה ו/או מתודה עם הרשאת הגישה protected יהיו נגישים באופן ישיר מכל מקום שבו הם היו נגישים אילו הרשאת הגישה שלהם הייתה package ובנוסף גם מ-classes אשר שייכים ל-package אחר ובלבד שמדובר ב-classes אשר יורשים מה-class שבו הם מוגדרים.

התרשים הבא מתאר הגדרה של class אשר כולל שני משתנים ושלוש מתודות. המשתנה balance והמשתנה id מוגדרים עם הרשאת הגישה private ובכך מובטח שהגישה הישירה אליהם אפשרית בגבולות ה-class המוגדר בלבד. המתודות withdraw ו-deposit מוגדרות עם הרשאת הגישה protected ובכך מובטח שהגישה הישירה אליהם לצורך קריאה להפעלתן אפשרית רק אם היא נכתבת בגבולות ה-class המוגדר או בגבולות של class אחר ששייך לאותו package או בגבולות של class אחר ששייך ל-package אחר אך שיורש מה-class המוגדר. המתודה getBalance מוגדרת עם הרשאת הגישה public ולכן היא ניתנת להפעלה מכל מקום.



© 2009 Haim Michael. All Rights Reserved.

משתנים (Attributes) שמוגדרים ב-Class

ב-Class Diagram כל classs מתואר באמצעות מלבן נפרד אשר מחולק לשלושה חלקים. בחלק העליון רושמים את שם ה-class. בחלק האמצעי מפרטים את המשתנים (ה-attributes) אשר יוגדרו בו.

כל משתנה (attribute) יתואר בשורה נפרדת באופן הבא:

visibility / name : type multiplicity = default {properties and constraints}

לא כל המרכיבים בתיאור של משתנה (attribute) בודד הם מרכיבים שחובה לכלול. ניתן גם להסתפק ב-name בלבד.

ה-visibility יכול להיות כל אחת מהאפשרויות הבאות:

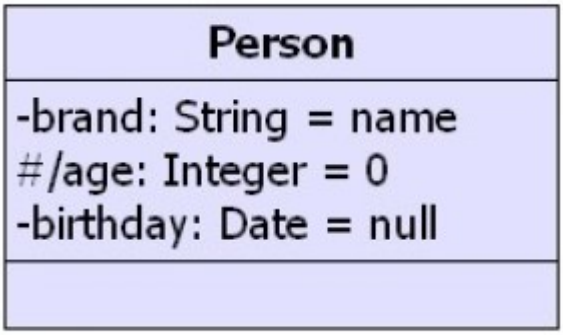
public +

private -

protected #

package ~

הסימן / מתאר מצב שבו ערכו של המשתנה המתואר הוא Derived Attribute. משתנה שנחשב ל-Derived Attribute הוא משתנה שערכו תמיד מתבסס על ערכו (או ערכם) של משתנה (או משתנים) אחר(ים). כך למשל, בדוגמא הבאה ערכו של משתנה כגון age בתיאור של ה-class ששמו Person מתבסס על ערכו של המשתנה birthday אשר מכיל את תאריך יום ההולדת שלו.



© 2009 Haim Michael. All Rights Reserved.

ה-name הוא השם שאנו בוחרים לתת ל-class המתואר. בדיאגרמה של הדוגמא האחרונה מתואר class ששמו Person.

ה-type שמופיע מייד לאחר הנקודותיים יכול להיות או שם של class אחר או שם של אחד מה-primitive types שבהם שפת התרשימים UML תומכת ובהתאם לכלי שעימו יוצרים את תרשים ה-UML.

מייד לאחר ציון ה-type של המשתנה ניתן לציין את ה-multiplicity שלו. בציון multiplicity של משתנה ניתן לתת ביטוי לכך שמשתנה מסויים משמש להחזקת מספר גדול מ-1 של ערכים באמצעות מערך או מבנה נתונים אחר.

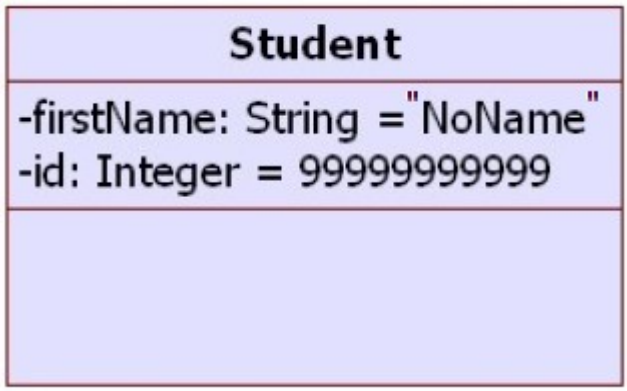
את ה-multiplicity מציינים באמצעות מתן טווח ערכים אפשרי בצירוף שתי נקודות בין שני הערכים וסוגריים מרובעות אשר תוחמים אותם. כך למשל, בדוגמא הבאה ה-multiplicity של המשתנה wheels נמצא בטווח הערכים 2 ו-3. המשמעות היא שבכל אובייקט שיווצר מה-class ששמו Motorcycle יהיה משתנה ששמו wheels והוא ייצג בין 2 ל-3 ערכים (כלומר, כל אובייקט ייצג אופנוע עם 2 או 3 גלגלים). באופן דומה, ה-multiplicity של המשתנה seats הוא בין 1 ל-3 (כולל) והמשמעות לכך היא שכל אובייקט שייווצר מ-Motorcycle ייצג אופנוע עם מושב אחד או שניים או שלושה.

כאשר לא מציינים את ה-multiplicity של משתנה מסויים ברירת המחדל היא 1. זהו המקרה הרגיל.

כאשר משתמשים בסימן * כדי לציין את ה-multiplicity אז המשמעות היא שהמשתנה האמור ישמש לייצוג ערך אחד או יותר (עד אינסוף) או שלא ישמש לייצוג אף ערך. בדוגמא הבאה ה-multiplicity של speakers היא *, והמשמעות היא שכל אובייקט שיווצר מ-Motorcycle ייצג אופנוע ללא רמקולים או אופנוע עם רמקול אחד או אופנוע עם מספר אינסופי של רמקולים.

© 2009 Haim Michael. All Rights Reserved.

לאחר ציון ה-multiplicity ניתן למקם את סימן השיוויון ומייד לאחריו את ערך ברירת המחדל של המשתנה המוגדר.
בדוגמא הבאה ערך ברירת המחדל של המשתנה firstName הוא המחרוזת 'NoName' וערך ברירת המחדל של המשתנה id הוא המספר 99999999999.



בהמשך, לאחר ציון ה-default value ניתן לציין בתוך סוגריים מסולסלות properties ו-constraints שמספקים תיאור נוסף למשתנה (ה-attribute) המתואר.

ה-properties האפשריים כוללים את האפשרויות הבאות:

readOnly
כאשר רושמים בתוך סוגריים מסולסלות בשורה שמתארת את המשתנה readOnly המשמעות היא שלאחר שנוצר אובייקט מה-class המתואר הערך שנמצא בתוך המשתנה האמור לא ניתן לשינוי. ניתן לפנות אל המשתנה כדי לקבל את ערכו אך לא ניתן לפנות אליו כדי לשנותו. השימוש ב-readOnly מתאים לאותם מקרים שבהם אנו מעוניינים לתת ביטוי לעובדה שמשתנה מסויים מרגע איתחולו לא ניתן לשינוי. דוגמא קלאסית יכולה להיות המשתנה id אשר מחזיק את ערך תעודת הזהות של אובייקט מטיפוס Person. מרגע היווצרות האובייקט ואיתחולו של המשתנה id בערך התחלתי לא ניתן לשנותו.

union
כאשר מציינים שמשתנה מסויים נחשב ל-Union באמצעות הוספת המילה 'union' אל תוך הסוגריים המסולסלות בשורה שמתארת את המשתנה האמור המשמעות היא שערכו של המשתנה האמור יכול להיות אחד מתוך קבוצה של ערכים אפשריים. כך למשל, ניתן באמצעות הוספת המילה union לתאר מקרה שבו derived variable יכול להכיל ערך שהוא אחד מתוך קבוצה של ערכים האפשריים של משתנים אחרים.

© 2009 Haim Michael. All Rights Reserved.

Subset <Attribute Name>

באמצעות הוספת property זה נותנים ביטוי לכך שהמשתנה המתאור יכול להכיל ערך שהוא אחד מתוך קבוצה ערכים אפשריים של משתנה (Attribute) אחר.

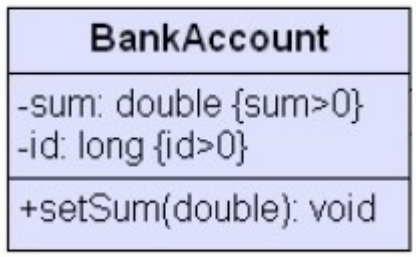
Redefines <Attribute Name>

באמצעות הוספת property זה נותנים ביטוי לכך שהמשתנה המתאור מהווה למעשה alias (שם נרדף) למשתנה אחר.

Composite

באמצעות הוספת property זה נותנים ביטוי לכך שהמשתנה המתאור לוקח חלק בקשר שבין המחלקה המתוארת ומחלקה אחרת.

ה-constraints מתוארים באמצעות ביטוי בוליאני או באמצעות טקסט פשוט. כך למשל, בדוגמא הבאה מתואר ה-class ששמו BankAccount אשר כולל את המשתנה sum שמתואר כמשתנה אשר כל ערך אפשרי שלו יהיה בהכרח חיובי או 0. המשתנה id מתואר כמשתנה שהערך האפשרי שלו חייב להיות חיובי.



© 2009 Haim Michael. All Rights Reserved.

משתנים סטטיים (Static Attributes)

משתנים סטטיים מתוארים באמצעות הוספת קו תחתי מתחת לשורה שמתארת את המשתנה. בדוגמא הבאה מתואר המשתנה הסטטי bank.

© 2009 Haim Michael. All Rights Reserved.

מתודות שמוגדרות במחלקה (Operations)

ב-Class Diagram כל class מתואר באמצעות מלבן נפרד אשר מחולק לשלושה חלקים. בחלק העליון רושמים את שם ה-class. בחלק האמצעי מפרטים את המשתנים (ה-attributes) אשר יוגדרו בו. בחלק התחתון מפרטים את המתודות (ה-operations) אשר יוגדרו בו.

כל מתודה (operation) תתואר בשורה נפרדת בפורמט הבא:

visibility name (parameters): return_type {properties & constraints}

לא כל המרכיבים חייבים להופיע.

ה-visibility שניתן לציין הוא כל אחת מהאפשרויות שזמינות בעת ציון ה-visibility של משתנה (attribute):

+

-

#

~

את השם של המתודה (ה-operation) מקובל לכתוב באותיות קטנות ובמידה שהוא כולל יותר ממילה אחת מקובל כל מילה (החל מהמילה השניה) לכתוב עם אות ראשונה גדולה.

כל פרמטר שמצויין בתוך הסוגריים העגולות חייב להופיע בפורמט הבא:

direction parameter_name : type [multiplicity] = default_value

לא כל המרכיבים חייבים להופיע.

ה-direction יכול להיות כל אחת מהאפשרויות הבאות:

in

כאשר מדובר בפרמטר שמשמש להעברת ערך אל המתודה (ה-operation) בלבד.

© 2009 Haim Michael. All Rights Reserved.

out

כאשר מדובר בפרמטר שמשמש לקבלת ערך בחזרה מהמתודה (ה-operation). כך למשל, כאשר מדובר בתכנית ב-Java פרמטר זה יכול להיות משתנה שיקבל לתוכו reference של מערך שהמתודה תמלא בנתונים.

inout

כאשר הפרמטר משמש גם להעברת ערך אל המתודה (ה-operation) וגם לקבלת ערך בחזרה.

return

כאשר הערך שמועבר על ידי הפרמטר למתודה הוא גם הערך שהמתודה תחזיר. כך למשל, כאשר מדובר בתכנית שכתובה ב-Java פרמטר זה יכול להיות משתנה שיקבל לתוכו reference לאובייקט ואותו reference בדיוק גם יהיה הערך שהמתודה תחזיר.

מקובל לתת לפרמטרים שמות שמורכבים מאותיות קטנות בלבד, ובמקרה שהשם כולל יותר ממילה אחת אז החל מהמילה השניה לכתוב את האות הראשונה של כל מילה כאות גדולה.

ה-type של כל פרמטר יכול להיות כל type שיכול להיות למשתנה רגיל (attribute) שמוגדר ב-class.

ה-multiplicity שניתן לקבוע לכל פרמטר דומה לזו שניתן לקבוע לכל משתנה רגיל (attribute) שמוגדר ב-class.

בדומה להגדרה של משתנים רגילים (attributes) ב-class ניתן גם בהגדרה של פרמטרים לקבוע עבורם ערכי ברירת מחדל. כדי לעשות זאת יש להוסיף את הסימן '=' ולאחריו את ערך ברירת המחדל שרוצים לקבוע.

בהמשך, לאחר שמפרטים את הפרמטרים, ניתן להוסיף את הסימן ':' ומייד אחריו לציין את ה-type של הערך המוחזר.

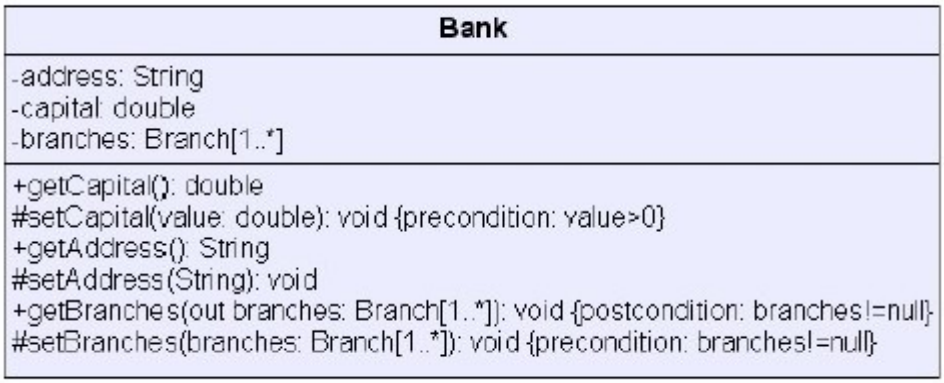
ה-type המוחזר יכול להיות כל אחד מה-types האפשריים בקביעת ה-type של משתנה (property) שמוגדר במחלקה.

בהמשך, בתוך סוגריים מסולסלות, ניתן לציין את ה-constraints ואת ה-properties של המתודה המתוארת.

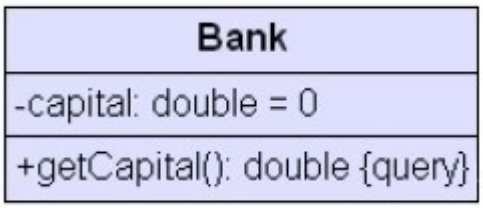
© 2009 Haim Michael. All Rights Reserved.

ה-constraints האפשריים כוללים preconditions ו-postconditions. ה-precondition מתאר תנאי שחייב להתקיים לפני שקוראים להפעלת המתודה המתוארת. ה-postcondition מתאר תנאי שחייב להתקיים לאחר שהמתודה מסיימת את פעולתה. הן את ה-preconditions והן את ה-postconditions כותבים כביטוי לוגי שערכו או true או false. את הביטוי הלוגי רושמים בתוך הסוגריים המסולסלות בצירוף המילה 'postcondition' או 'precondition' משמאל כשביניהם מפרידים נקודותיים ':'.

כך לדוגמא, בתרשים הבא מתוארת המתודה setCapital בצירוף precondition שמשמעותו היא שהערך של הפרמטר value חייב להיות חיובי כאשר יש קריאה להפעלתה של המתודה. המתודה getBranches מופיעה בצירוף ה-postcondition שאומר כי הערך של הפרמטר branches שבאמצעותו המתודה מחזירה את תשובתה חייב להיות שונה מ-null. המתודה setBranches מופיעה בצירוף ה-precondition שאומר כי כאשר יש קריאה להפעלתה הערך שנשלח באמצעות הפרמטר branches חייב להיות שונה מ-null.



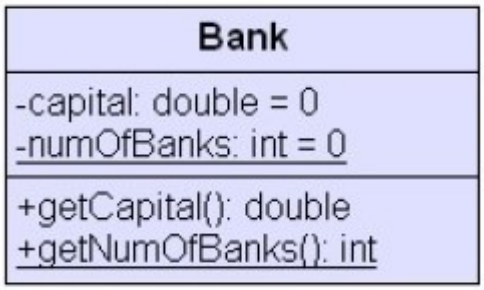
אחד ה-properties האפשריים הוא 'query' אשר מתאר מתודה שבפעולתה היא לא משנה אף ערך באובייקט וכל תפקידה לבדוק ולהחזיר ערך מבוקש (בדומה ל-query שמבצעים מול בסיס נתונים). בדוגמא הבאה, המתודה getCapital מסומנת באמצעות {query} כדי לתת ביטוי לכך שכאשר יש קריאה להפעלתה היא לא משנה שום ערך באובייקט שעליו היא פועלת.



© 2009 Haim Michael. All Rights Reserved.

מתודות סטטיות (Static Operations)

כדי לסמן מתודה כמתודה סטטית יש להוסיף קו תחתי מתחתיה. הדוגמא הבאה מציגה class אשר כולל בתוכו את ההגדרה המתודה הסטטית getNumOfBanks.

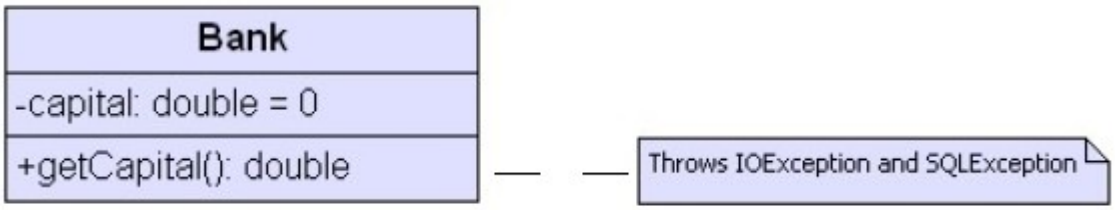


© 2009 Haim Michael. All Rights Reserved.

תיאור תקלות שעלולות להתרחש (Operations Exceptions)

כאשר בקריאה להפעלת מתודה יש סכנה שתתרחש תקלה (ייזרק exception) מקובל להוסיף note ובו תיאור ה-exception שעלול להיזרק ולחברו בקו מקווקו למתודה שבה מדובר.

הדוגמא הבאה מציגה תיאור של מחלקה שכוללת בתוכה את ההגדרה של מתודה שבקריאה להפעלתה עלולה להתרחש תקלה (עלול להיזרק exception) מסוג IOException או מסוג SQLException.

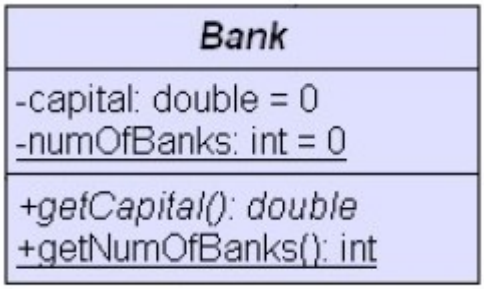


© 2009 Haim Michael. All Rights Reserved.

מחלקה אבסטרקטית (Abstract Class)

תיאור מחלקה אבסטרקטית נעשה באמצעות כתיבת שמה באותיות נטויות (italics). תיאור מתודה כאבסטרקטית נעשה באמצעות כתיבת שמה באותיות נטויות (italics).

הדוגמא הבאה מציגה תיאור של המחלקה האבסטרקטית Bank, אשר כוללת הגדרה של מתודה אבסטרקטית אחת בשם getCapital.



ניתן לשים לב לכך שגם השם של ה-class וגם השם של המתודה האבסטרקטית כתובים באמצעות גופן נטוי (italics).

© 2009 Haim Michael. All Rights Reserved.

קשר של Dependency בין Classes שונים

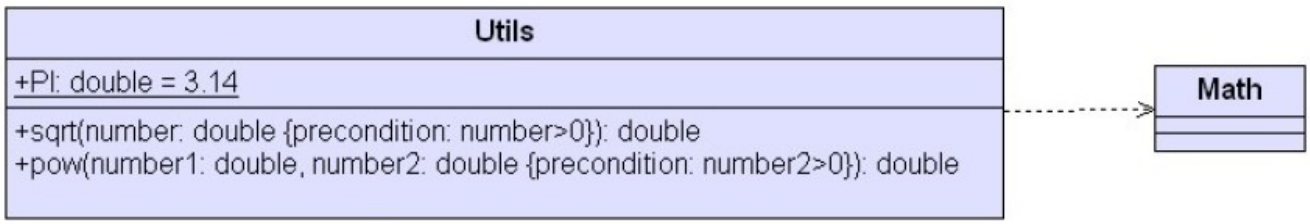
בין class נתון ל-class אחר קיים קשר של Dependency כאשר ה-class הנתון עושה שימוש ב-class האחר. קשר זה יכול לבוא לידי ביטוי במגוון דרכים שונות.

כך למשל, התלות יכולה למשל לבוא לידי ביטוי בכך שמתודה סטטית שמוגדרת ב-class הנתון קוראת להפעלתה של מתודה סטטית שמוגדרת ב-class האחר.

דוגמא אחרת למצב התלות המתואר יכולה להתרחש כאשר מתודה רגילה שמוגדרת ב-class הנתון, כאשר היא פועלת על אובייקט מסויים מטיפוס ה-class הנתון היא קוראת להפעלת מתודה שהוגדרה ב-class האחר אשר יכולה להיות הן מתודה סטטית והן מתודה רגילה.

קשר של תלות מקובל לתאר באמצעות קו חץ מקוקו אשר יוצא מה-class הנתון ומצביע אל ה-class האחר.

הדוגמא הבאה מציגה קשר של Dependency שקיים בין ה-class הנתון ששמו Utils ל-class האחר ששמו Math.



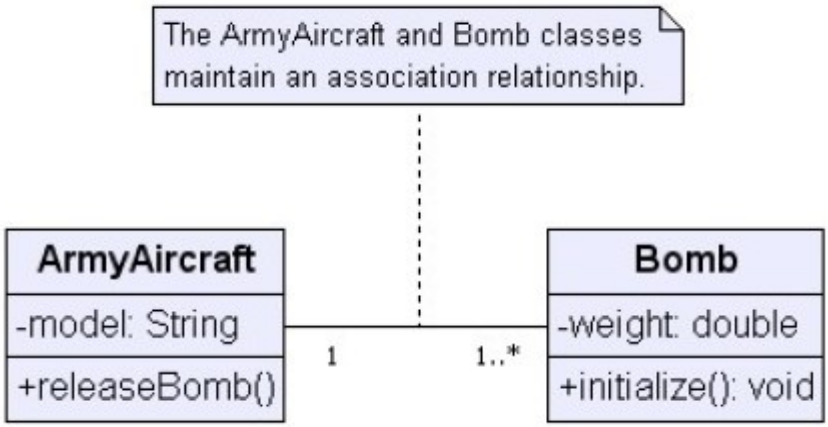
כאשר קיים קשר של Dependency בין class נתון ל-class אחר מקובל לומר שה-class הנתון 'uses a' את ה-class האחר.

© 2009 Haim Michael. All Rights Reserved.

קשר של Association בין Classes שונים

כאשר בין שני classes שונים מתקיים קשר של Association המשמעות היא שקיים קשר בין אובייקטים שנוצרים מכל אחד משני ה-classes באופן שבו אחד מהם (לפחות) מחזיק ב-reference של האובייקט מה-class האחר למשך פרק זמן כלשהו ובנוסף, משך חייו של האובייקט האחד שונה ממשך חייו של האובייקט האחר.

קשר של Association בין שני classes מתואר באמצעות קו רציף רגיל. ניתן להוסיף note אשר יהיה מחובר בקו מקווקו לקו הרציף שמתאר את קשר ה-Association ולהוסיף בתוכו הסבר נוסף בנוגע לקשר.



הדוגמא הבאה מציגה קשר של Association בין ה-class ששמו ArmyAircraft ל-class ששמו Bomb. קשר ה-Association בין השניים מתואר באמצעות note אשר כולל הסבר נוסף.

ניתן להוסיף בכל אחד מקצותיו של קשר ה- Association את ערך ה-multiplicity שמתקיים בקשר ובכך לתת ביטוי למספר האובייקטים מכל אחד מה-classes אשר משתתפים בקשר.

כך למשל, בדוגמא האחרונה ניתן להבין שבקשר ה-Association שמתקיים בין ה-class ששמו ArmyAircraft וה-class ששמו Bomb משתתף אובייקט אחד מטיפוס ArmyAircraft כנגד אובייקט אחד או יותר מטיפוס Bomb, ובמילים אחרות, ניתן להבין שבמציאות המתוארת כל מטוס נושא פצצה אחת (לפחות) או יותר.

בדוגמא הבאה ניתן להבין שלכל אובייקט מטיפוס Car מחוברים בין 4 (כולל) ל-10 (כולל) אובייקטים מטיפוס CarWheel, ובמילים אחרות, ניתן להבין שהתרשים מתאר מציאות שבה לכל מכונית יש בין 4 ל-10 גלגלים.

© 2009 Haim Michael. All Rights Reserved.

האפשרויות הקיימות בעת מתן ביטוי ל-multiplicity הן:

0..1

כאשר בקשר משתתף אובייקט אחד לכל היותר וייתכן גם מקרה שלא יהיה אף אובייקט מהטיפוס שאליו ה-multiplicity מתייחס.

1

כאשר בקשר משתתף אובייקט אחד בדיוק. זהו גם ערך ברירת המחדל שמתקיים כאשר ה-Multiplicity לא מצויין.

0..*

כאשר בקשר משתתפים מספר לא ידוע של אובייקטים וייתכן אף מצב שבקשר לא משתתף אף אובייקט מה-class שאליו ה-multiplicity האמור מתייחס.

1..*

כאשר בקשר משתתפים אובייקט אחד או יותר מה-class שאליו ה-multiplicity האמור מתייחס.

n

ניתן גם לציין מספר סופי של אובייקטים שמשתתפים בקשר.

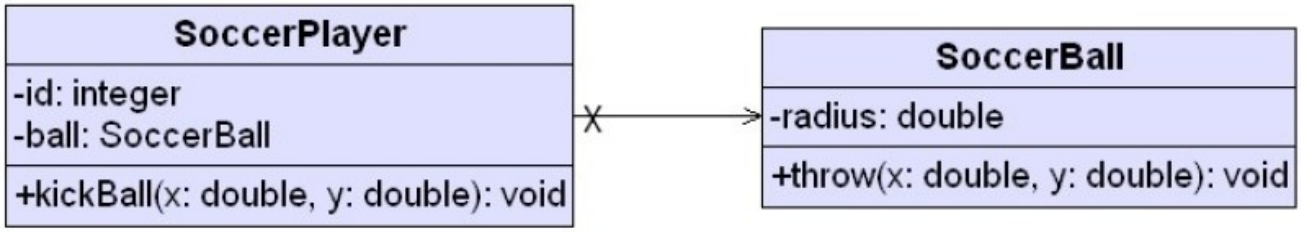
n..m

ניתן לציין טווח ברור וסופי של אובייקטים אשר משתתפים בקשר. כך למשל, ניתן לציין שבקשר משתתפים מספר כלשהו של אובייקטים בטווח n (כולל) עד m (כולל).

ניתן להוסיף ראש חץ בכל אחד משני קצותיו כדי לתת ביטוי לכיוון הקשר (ובמילים אחרות, לתת ביטוי לכך שאובייקט מסויים מבין השניים הוא זה שמחזיק ב-reference של האובייקט האחר). ניתן גם להוסיף X באחד משני צידי הקו כדי לתת ביטוי לכך שהקשר לא מתקיים בכיוון מסויים, וליתר דיוק, לתת ביטוי לכך שאובייקט שבצד האחר לא מחזיק ב-reference לאובייקט/ים מה-class שבצידו מופיע ה-X. בדוגמא הבאה ניתן לראות קשר שיוצא מה-class ששמו SoccerPlayer לכיוון ה-class ששמו SoccerBall. החץ שמופיע בכיוון של ה-class ששמו SoccerBall מעיד על כך שאובייקט שנוצר מה-class ששמו SoccerPlayer מחזיק ב-reference לאובייקט מסוג SoccerBall. העובדה שבצד

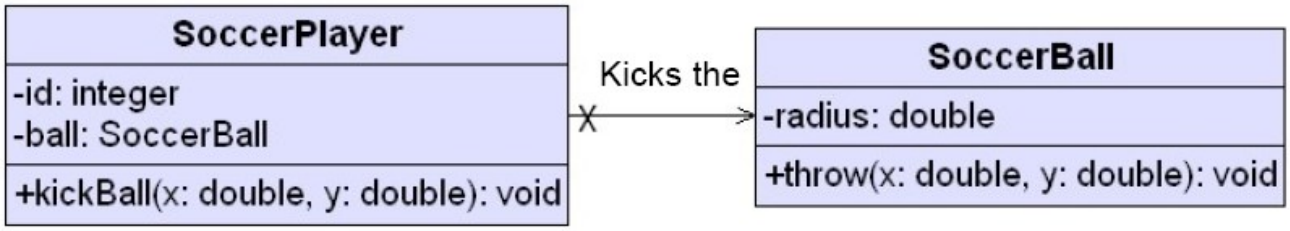
© 2009 Haim Michael. All Rights Reserved.

של ה-class ששמו SoccerPlayer יש סימן X מעידה על כך שאובייקט מטיפוס SoccerBall לא מחזיק ב-reference לאובייקט מטיפוס SoccerPlayer.



כאשר קיים קשר של Association בין class מסויים ל-class אחר נהוג לומר שה-class המסויים 'has a' את ה-class האחר.

בצמידות לקו שמתאר את קשר ה-Association שמתקיים בין שני classes שונים ניתן להוסיף טקסט נוסף אשר מתאר את הקשר. בדוגמא הבאה הטקסט 'Kicks The' מהווה תיאור נוסף לקשר בכך שהוא בא לתאר את העובדה שכל אובייקט מטיפוס SockerPlayer ניתן לומר עליו שהוא kicks the אובייקט מטיפוס SockerBall.



© 2009 Haim Michael. All Rights Reserved.

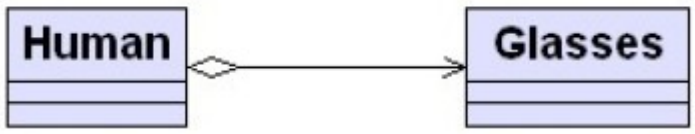
קשר של Aggregation בין Classes שונים

קשר של Aggregation הוא למעשה קשר של Association בעוצמה יותר חזקה. ההבדל לעומת קשר של Association הוא שבין שני classes שמתקיים ביניהם קשר של Aggregation קיים קשר של בעלות בין האובייקטים שנוצרים מכל אחד מה-classes במובן שקיים קשר בין משך החיים של אובייקטים מה-class האחד למשך החיים של אובייקטים מה-class האחר.

כאשר בין שני classes מתקיים קשר של Aggregation נהוג לומר כי בין השניים מתקיים קשר מסוג "owns a".

מקובל לתאר קשר של Aggregation באמצעות קו חץ בצירוף יהלום ריק אשר יוצא מה-class שאובייקטים שנוצרים ממנו מקיימים קשר של בעלות ביחס לאובייקטים שנוצרים מה-class האחר. היהלום הריק מופיע בצד ה-class שאובייקטים אשר נוצרים ממנו מקיימים קשר של בעלות ביחס לאובייקטים מה-class האחר.

בדוגמא הבאה מתקיים קשר של Aggregation בין ה-class ששמו Human ל-class ששמו Glasses במובן שכל אובייקט שנוצר מה-class ששמו Human מחזיק ב-reference לאובייקט שנוצר מה-class ששמו Glasses. כל אובייקט מטיפוס Human מחזיק בבעלותו reference לאובייקט שנוצר מטיפוס Glasses. התרשים מתאר מציאות שבה לכל אדם יש זוג משקפיים שבהם הוא משתמש בפרק זמן ארוך עד כי כמעט שמתקיימת זהות בין משך החיים של האדם ומשך החיים של המשקפיים.



© 2009 Haim Michael. All Rights Reserved.

קשר של Composition בין Classes שונים

קשר מסוג Composition הוא למעשה קשר מסוג Aggregation בעוצמה יותר חזקה. קשר מסוג Composition מתואר בדומה לקשר מסוג Aggregation בהבדל אחד. במקום להשתמש ביהלום ריק משתמשים ביהלום מלא.

כאשר בין שני classes מתקיים קשר מסוג Composition המשמעות היא שאובייקטים שנוצרים מה-class האחד מקיימים קשר של בעלות ביחס לאובייקטים שנוצרים מה-class האחר באופן שבו מתקיימת כמעט זהות מלאה בין משך החיים של האובייקט/ים מה-class הראשון ומשך החיים של האובייקט/ים מה-class השני. בקשר מסוג Aggregation לא קיימת זהות כל חזקה בין משכי החיים של שני האובייקטים.

כאשר בין שני classes מתקיים קשר מסוג Composition האובייקט שנמצא בבעלותו של האובייקט האחר לא יכול להיות בו זמנית גם בבעלותו של אובייקט נוסף. בקשר מסוג Aggregation אובייקט שנמצא בבעלותו של אובייקט אחד יכול להיות באותה עת גם בבעלותו של אובייקט נוסף.

כאשר בין שני classes מתקיים קשר מסוג Composition נהוג לומר כי בין שני ה-classes מתקיים קשר מסוג "is part of". בדוגמא הבאה מתואר קשר שבו כל אובייקט מטיפוס Heart מהווה למעשה חלק מאובייקט מטיפוס Human. התרשים מתאר מציאות שבה לכל אדם יש לב ועל פי רוב משך החיים של האדם זהה כמעט לחלוטין למשך החיים של הלב שלו.

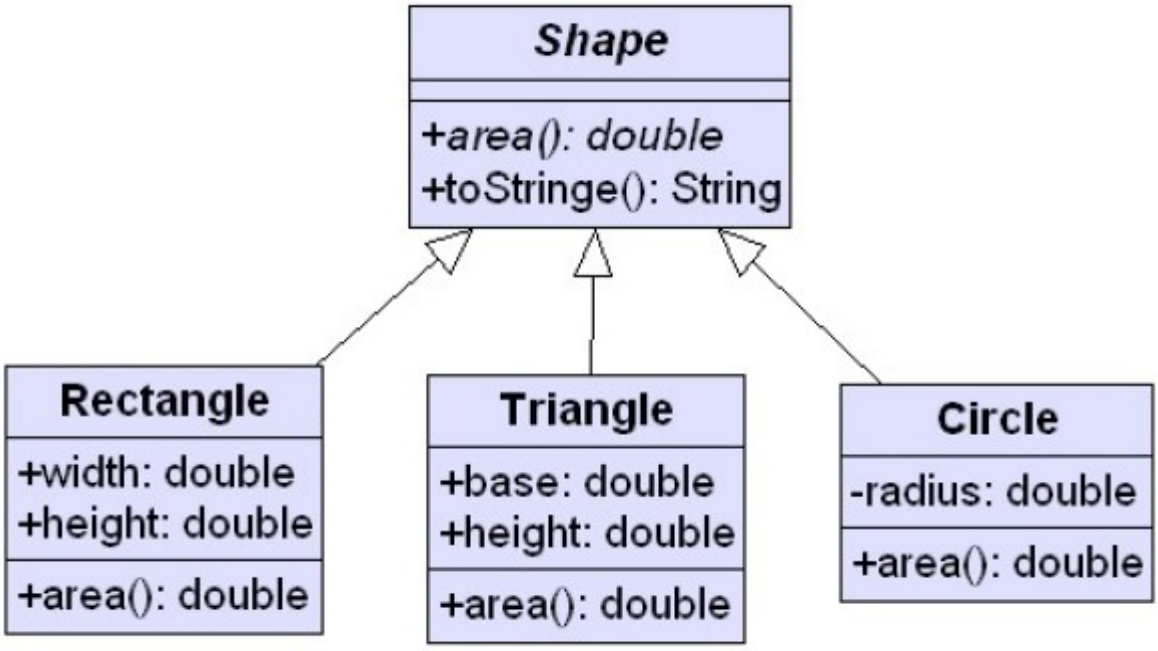
© 2009 Haim Michael. All Rights Reserved.

תיאור Class שיורש מ-Class אחר

כאשר בין שני classes מתקיים קשר של הורשה (Generalization) נהוג לומר שה-class שיורש מה-class האחר מקיים קשר של "is a" באופן שבו ניתן לומר שכל אובייקט מה-class היורש הוא גם מטיפוס ה-class המוריש.

נהוג לתאר קשר של הורשה בין שני classes באמצעות קו רציף וראש חץ ריק וסגור אשר יוצא מה-class היורש ומכוון אל ה-class המוריש.

הדוגמא הבאה מציגה שלושה classes אשר יורשים מה-class ששמו Shape.

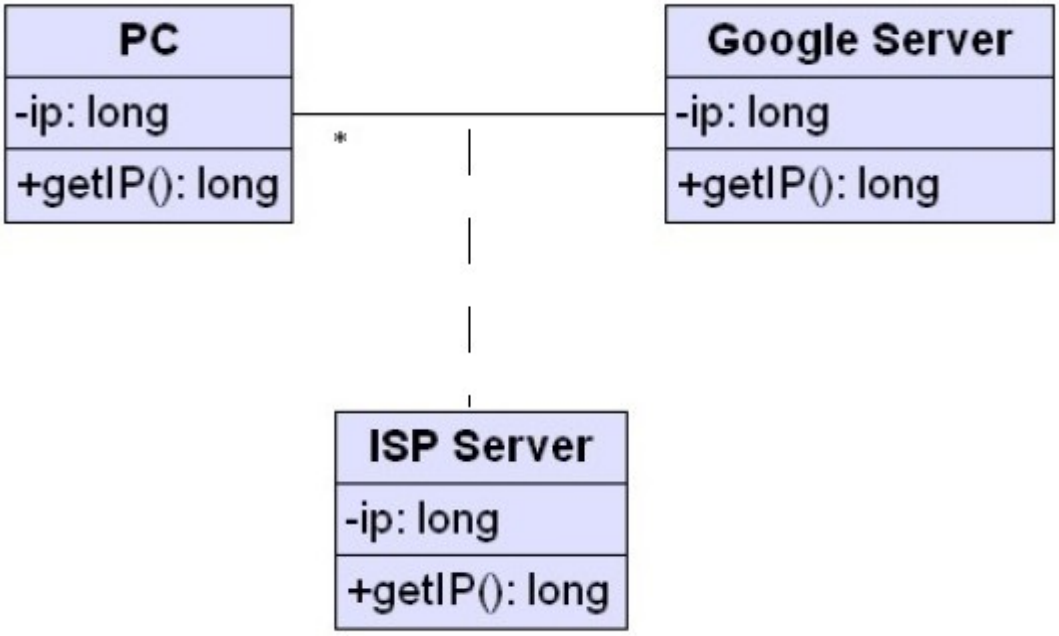


© 2009 Haim Michael. All Rights Reserved.

תיאור קשר בין Classes באמצעות Association Class

כאשר בין שני classes שונים קיים קשר של Association ניתן להציג תיאור נוסף של הקשר האמור באמצעות ציור של class נוסף אשר מחובר באמצעות קו מקווקו לקו הרציף שמתאר את קשר ה-Association האמור. ה-class הנוסף יכול לספק תיאור נוסף לקשר ה-Association האמור.

בדוגמא הבאה נעשה שימוש ב-class ששמו ISPServer כדי לתאר את קשר ה-Association שמתקיים בין PC ו-GoogleServer.



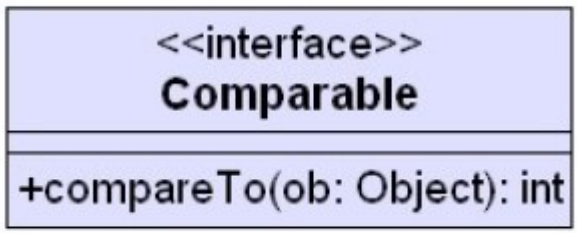
© 2009 Haim Michael. All Rights Reserved.

תיאור Interfaces ו-Classes שמיישמים אותם

כדי לתאר interface ניתן להשתמש בסימול הגראפי המקובל לתיאור class בצירוף ה-stereotype:

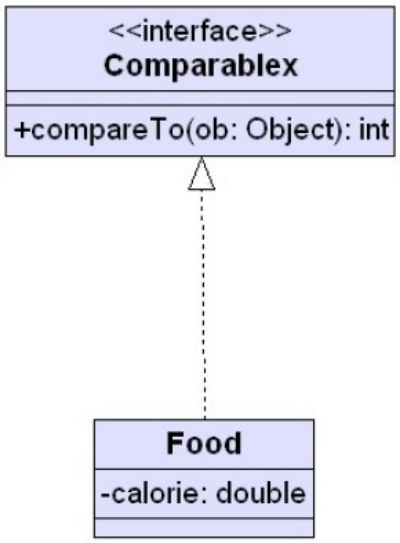
<<interface>>

בדוגמא הבאה מתואר ה-interface ששמו Comparable:



אין צורך לכתוב את שם ה-interface באמצעות אותיות בכתב italic. כמו כן, גם אין צורך לכתוב את כל אחת מהמתודות בכתב italic. היותן של המתודות אבסטרקטיות ברור מעצם הגדרת בתוך ה- interface.

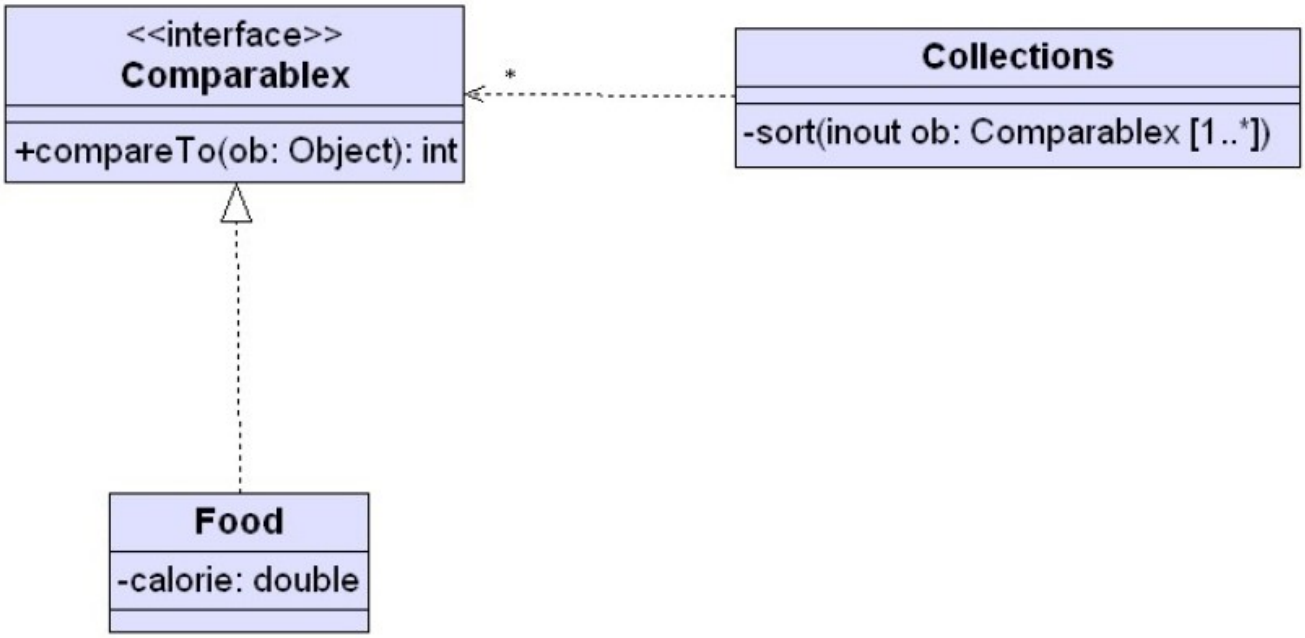
כדי לתאר class אשר מיישם interface יש לצייר קו מקווקו שיוצא מה-class אל ה-interface ובקצהו ראש חץ סגור וריק. הדוגמא הבאה מתארת את ה-class ששמו Food כ-class אשר מיישם את ה-interface ששמו Comparablex.



© 2009 Haim Michael. All Rights Reserved.

את כל אחד מסוגי הקשרים (Dependency, Association, Aggregation ו- Composition) ניתן גם לתאר כקשר בין Class ל-Interface או אפילו כקשר בין שני Interfaces שונים.

הדוגמא הבאה מציגה זאת.

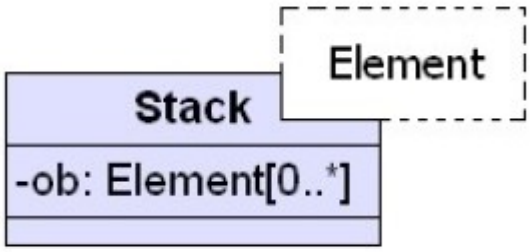


© 2009 Haim Michael. All Rights Reserved.

תיאור Classes שפועלים כ-Templates

כאשר רוצים לתאר class אשר כולל בהגדרתו parametric type אשר נקבע בכל עת שיוצרים אובייקט מה-class ניתן לתת לכך ביטוי באמצעות ציור מלבן מקווקו בפינתו הימנית העליונה של המלבן שמתאר את ה-class וציון שמו של ה-parametric type ב-class שמוגדר. בדוגמא הבאה מצויין הפרמטר Element בתור ה-parametric type שנעשה בו שימוש בהגדרה של Stack.

בכל עת שבו נוצר אובייקט מה-class ששמו Stack יצויין ה-type שבהגדרת ה-class מופיע באמצעות הפרמטר ששמו Element.



© 2009 Haim Michael. All Rights Reserved.