דיאגרמה לתיאור Packages

הקדמה לדיאגרמת packages

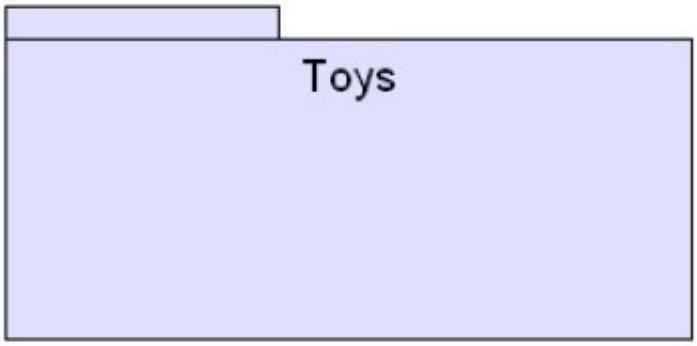
באמצעות דיאגרמת packages ניתן לתת ביטוי לקבוצות שונות של אלמנטים במערכת שלנו. דיאגרמה זו תשמש אותנו בדרך כלל למתן ביטוי לקבוצות השונות של ה-classes שאנו מגדירים (קבוצות אשר ידועות ב-Java בשם packages וב-C# בשם name spaces).

בעצם הגדרתו של package כל פניה לכל אחד מהאלמנטים ששייכים לו צריכה להיעשות תוך שימוש בשם של ה-package בצירוף השם של האלמנט (בדומה לאופן הגישה ל-classes שמוגדרות ב-Java כ-classes אשר שייכים ל-package מסויים ומסיבה זו יש לגשת אליהם בצירוף שם ה-package באופן שידוע כ-full qualified name).

תיאור גראפי של package

כל package יתואר באמצעות מלבן שלצד שמאל של חלקו העליון מוצמד טאב שבתוכו רשום השם של ה-package.
לחילופין, ניתן גם לרשום את השם של ה-package בתוך המלבן עצמו.

בדוגמא הבאה יש תיאור של ה-package ששמו Toys.

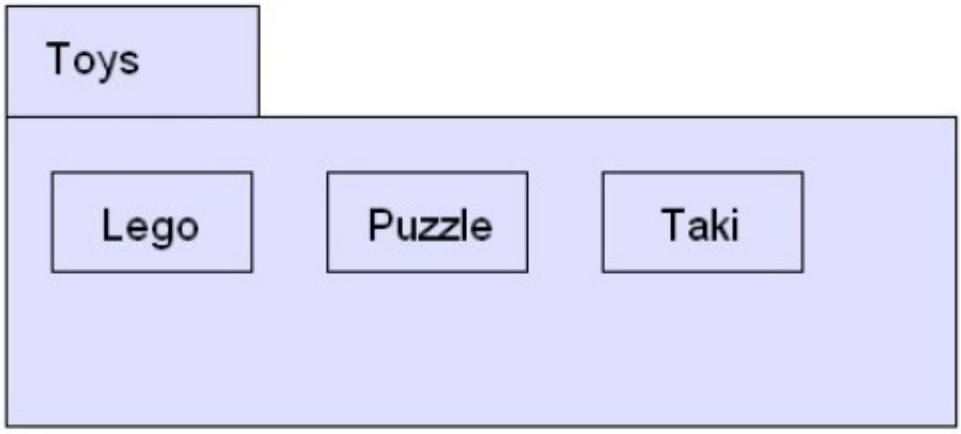


בתוך המלבן ניתן לצייר את האלמנטים שה-package כולל בתוכו. בדוגמא הבאה ניתן לראות שה-package ששמו Toys כולל בתוכו את ה-classes הבאים:

Lego

Puzzle

Taki



הרשאות הגישה של אלמנטים שכלולים ב-package

כל אחד מהאלמנטים שמשוייכים ל-package יכול להיות עם אחת משתי הרשאות הגישה הבאות:

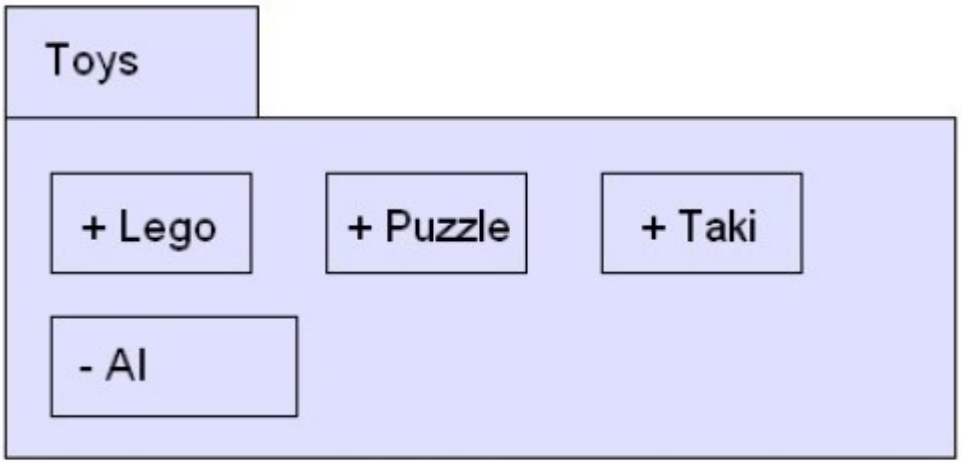
private

או

public

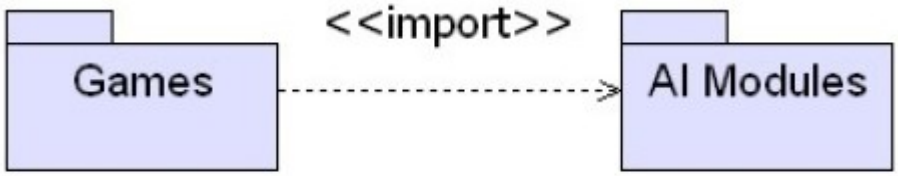
אלמנט עם הרשאת הגישה private יהיה נגיש אך ורק לאלמנטים ששייכים לאותו package. כדי לסמן אלמנט כאלמנט עם הרשאת הגישה private יש להוסיף את הסימן '-' משמאל לשמו. אלמנט עם הרשאת הגישה public יהיה נגיש לכולם מכל מקום. כדי לסמן אלמנט כאלמנט עם הרשאת הגישה public יש להוסיף את הסימן '+' משמאל לשמו.

בדיאגרמה הבאה מוצג ה-package ששמו Toys ובתוכו ארבעה אלמנטים (classes). ה-class ששמו AI מופיע עם הרשאת הגישה private ומסיבה זו הוא נגיש רק ל-classes האחרים שמופיעים בדיאגרמה בתוך ה-package ששמו Toys. ה-classes האחרים יהיו נגישים לכל class מכל package כיוון שהם מוגדרים עם הרשאת הגישה public.



ביצוע import ל-package

כדי להימנע מכתיבת ה-full qualified name של כל אלמנט ששייך ל-package אחר ושרוצים להשתמש בו ניתן לבצע את הפעולה import ל-package האחר. פעולת import ל-package אחר מתוארת באמצעות קו חץ מקווקו אשר מצביע על ה-package שרוצים לעשות לו import. כמו כן, יש לרשום מעל הקו האמור <<import>>.



יש לשים לב לכך שפעולת ה-import לא נוגעת לאלמנטים עם הרשאת הגישה private. היא נוגעת לאלמנטים עם הרשאת הגישה public בלבד. אלמנטים עם הרשאת הגישה private במילא לא נגישים מחוץ ל-package שאליו הם שייכים. פעולת ה-import נועדה אך ורק להקל עלינו בכך שהיא מאפשרת לנו לגשת לכל אחד מהאלמנטים שמיובאים מבלי שיהיה צורך לרשום את ה-full qualified name של כל אחד מהם. פעולת ה-import לא נוגעת לאלמנטים עם הרשאת הגישה private שממשיכים להיות בלתי נגישים מחוץ לגבולות ה-package שאליו הם שייכים.

ביצוע access ל-package

כתוצאה מביצוע פעולה של import ל-package מסויים (בדוגמא האחרונה: AI Modules) כל האלמנטים ששייכים ל-package המיובא (AI Modules בדוגמא האחרונה) נחשבים כעת לאלמנטים עם הרשאת הגישה public אשר שייכים ל-package המייבא (בדוגמא האחרונה: Games). מסיבה זו, אם קיים package שלישי אשר מבצע import ל-package המייבא האחרון (בדוגמא האחרונה: Games) אז כל האלמנטים שיובאו מה-package הראשון יהיו נגישים ללא צורך בכתיבת ה-full qualified name גם מתוך ה-package השלישי.

בדוגמא הבאה, כל האלמנטים שמיובאים מ-AI Modules ל-Games מיובאים כעת ל-Mobile Games ולפיכך הם נגישים ללא צורך בכתיבת ה-full qualified name גם על ידי Mobile Games וגם על ידי Games.

לחילופין, במקום לבצע import ניתן לבצע access. בפעולה access אשר package מסויים מבצע ל-package אחר כל האלמנטים עם הרשאת הגישה public ששייכים ל-package האחר נגישים ללא צורך בכתיבת ה-full qualified name גם על ידי ה-package המסויים. עם זאת, להבדיל מביצוע של הפעולה import כאשר מבצעים את הפעולה access כל האלמנטים המיובאים מה-package האחר נחשבים עבור ה-package המסויים כאלמנטים עם הרשאת הגישה private ומסיבה זו, אם package שלישי מבצע כעת import ל-package המסויים שזה עתה ביצע access ל-package הראשון אז האלמנטים ששייכים ל-package הראשון לא יהיו נגישים ל-package השלישי.

את הפעולה access מתארים באמצעות קו חץ מקוקו שעליו רושמים <<access>>.

בדוגמא הבאה, ה-package ששמו Games מבצע access ל-package ששמו AI Modules. מסיבה זו, כל האלמנטים (עם הרשאת הגישה public בלבד) ששייכים ל-package ששמו AI Modules נגישים כעת מתוך Games ללא צורך בכתיבת ה-full qualified name. עם זאת, להבדיל ממה שקורה כאשר מבצעים import האלמנטים שיובאו נחשבים כעת כאלמנטים ששייכים ל-Games עם הרשאת הגישה private ומסיבה זו, ה-package השלישי בדוגמא (ה-package ששמו Mobile Games) שכעת מבצע import ל-package ששמו Games - כל האלמנטים שבמקור שייכים ל-AI Modules לא ייובאו אליו.

ביצעו merges בין שני packages

כאשר package נתון מבצע merge עם package אחר אז כל אחד מהאלמנטים ששייכים ל-package הראשון שקיים עבורו אלמנט עם שם זהה ב-package השני מתמזג עם האלמנט בשם הזהה ששייך ל-package האחר באופן שבו נוצר אלמנט חדש שמתקבל מביצוע generalization של האלמנט מה-package היוזם ביחס לאלמנט התואם מה-package האחר.

אלמנטים עם הרשאת הגישה private לא משתתפים בתהליך המיזוג.

אם שני ה-packages כוללים sub packages עם שם זהה אז תהליך המיזוג נמשך באופן רקורסיבי בין האלמנטים ששייכים לכל אחד מה-sub packages.

האלמנטים המקוריים בכל אחד מה-packages עדיין נגישים כאשר משתמשים ב-full qualified name.

אלמנטים ו-sub packages שקיימים רק באחד משני ה-packages נותרים ללא שינוי ב-package החדש שמתקבל.

כל פעולה של import מ-package ש-package אחר ביצע עליו פעולה של merge תהפוך לפעולה של import מה-package החדש שמתקבל לאחר שה-merge בוצע.

את הפעולה merge מתארים באמצעות חץ מקוקו שיוצא מה-package שיוזם את ה-merge אל ה-package האחר. בצמוד לחץ המקווקו יש לרשום <<merge>>.

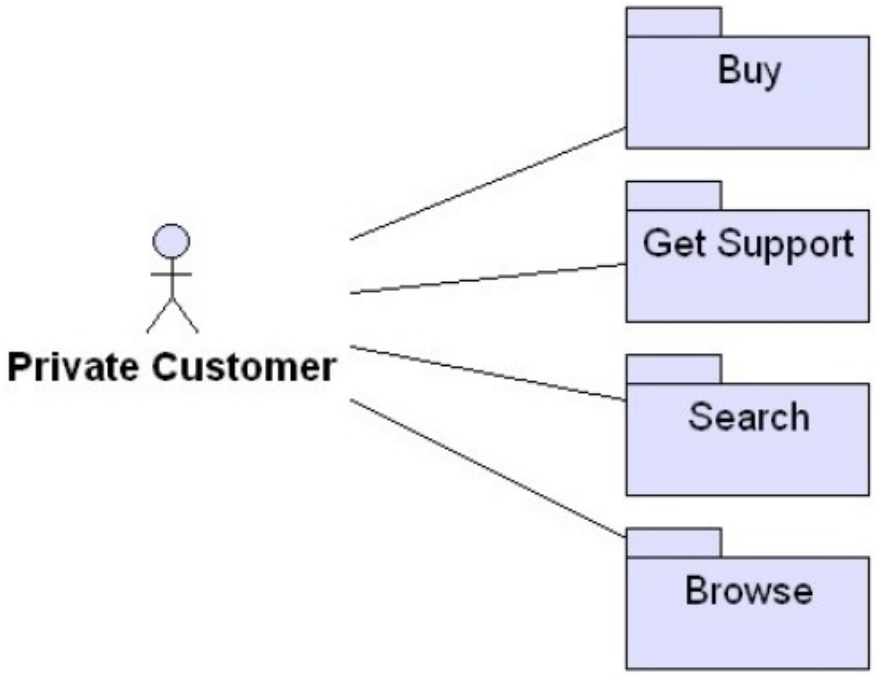
הדוגמא הבאה מציגה פעולה של merge שה-package ששמו Mobile Games יוזם ביחס ל-package ששמו Games.

שילוב דיאגרמת use case ודיאגרמת packages

ניתן לשלב בין דיאגרמת use case ודיאגרמת packages באופן שבו האליפסה אשר מייצגת use case מסויים מוחלפת בסימון הגרפי המקובל ל-package. בדרך זו, ניתן לתת ביטוי ל-packages שמכילים את ה-classes שמהווים את המימוש לכל אחד מה-use cases שקיימים.

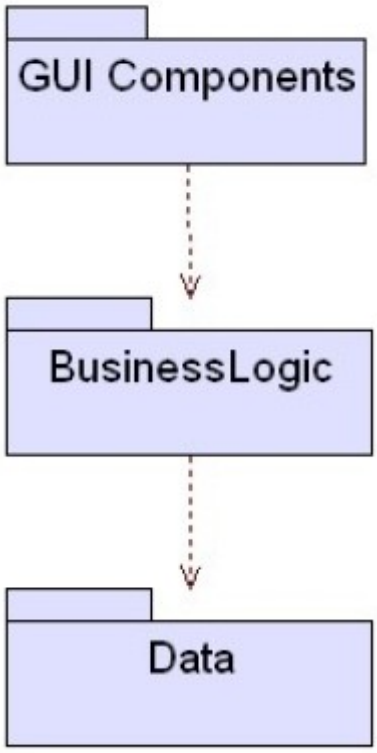
בדוגמא הבאה ניתן לראות שה-package ששמו Maintenance כולל למעשה את ה-classes שמהווים מימוש של ה-use cases השונים שבהם מעורב ה-Administrator.

בדוגמא הבאה ניתן לראות את שמות ה-packages אשר כוללים את ה-classes שמהווים את המימוש של כל אחד מה-use cases השונים שבהם מעורב ה-Actor ששמו Private Customer.



קשר של תלות בין packages

את התלות שקיימת בין package מסויים באחר ניתן לבטא באמצעות קו חץ מקווקו אשר יוצא מה-package המסויים ומצביע אל כיוון ה-package האחר.



בדוגמא זו ניתן לראות שה-package ששמו GUI Components תלוי ב-packages ששמו Business Logic אשר תלוי ב-package ששמו Data.

תרשים התלויות בין ה-packages השונים מתווה, בדר"כ, את תהליך הפיתוח באופן שבו מתחילים בפיתוח של ה-packages שבהם ה-packages האחרים תלויים.