מבוא לתיכנות מונחה עצמים – דגמי-עיצוב (design patterns)

מקורות:

Marty Stepp, M. Ernst, S. Reges, D. Notkin, R. Mercer, קרן כליף

דגמי-עיצוב – למה צריך את זה?

- כי יש בעיות בעיצוב מערכות תוכנה, שחוזרות על עצמן שוב ושוב במערכות שונות.
- לבעיות האלו ישנן פתרונות מוכרים. כשאנחנו משתמשים בהם, קל יותר למתכנתים אחרים להבין את הפתרון שלנו.

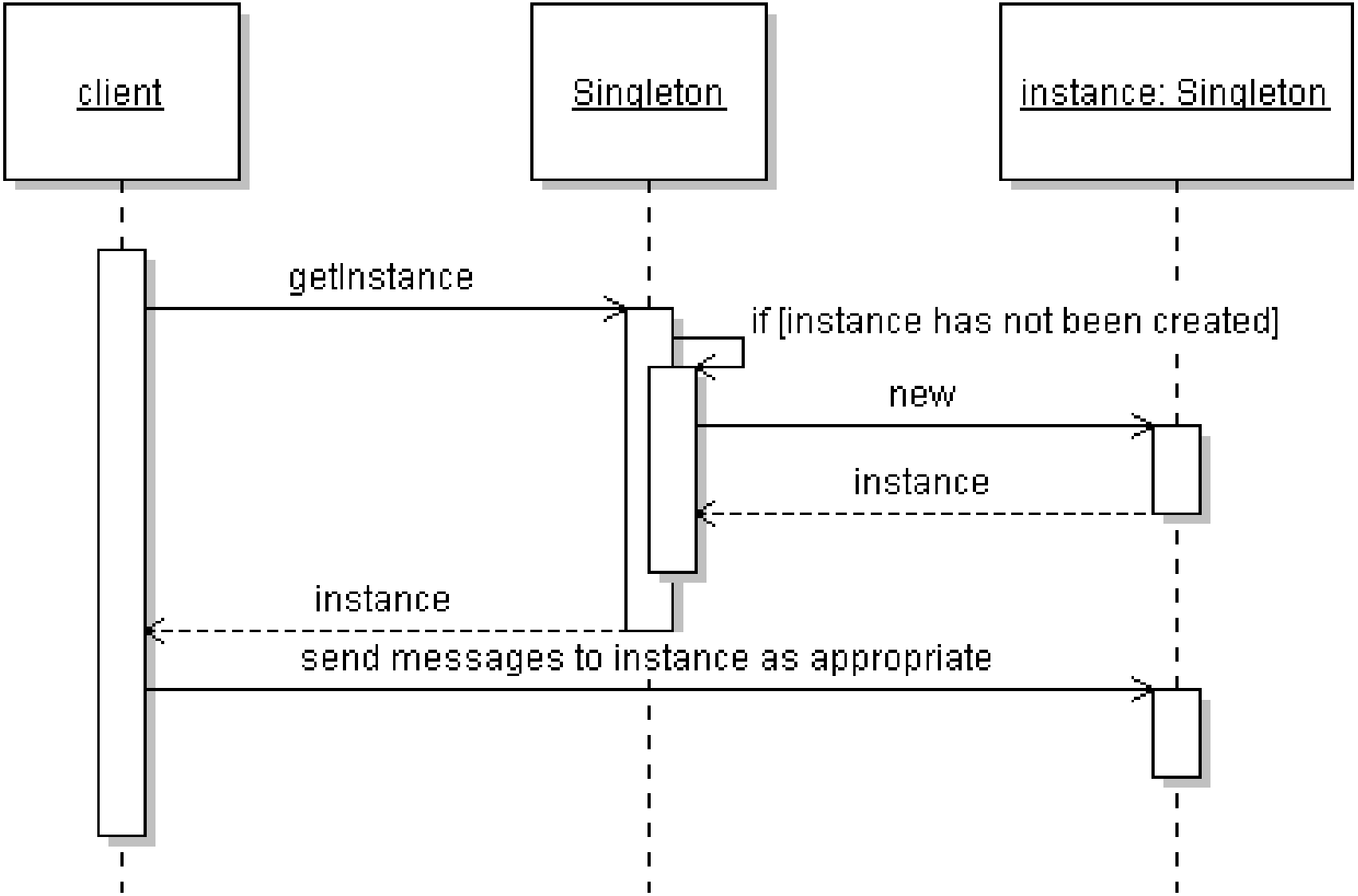
דגמי-עיצוב מקובלים

- **דגמי יצירה – creational patterns:**
  - **Factory Method** **Abstract Factory** **Singleton**
  - Builder **Prototype**
- **דגמי מבנה – structural patterns:**
  - **Adapter** Bridge **Composite**
  - Decorator Facade **Flyweight**
  - **Proxy**
- **דגמי התנהגות – behavioral patterns:**
  - Command Interpreter Iterator
  - Mediator **Observer** State
  - Strategy Chain-of-Responsibility Visitor
  - **Template**

סינגלטון - Singleton

- **המטרה**: שיהיה רק עצם אחד מסוג מסויים במערכת, אבל שיהיה אפשר לגשת אליו מכל מקום במערכת.
  - **לדוגמה**: מתרגם עברית-אנגלית, מערכת קבצים, מערכת משתמשים, מאגר-נתונים, Comparator
- **פתרון אפשרי**: סינגלטון: בנאי פרטי, שדה סופי.

סינגלטון – איתחול "עצל"



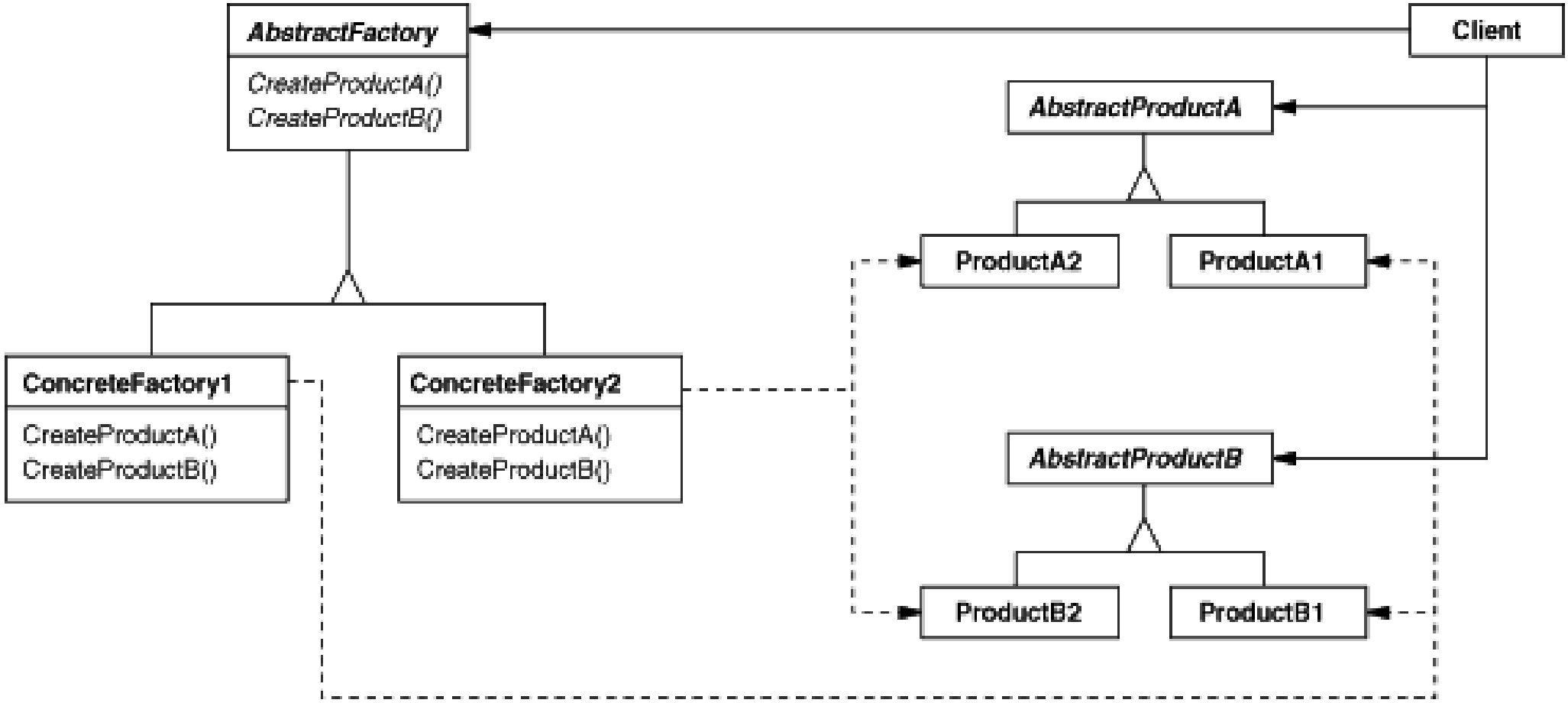
מתודות מפעל – Factory methods

- **המטרה**: לקבוע את סוג העצם לפי פרמטרים שונים.
  אי-אפשר להשתמש ב-new כי new חייב לקבל את הסוג.
- **פתרון אפשרי**: מתודות "מפעל" – קוראות לבנאי המתאים עם הפרמטרים המתאימים.
  - דוגמה: https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/ThreadPoolExecutor.html
  - לעומת: https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/Executors.html#newCachedThreadPool--

מפעל אבסטרקטי – Abstract factory

- **המטרה**: לייצר משפחה של עצמים מאותו סוג, כאשר הסוג נקבע תוך כדי ריצה.
- **פתרון אפשרי**: בונים ממשק של "מפעל", עם כמה מימושים שונים.

מפעל אבסטרקטי – תרשים מחלקות



אב-טיפוס – Prototype

- **המטרה**: ליצור עצם עם פרמטרים נתונים ע"י המשתמש, תוך שינוי מספר קטן של פרמטרים.
  - דוגמא: העתק/הדבק בתוכנת ציור
- **פתרון אפשרי**: מתודת clone לכל העצמים.
  - הערה: יש clone בג'אבה, אבל רוב המומחים ממליצים לא להשתמש בו אלא לכתוב חדש.

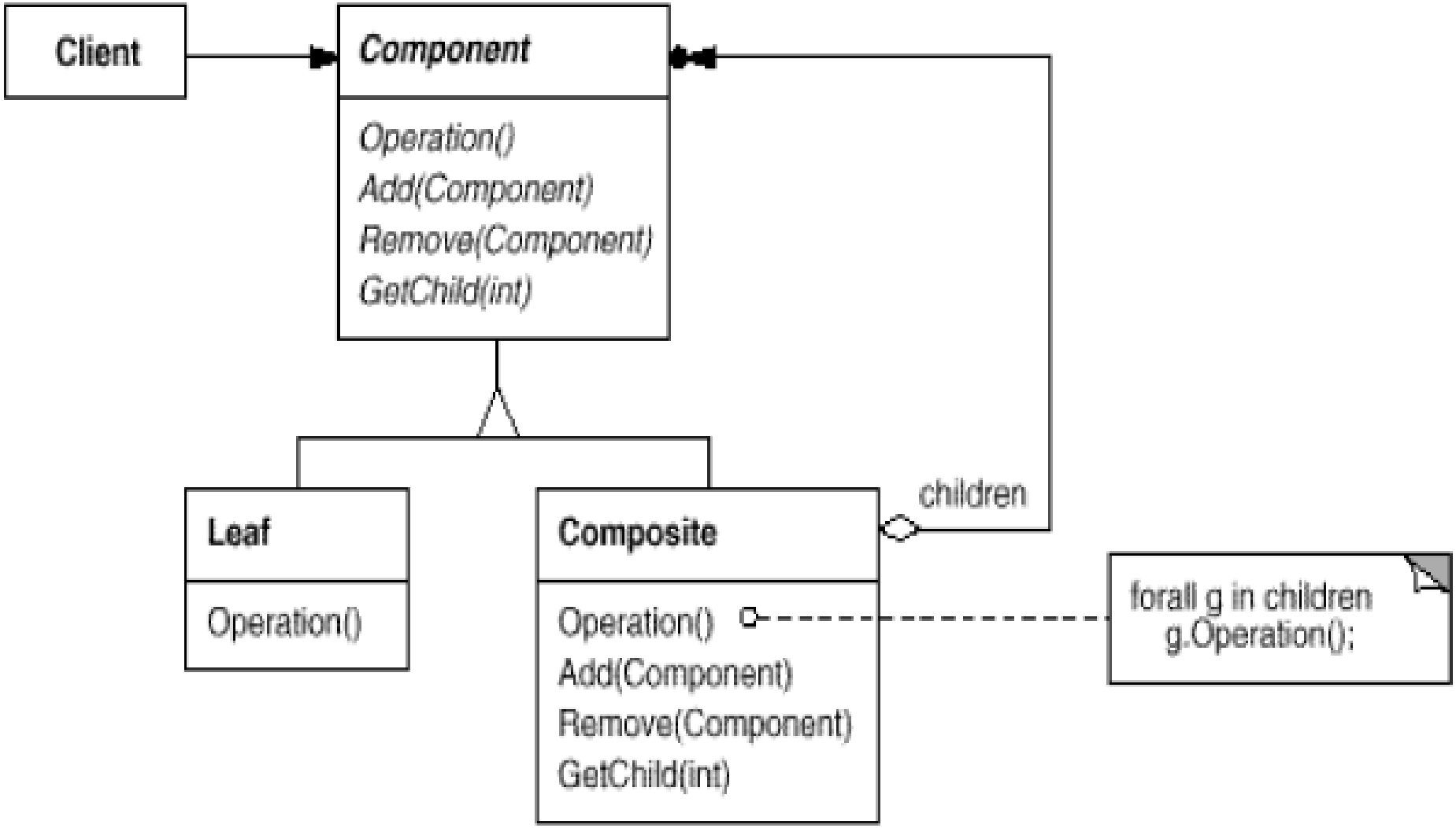
Flyweight

- **המטרה**: למנוע יצירת עצמים מיותרים, כאשר הזהות של עצם תלויה רק בפרמטרי האיתחול.
  - דוגמה: תאריכים, מחרוזות, קבצים
- **פתרון אפשרי**: שמירת עצמים במפה סטטית.
  שימוש במתודות-מפעל כדי ליצור עצם חדש או להחזיר עצם קיים לפי הצורך.

הרכב – Composite

- **המטרה**: ליצור עץ של עצמים, ולאפשר למשתמשים להתייחס ל"עלה" ול"צומת פנימי" באותה צורה.
  - דוגמא: קובץ xml, תיקיה במחשב, עץ קטגוריות.
- **פתרון אפשרי**: מגדירים מחלקה מופשטת "צומת בעץ" עם שתי מחלקות יורשות "צומת פנימי" ו"עלה".

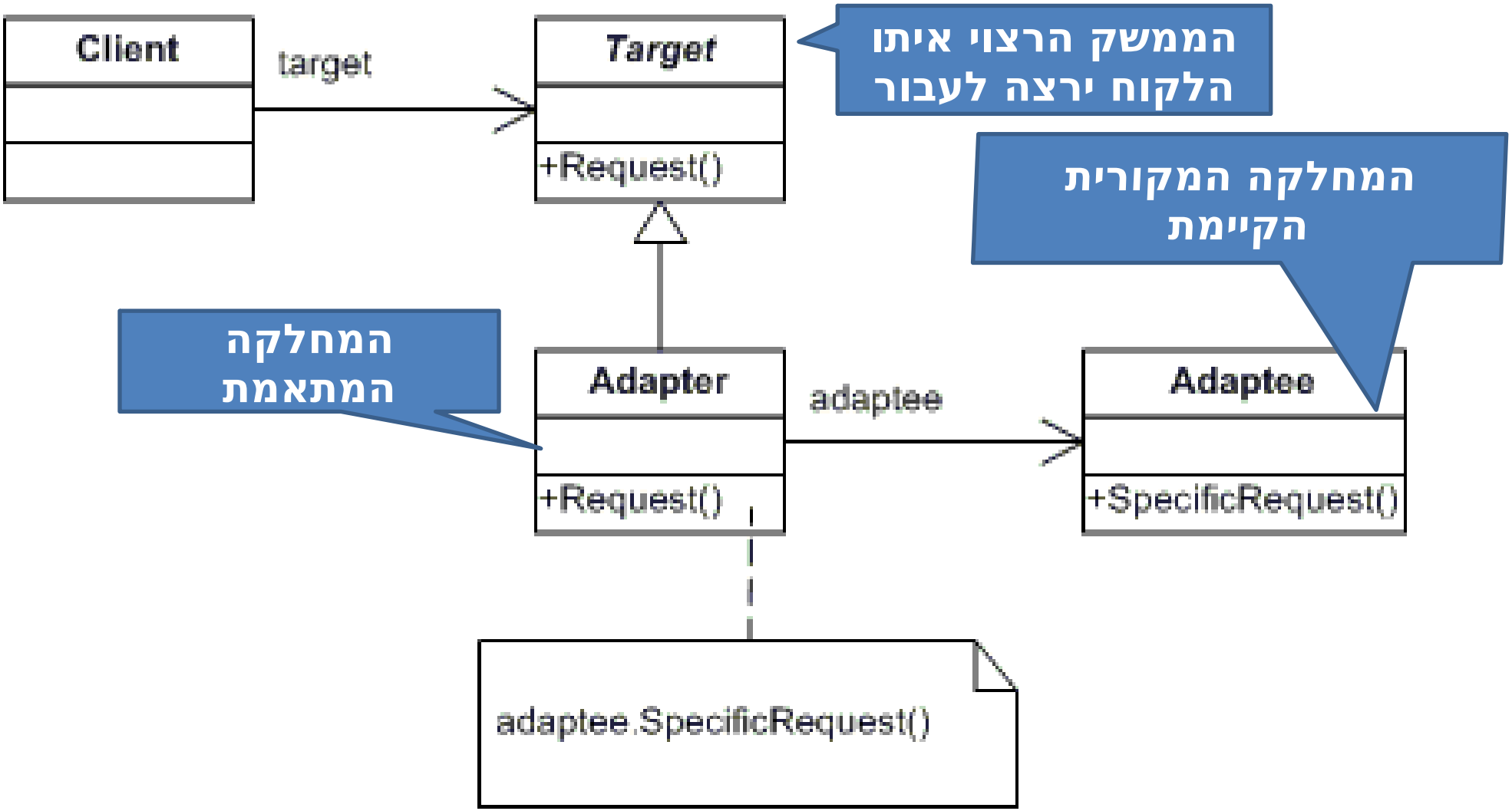
הרכב – תרשים מחלקות



מתאם – Adapter

- **המטרה**: להשתמש במחלקה א עם ממשק ב; מחלקה א לא מממשת את ב ואי אפשר לשנות את הקוד שלה.
  - דוגמה: שימוש ב-Iterator כ-Iterable.
  - דימוי: מתאם שקע חשמלי.
- **פתרון אפשרי**: מגדירים מחלקה שעוטפת את מחלקה א וממשת את ממשק ב.

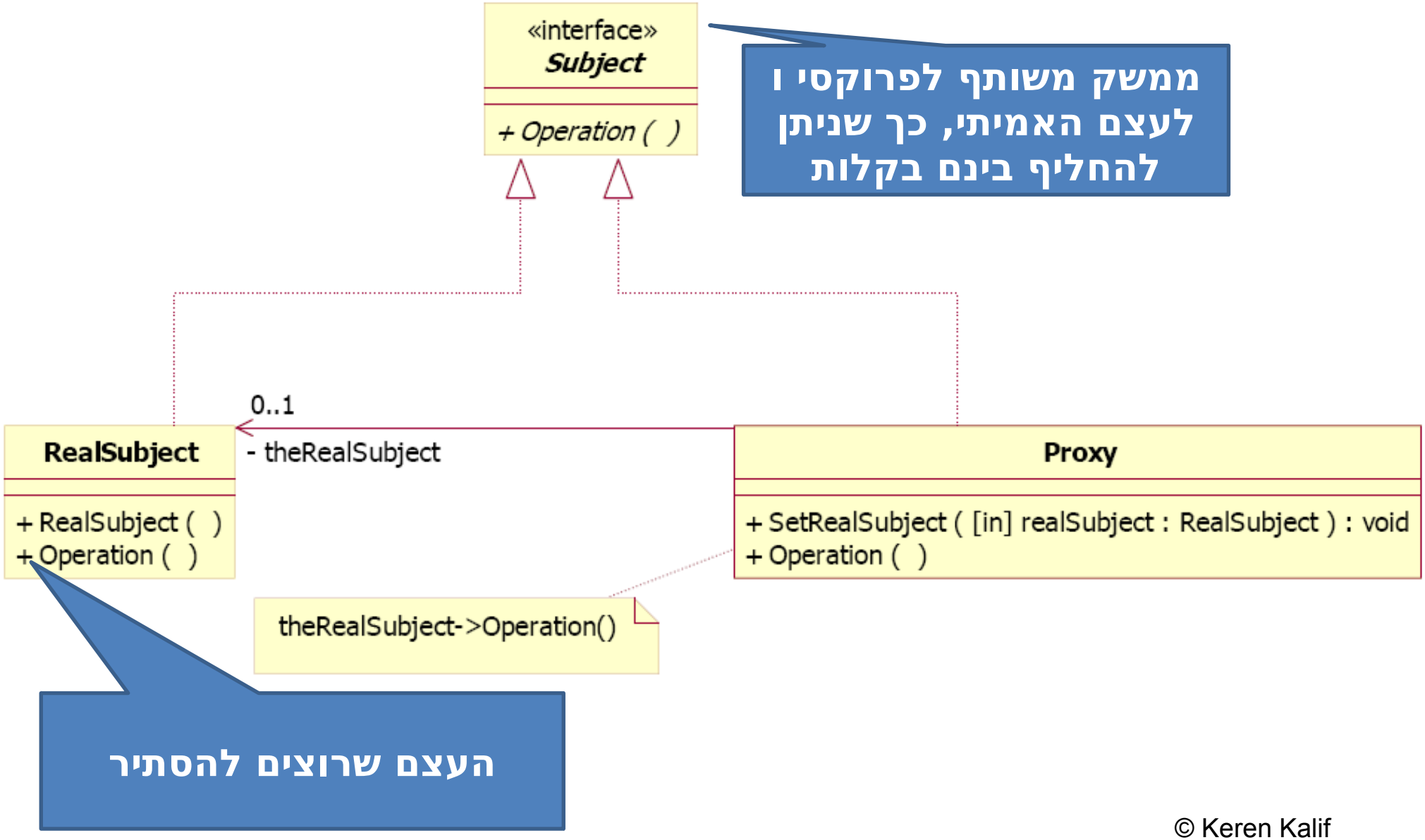
מתאם – תרשים מחלקות



פרוקסי ("בא כוח") – Proxy

- **המטרה**: הסתרת מתודות "מסוכנות" של מחלקה.
  - דוגמה: מחלקה לביצוע פעולות של מערכת ההפעלה.
- **פתרון אפשרי:** הסתרת עצם המסוכן בתוך עצם פחות מסוכן הכולל רק את הפעולות שרוצים לאפשר.

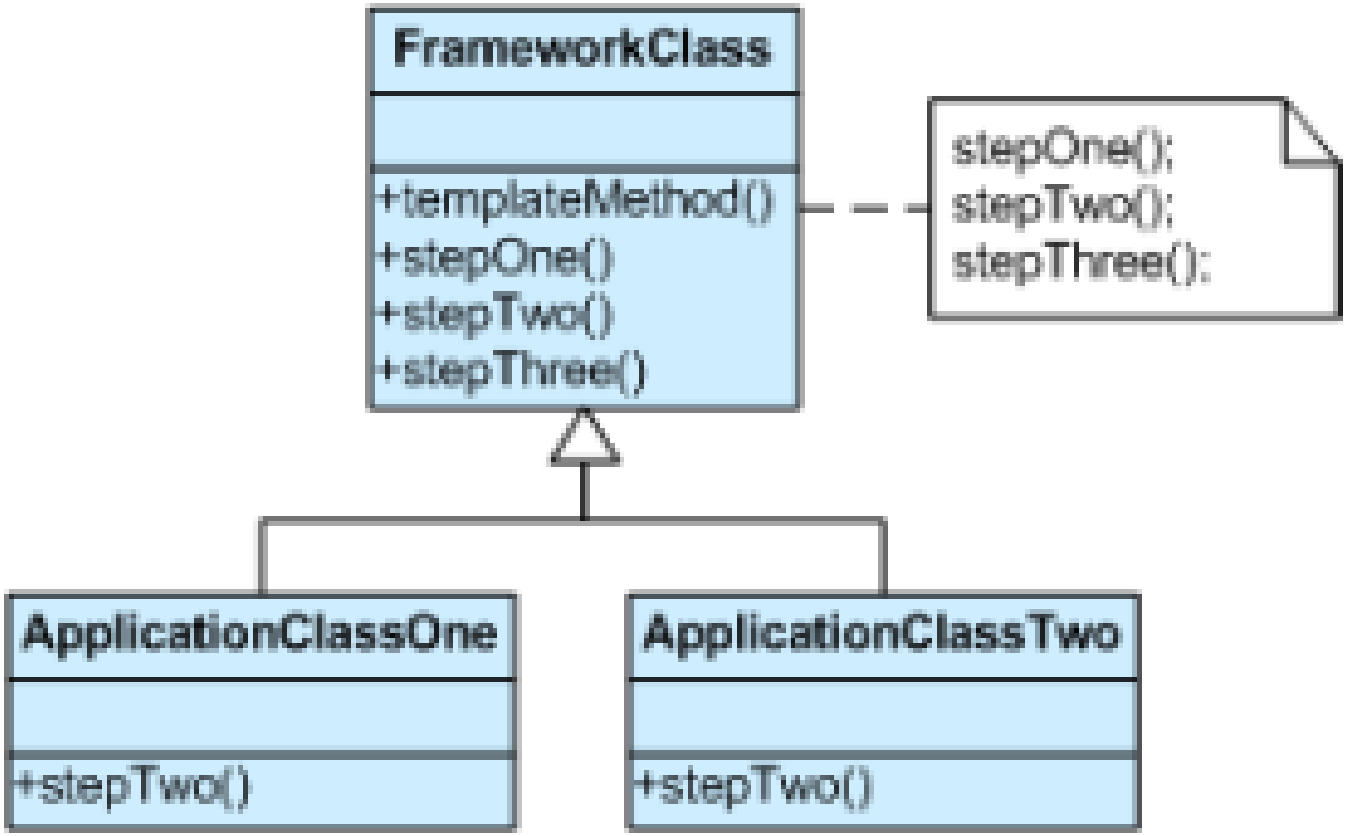
פרוקסי – תרשים מחלקות



מתודת תבנית – Template method

- **המטרה**: כתיבת אלגוריתם בראשי-פרקים; מימוש כל שלב בכמה דרכים שונות.
  - שימושי במיוחד בסימולציות של אלגוריתמים שונים.
- **פתרון אפשרי:** האלגוריתם יהיה מחלקה מופשטת; כל מימוש יהיה מחלקה יורשת.

תבנית – תרשים מחלקות



צופה - Observer

- **המטרה**: כשעצם מסויים משתנה, רוצים שהרבה עצמים אחרים יקבלו הודעה, אבל לא רוצים שכל אחד מהם יצטרך לבדוק בעצמו.
  - דוגמה: הודעה על שינוי בקובץ.
- **פתרון אפשרי:** כל מי שרוצה לקבל הודעה על שינוי, צריך להירשם אצל העצם שעלול להשתנות. כשהוא משתנה, הוא מודיע לכל הצופים שלו.

צופה – תרשים מחלקות