

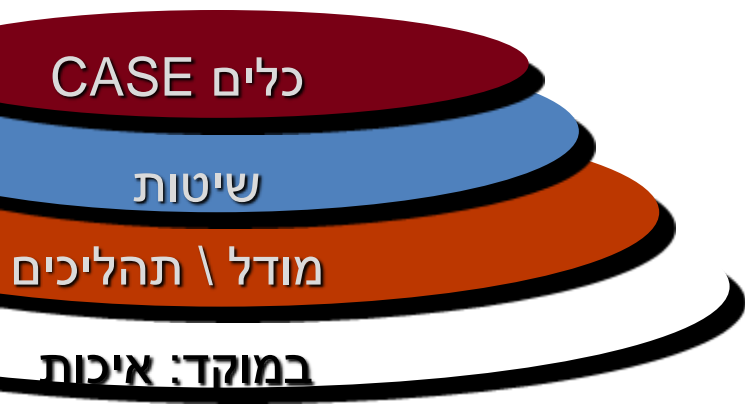
הנדסת תוכנה

8. בדיקות - II

הדגמה, פיתוח מונחה התנהגות

[Pragmatic Programmer Tip](#) : **Test Early. Test Often.**

Test Automatically. Tests that run with every build are much more effective than test plans that sit on a shelf.



מה היום?

- בדיקות (תזכורת)
- הדגמה
- פיתוח מונחה התנהגות (BDD)
- פרויקט
 - מצגות סבב 1
 - מעבר לסבב 2

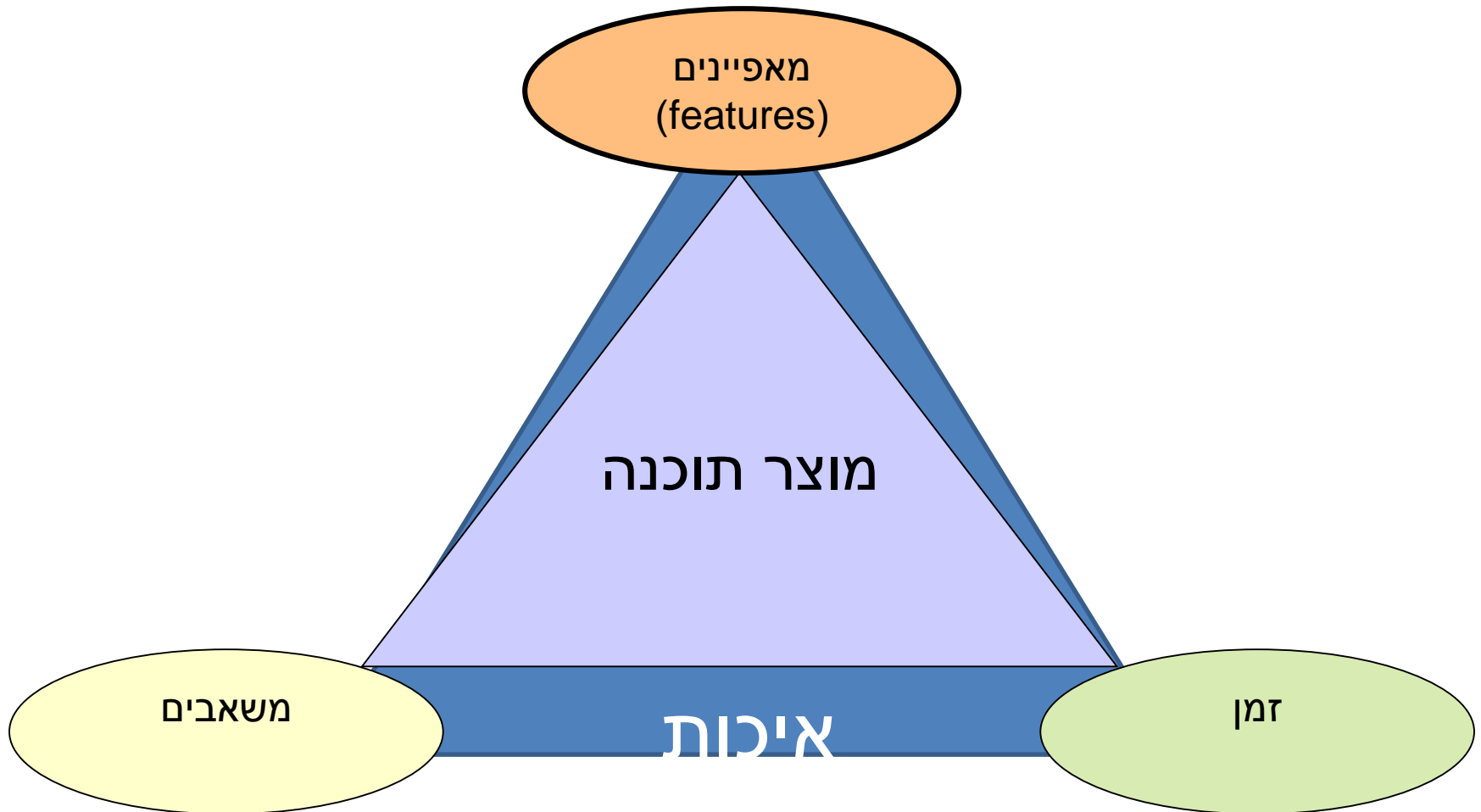
בדיקות תוכנה

- למה לבדוק?
- איך לבדוק?
- אילו בדיקות?
- מי בודק? מתי?
- כמה לבדוק?

Boeing 787 wing break test –

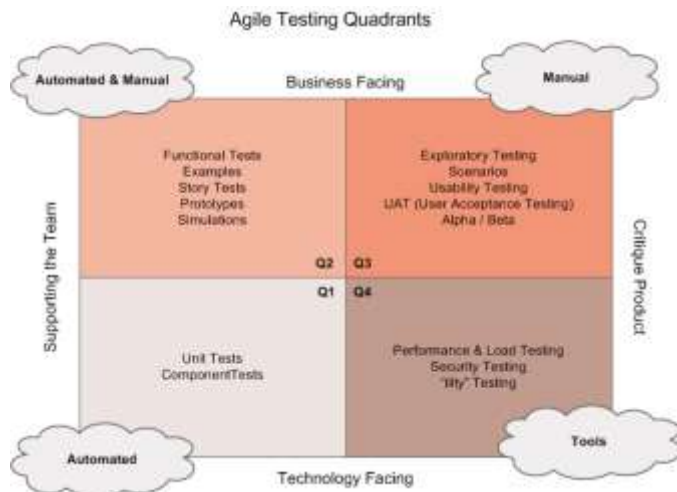
- האם זה באמת כדאי?
- למה עכשיו?

תזכורת: פרויקט תוכנה:



אילו בדיקות?

- דיבאג (ניפוי שגיאות)
- בדיקות יחידה (unit test)
- בדיקות עומס, בטיחות, גישוש, שמישות, A/B ועוד
- בדיקות אינטגרציה
- בדיקות קצה לקצה
- בדיקות מערכת
- בדיקות קבלה
- בדיקות רגרסיה
- סקרי קוד...



- [100 Types of Software Testing You Never Knew Existed](#)

ננסה להתמקד

- בדיקות קצה לקצה (משתמש\קבלה\פונקציונליות)
 - האם המערכת עובדת בשלמותה?
 - **בפרייקט**: ניסוח בדיקה באמצעות תרחש או סיפור
- בדיקות אינטגרציה
 - האם הקוד שכתבנו עובד מול קוד אחר
 - האם אי אפשר להסתפק בסוג הראשון?
- **בדיקות יחידה (מפתח)**
 - **האם המודולים עושים את הדבר הנכון? נוחים לשימוש? ע"י מי?**

בדיקת יחידה

- הגדרה: בדיקת יחידה היא קוד שקורא לקוד אחר ובודק אח"כ נכונות של טענות מסוימות. "יחידה" היא "קטנה" בד"כ פונקציה, מתודה – בד"כ נכתבת באמצעות framework (בהמשך)
- System Under Test (SUT) – הדבר שאותו אנחנו בודקים

בדיקות יחידה (אוטומטיות)

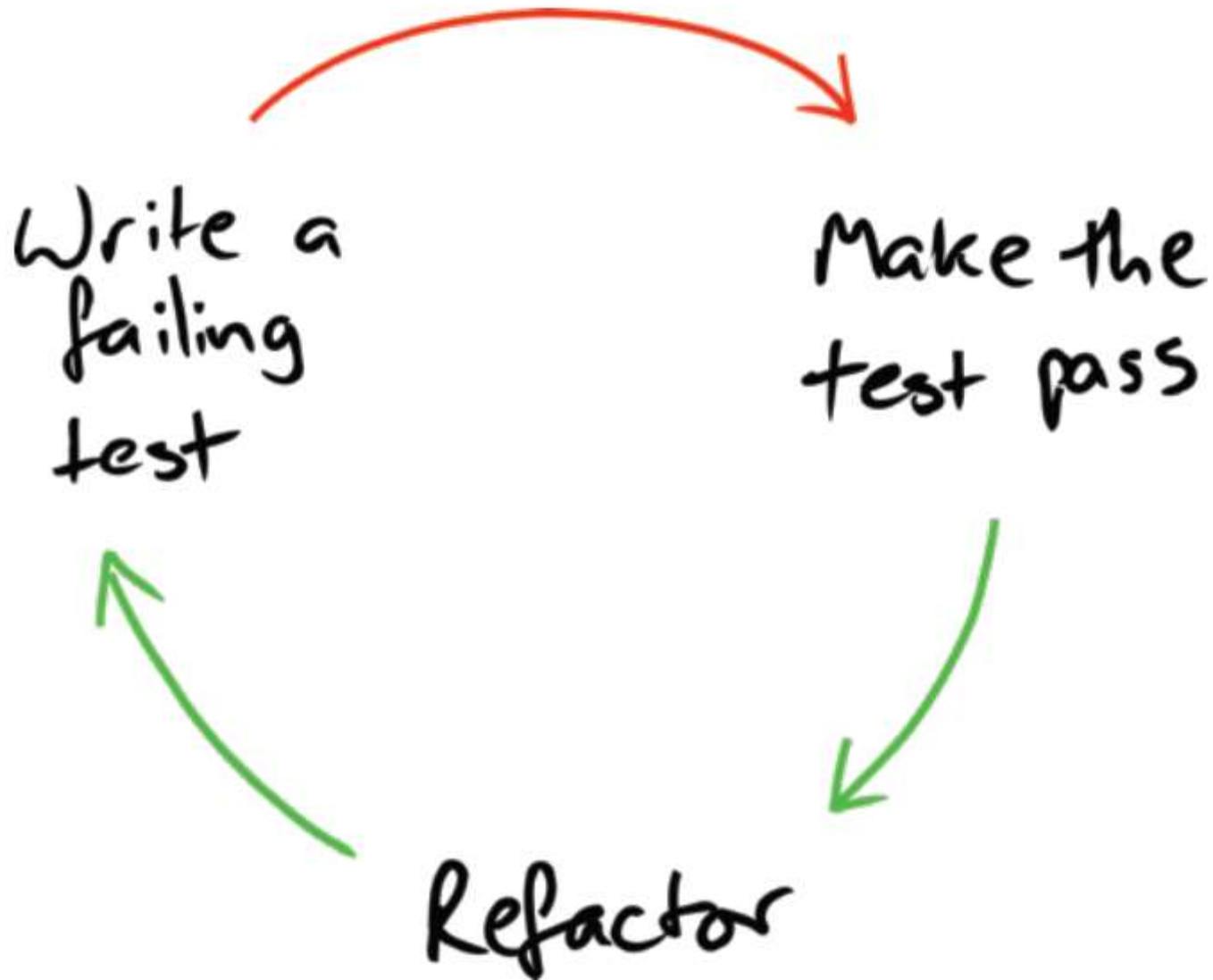


• יתרונות

- נכונות (ובמיוחד בשפות דינמיות)
- פחות זמן ב-debugger, רגרסיה
- תיעוד "חי"
- לעומת בדיקות אחרות: קלות ומהירות
- מאפשרות בדיקות ידניות משמעותיות יותר
- הורדת עלויות \ אפשרות לשינויים ...

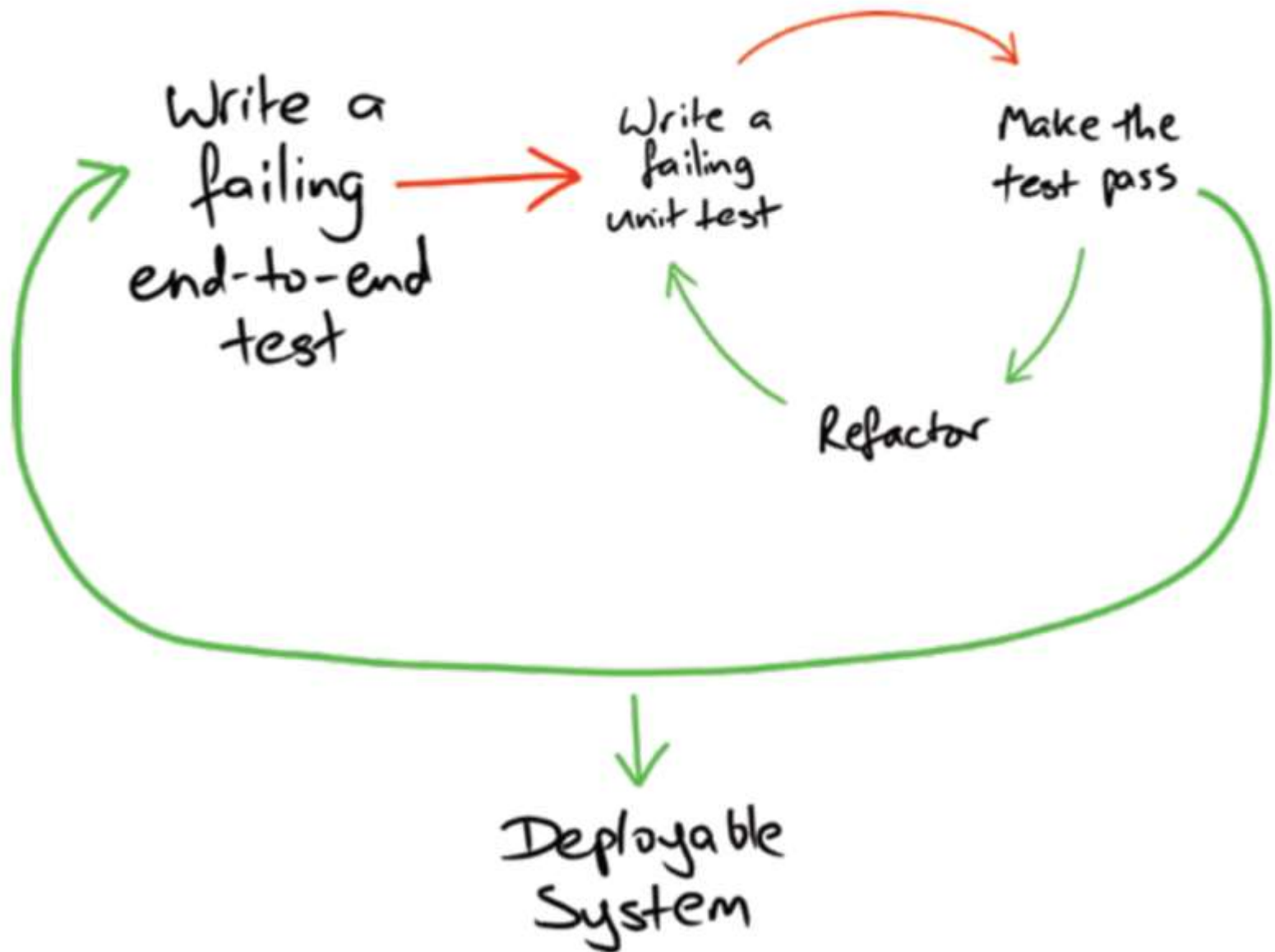
• חסרונות

- קוד (תחזוקה, תיכון, בדיקות – **כיצד נמנע זאת?**)
- זמן לימוד, כתיבה והרצה (אולי נסתפק באינטגרציה וקבלה?)
- יכולות לתת תחושת בטחון מזויפת
- לא תמיד קל עבור קוד קיים (legacy)



TDD יתרונות

- כיסוי טוב יותר ואוטומטי (ופחות באגים)
- תיכון: Test Driven Design, פשטות, חשיבה כלקוח (ראשוני של הקוד API), התמודדות עם הטיית אישור
- תיכון מתמשך – מודולריות, צמידות נמוכה, 'YAGNI, אפשור שינוי
- דיבאג מוקדם (מה קורה עם משאירים לסוף? אס"ק)
- חסרונות? (לעומת Test After Development)



Unit tests should be FIRST

(adopted from A. Fox, Berkeley)

- **Fast:** run (subset of) tests quickly (since you'll be running them *all the time*)
- **Independent:** no tests depend on others, so can run *any subset* in *any order*
- **Repeatable:** run N times, get same result (to help isolate bugs and enable automation)
- **Self-checking:** test can *automatically* detect if passed (*no human checking* of output)
- **Timely:** written about the same time as code under test (with TDD, written *first!*)

xUnit Frameworks

- כלים לבדיקות יחידה

- '94, Kent Beck, SUnit– Small Talk

- ~'00, +E. Gamma, JUnit ("Test Infected")

- ייצוא לשפות רבות: CppUnit, PyUnit ועוד

- <http://www.xprogramming.com/software>

- http://en.wikipedia.org/wiki/List_of_unit_testing_frameworks

- ארכיטקטורה סטנדרטית לבדיקות יחידה

רכיבים עיקריים בקוד בדיקה (JUnit)

// SUT

```
public class Calc {  
    public int add(int a, int b) {  
        return a+b;  
    }  
}
```

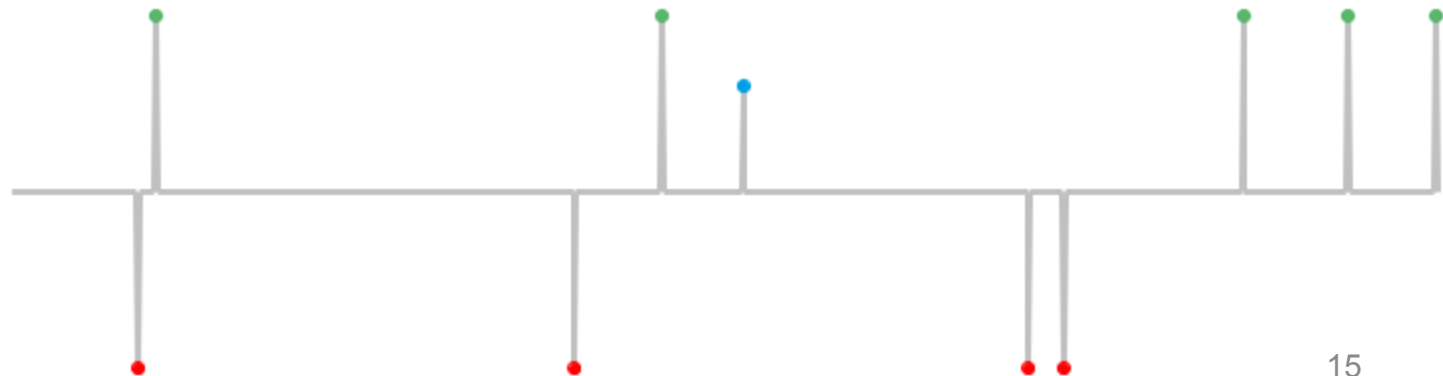
// Unit Test

```
import org.junit.Test;  
import static org.junit.Assert.assertEquals;
```

```
public class CalcTest {  
    @Test  
    public void testAdd() {  
        int result = new Calc().add(2, 3);  
        assertEquals(5,result);  
    }  
}
```

כלים - Java

- Eclipse + JUnit (built in)
- Optional plug-ins:
 - Git/github: Egit, Mylyn
 - Gamification: [pulse](#), [TDGotchi](#) (Help->Install New Software)
 - Code Coverage: EclEmma



Simple TDD

- FizzBuzz with Junit
- (nice TDD and Junit intro [slides](#))

TDD Coding Kata

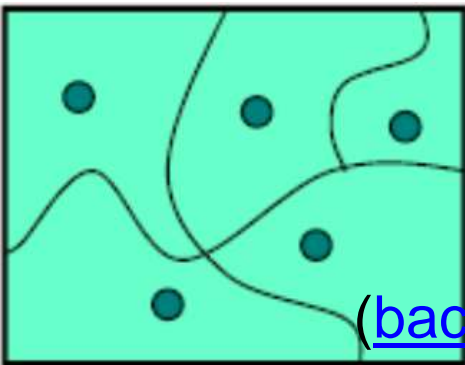
- String Calculator Kata
 - <http://www.osherove.com/tdd-kata-1/>
 - <http://www.21apps.com/agile/tdd-kata-by-example-video/>
- Many Others:
 - <https://github.com/garora/TDD-Katas>
 - Advanced: [GildedRose Kata](#) ([screencast](#))
- Yours?



הדגמה: RPN Calculator

- [SPEC](#) for TDD (real: [HP](#) and even OSx)
- Conclusions
 - Design by spec
 - Red-Green-Refactor, different hats
 - Commits (when?)
 - Code Smells: Feature Envy
 - BDD style: test fixture
 - One assert per test
 - Test vs. debug
 - Finally Push repository to github
 - Explore advanced ideas with this example? [video](#)

בהרצאת המשך \ נושאים מתקדמים



- בדיקות יחידה 2.0 למשל...

- בדיקות קבלה
- מאפיינים מתקדמים של xUnit: אתחולים, חריגות,
- סביבות שונות (.Net), קוד פתוח (למשל ב backbone.js)
- מחלקות שקילות, קופסא שחורה/לבנה, פרמטרים, כיסוי, תלות, אינטראקציה עם רכיבים אחרים, התנהגות מול מצב ([Google ToT](https://www.google.com/search?q=Google+ToT))
- כלים נוספים, אוטומציה, Continuous Integration
- בדיקות לניידים \ ענן \ רשת \ UI וכו'
- כיצד למצוא את [הבדיקה הבאה](#)
- כיצד להטמיע TDD בארגון? בדיקות לקוד קיים...

- משימה אישית: TDD (עמוד הבא)

- קריאה מומלצת להרצאת המשך:
[Using Mock Objects](#)



לסיכום

- בדיקות, פיתוח מונחה התנהגות
- בפרויקט

– בכל סבב: ניסוח בדיקת קבלת לתרחיש עיקרי

– סבב 2: חליפת בדיקות לרכיב מרכזי (וכתמיכה בשלד המוצר)