מחזור ב Java
תוכנות מונחת בדיקות
(Test Driven Development)
תרגול 8 – תスポה 1
String Immutability

- **Strings are constants**
  
  ```java
  String s = " Tea ";
  s = s.trim();
  s = s.replace('T', 'S');
  ```

- **A string reference may be set:**
  
  ```java
  String s = "Tea";
  s = "Sea";
  ```

```java
s
```

```java
" Tea ", "Tea", "Sea"
```

```java
s
```

```java
"Tea", "Sea"
```
String Interning

- Avoids duplicate strings

```java
String[] array = new String[1000];
for (int i = 0; i < array.length; i++) {
    array[i] = "Hello world";
}
```

An immutable string. Thus, can be shared.
String Interning (cont.)

- All string literals and string-valued constant expressions are interned.

- String literals: "Hello", "World"
- String-valued constant expression: "Hello" + "World"

Diagram:
- String literals and string-valued constant expression are interned in the String Pool.
String Interning (cont.)

- The *String* class has a static private pool of internal strings.

- `myString.intern()` implementation:

  ```java
  if \( \exists s \in pool : myString.equals(s) == true \)
  return s;
  else
  add myString to the pool
  return myString;
  ```

**equals**: compares characters
**==**: compares references
String Interning (cont.)

If:
String s1 = "ab";
String s2 = "ab" + "ab";
String s3 = "aba" + "b";
String s4 = s1 + s1;
String s5 = s1 + s1;
String s6 = s1 + "ab";

Then:

- \( s4.equals(s2) \) is true
- \( s4 == s2 \) is false
- \( s4 == s5 \) is false
- \( s2 == s3 \) is true
- \( s2 == s6 \) is false
- \( s4.intern() == s2 \) is true
- \( s4.intern() == s5.intern() \) is true
String Constructors

- Use implicit constructor:

  ```java
  String s = "Hello";
  // (string literals are interned)
  ```

Instead of:

```java
String s = new String("Hello");
// (causes extra memory allocation)
```
The StringBuilder Class

- Represents a **mutable** character string
- Main methods: `append()` & `insert()`
  - accept data of any type
  - If: \( \text{sb} = \text{new StringBuilder("123")} \)
    - Then: \( \text{sb}.append(4) \)
    - is equivalent to
      \( \text{sb}.insert(\text{sb}.length(), 4) \)

Both yield "1234"
The Concatenation Operator (+)

- String conversion and concatenation:
  - "Hello " + "World" is "Hello World"
  - "19" + 8 + 9 is "1989"

- Concatenation by StringBuilder
  
  ```java
  int i = 8; String x = "19" + i + 9;
  
  is compiled to the equivalent of:
  ```
  ```java
  String x = new StringBuilder().append("19").append(8).append(9).toString();
  ```
StringBuilder vs. String

■ Inefficient version using String

```java
public static String duplicate(String s, int times) {
    String result = s;
    for (int i = 1; i < times; i++) {
        result = result + s;
    }
    return result;
}
```

A new String object is created each time
StringBuilder vs. String (cont.)

More efficient version with StringBuilder:

```java
public static String duplicate(String s, int times) {
    StringBuilder result = new StringBuilder(s);
    for (int i = 1; i < times; i++) {
        result.append(s);
    }
    return result.toString();
}
```

no new Objects
public static String duplicate(String s, int times) {
    StringBuilder result =
            new StringBuilder(s.length() * times);
    for (int i = 0; i < times; i++) {
        result.append(s);
    }
    return result.toString();
}
StringBuilder vs. StringBuffer

- StringBuilder has the same API as StringBuffer, but with no guarantee of synchronization.

- StringBuilder is a replacement for StringBuffer when there is only a single thread.

- Where possible, it is recommended to use StringBuilder as it will be faster under most implementations.
Test-Driven Development By Example
By Kent Beck
Publisher: Addison Wesley
Date: November 08, 2002
ISBN: 0-321-14653-0
Pages: 240
bdd קבצי תחילה

- קוד נקי שעובד (clean code that works)
- קוד חדש 写ש 写ת רכז אחריו שבדיקה אוטומטית נכשלת
  `- הסרת כפילויות
  `- התכן (design) מלווה בקידוד
      `- המתקין 写ת את 百变ו
      `- קומפילציה מאירה
  `- קידוד שולש בין 百変
  `- קידוד שולש בין 百変

שלכות טכניות

- התכן (design) מלווה בקידוד
- המתקין 写ת את 百变ו
- קומפילציה מאירה
- קידוד שולש בין 百変
- קידוד שולש בין 百変
אדום – ירוק – שחטווב

◼ אדום
  ❑ כתוב בדיקה שנכשלת (אולי איפיול לא עוברת קומפילציה)

◼ ירוק
  ❑ תעשה במעיריהות השבבים הצלחת (חֶפּר אֹליי שְׁחֵית פּרוֹת
  קדשון של עקרונות תכוניים)

◼ שחטווב (refactoring)
  ❑ הסר את הכפיליות בקוד שעראהحسنתו של הקודבשלב הקודם
וזה דוגמה פשוטה
בצאתון찍ת פונקציה המחשבת איבר בסדרת

Fibonacci

Test First

לדוגמה, יiphyונאוץ

 כתוב את הפונקציה בהшла של
שלבי העבודה

1. Quickly add a test.
2. Run all tests and see the new one fail.
3. Make a little change.
4. Run all tests and see them all succeed.
5. Refactor to remove duplication.
AI: How do we start?

The definitions of the meaningful source code (script)...

...Wanna check if it's possible to perform in the system...

What do you want the function to do?

We'll create a class where the test code is placed.

```java
public class TestFib extends TestCase {
    public void testFibonacci() {
        assertEquals(0, fib(0));
    }
}
```

- TODO List:
  - fib(0) == 0

Note: The code is written in Hebrew.
JUnit Annotations

- @Test
- @Before, @After
- @BeforeClass, @AfterClass
- @Ignore
- @Test(timeout=100)
תעשה שיתקمل

■ נסיף קוד מיניילו כי לא תור את בעיות הקומפקilik

```java
int fib(int i){
    return 0;
}
```

■ נריצי... (את קוד הבדיקה)

■ ירק
נוסיף עוד בדיקה

אפשר להוסיפו עוד מתודה בדיקה חדשה:

```java
public void testFibonacciOfOneIsOne() {
    assertEquals(1, fib(1));
}
```

נדיר...

נקחנו נסתפק ב:

```java
public void testFibonacci() {
    assertEquals(0, fib(0));
    assertEquals(1, fib(1));
}
```

נריי... אדום

TODO List:
- fib(0) == 0
- fib(1) == 1
תעשה שהה ירוק

נוסיף קוד מינימלי כדי להפוך את הפס לירוק

```c
int fib(int n) {
    if (n == 0)
        return 0;
    return 1;
}
```

נрит... (את קוד הבדיקה)

ירוק

**TODO List:**
- fib(0) == 0
- fib(1) == 1
הסרת כפילויות

cf. The double checking is in the test (not in the code) — we remove them (refactoring)

```java
public void testFibonacci() {
    int cases[][] = {{0, 0}, {1, 1}};
    for (int i = 0; i < cases.length; i++)
        assertEquals(cases[i][1], fib(cases[i][0]));
}
```

We checked that nothing is broken (or no new bugs)

I see that there is no column with

(... new column against)

פירות
נוסיף עוד בדיקה

קל להוסיף את הבדיקה בפונקציה הבדיקה המשונתה

(6 הקשורים麦克לדת בלבד)

public void testFibonacci() {
    int cases[][] = {{0,0},{1,1},{2,1}};
    for (int i = 0; i < cases.length; i++)
        assertEquals(cases[i][1], fib(cases[i][0]));
}

נество

עדיין ירוק

TODO List:
- fib(0) == 0
- fib(1) == 1
- fib(2) == 1

ברר

עדין ירוק
public void testFibonacci() {
    int cases[][] = {{0, 0}, {1, 1}, {2, 1}, {3, 2}};
    for (int i = 0; i < cases.length; i++)
        assertEquals(cases[i][1], fib(cases[i][0]));
}

TODO List:
- fib(0) == 0
- fib(1) == 1
- fib(2) == 1
- fib(3) == 2
תעשה شيئיה ירוק

נוסיף קוד מינימלי כדי להפוך את הפס לרירוק

int fib(int n) {
    if (n == 0) return 0;
    if (n <= 2) return 1;
    return 2;
}

נוריצי... ירוק

_TODO List:

fib(0) == 0
fib(1) == 1
fib(2) == 1
fib(3) == 2

שכתוב:
מאיפה הגיעה הה-2?
זה בע="%d" 1+1
int fib(int n) {
    if (n == 0) return 0;
    if (n <= 2) return 1;
    return 1 + 1;
}

int fib(int n) {
    if (n == 0) return 0;
    if (n <= 2) return 1;
    return fib(n-1) + 1;
}
שכזה

fib(n-1) השני הוא בעと言う (2-n)כ

int fib(int n) {
    if (n == 0) return 0;
    if (n <= 2) return 1;
    return fib(n-1) + fib(n-2);
}

נוכליעבורה(2)כוסיונה

int fib(int n) {
    if (n == 0) return 0;
    if (n == 1) return 1;
    return fib(n-1) + fib(n-2);
}
Made a list of the tests we knew we needed to have working

Told a story with a snippet of code about how we wanted to view one operation

Made the test compile with stubs

Made the test run by committing horrible sins

Gradually generalized the working code, replacing constants with variables

Added items to our to-do list rather than addressing them all at once
בהקשר כללי יותר

- Write a test.
- Make it run
- Make it right
Write a test

- Think about how you would like the operation in your mind to appear in your code.

- You are writing a story. Invent the interface you wish you had.

- Include all of the elements in the story that you imagine will be necessary to calculate the right answers.
Make it run

- Quickly getting that bar to go to green dominates everything else
- If a clean, simple solution is obvious, then type it in
- If the clean, simple solution is obvious but it will take you a minute, then make a note of it and get back to the main problem, which is getting the bar green in seconds
- This shift in aesthetics is hard for some experienced software engineers
- They only know how to follow the rules of good engineering
- Quick green excuses all sins. But only for a moment
Make it right

- Now that the system is behaving, put the sinful ways of the recent past behind you

- Step back onto the straight and narrow path of software righteousness

- Remove the duplication that you have introduced, and get to green quickly
רואים עוד?

TDD עוד

XP עוד