מבוא לתיכנות מונחה עצמים – חריגות ובדיקות

חריגה – למה צריך את זה?

- מתודה נתקלה ב**מצב חריג** שאינה יודעת לטפל בו.
- המתודה מתריעה על המצב החריג (= זורקת חריגה).
- המתודה הקוראת יכולה לתפוס את החריגה ולטפל בה.
- אם המתודה הקוראת לא מטפלת בחריגה, היא ממשיכה הלאה למתודה הקוראת לקוראת, וכן הלאה, עד main.
- המטרה – מי שיודע לטפל במצב החריג, יעשה זאת.

מדרג החריגות

- Throwable
  - *Error*
    - *OutOfMemoryError*
    - *NoClassDefFoundError …*
  - Exception
    - *RuntimeException*
      - *NullPointerException*
      - *ArithmeticException*
      - *IllegalArgumentException…*
    - IOException
      - FileNotFoundException
      - UnsupportedEncodingException….

חריגות לא נבדקות

חריגות נבדקות

גם אתם יכולים להגדיר חריגות

```
public class IllegalColorException extends
RuntimeException {
   public IllegalColorException()
      {super();}
   public IllegalColorException(String message)
      {super(message);}
}
```

- *RuntimeException*
  - *NullPointerException*
  - *ArithmeticException*
  - *IllegalArgumentException*
  - ***IllegalColorException***

מה עושים עם חריגה כשתופסים אותה?

תלוי בסוג התוכנית ובסוג החריגה. יש כמה אפשרויות:

- הדפסה למסך – getMessage או toString
הדפסת שרשרת הקריאות למסך – printStackTrace.
- הדפסה לקובץ יומן (log) – לבדיקה ע"י המתכנתים.
- עטיפה בחריגה אחרת וזריקה מחדש – לטיפול ע"י רמה גבוהה יותר.

סגירת משאבים

- כשפותחים קובץ לקריאה או כתיבה, צריך לסגור אותו.
- אבל מה קורה אם יש חריגה לפני הפעולה close?
- הפתרון – לפתוח את הקובץ בתוך סוגריים אחרי ה-try.

בדיקות פנימיות – assert – למה זה טוב?

- כדי להכניס לקוד בדיקות פנימיות לצורך ניפוי-שגיאות.
- כברירת-מחדל, הבדיקות **לא יתבצעו**, כדי לחסוך זמן.
- אם רוצים שיתבצעו, צריך להפעיל את java עם פרמטר **-enableassertions** או **-ea**.
- בדיקות מסוג זה שימושיות ב**בדיקות-יחידה** – Unitest.
- כלי שימושי במיוחד לביצוע בדיקות יחידה – **Junit**.