אוניברסיטת
אריאל
בשומרון

תכנות מונחה עצמים – שיעור 1

בעז בן משה, benmo@g.ariel.ac.il

היכרות עם הקורס, חזרה על מבוא לחישוב:
ממשקים, ירושה, ספריות מתמטיות, קבצים,
טיפול בחריגים, מטלה 0

שיעור 1: נושאים

- היכרות, מבנה הקורס, סקר
- ממשקים,
- ירושה,
- עקרונות OOP
- Comperator,
- טיפול בחריגים – שימוש בסיסי
- שימוש בקבצים (טקסט)
- הקוד של השיעור:

https://github.com/simon-pikalov/Ariel_OOP2020_

- מטלה 0:

שיעור 1: הכירות + סקר

- היכרות
- צוות ההוראה
- מסגרת הקורס: סילבוס, מטלות - PBL
- שיטת לימוד \ הוראה: הכל במודל וב github
- מטרה: הולכים להיות מתכנתים + מתכננים טובים: "לומדים לשחות כמו שצריך"
- סקר: Java, ירושה?, ממשקים?, קבצים?, python?, אנגלית (English)?, Github?, MOOC?, עבודה בתחום?

לפני שמתחילים

דרישות:

- קורס מבוא בהנדסת תוכנה – ממש חשוב!
- כמה מילים על הנדסת תוכנה, קוד פתוח, out-sourcing.
- הקורס מתואם עם "אלגוריתמים 1", "תקשורת"
- יושרה: תקנון המחלקה, לא מעתיקים!
- "נאה דורש": כלים טובים ועבודה קשה

- מטרה: בעוד 3 חודשים (וגם עוד 10 שנים) תדעו לתכנת ו**לתכנן** באופן איכותי.

ממשקים

ממשק interface: מילה שמורה בjava שמייצגת אוסף תכנות (שיטות) ללא קוד – חתימות בלבד.

- מנגנון מאוד יעיל ושימושי, בפרט דוגמאות רבות ניתן למצוא במבני נתונים, ואלגוריתמים
- מאוד שימוש כדי להגדיר חלוקת עובדה ואבסטרקציה.
- מהווה מנגנון API – הכי נפוץ
- עוזר להפריד בין "מה"(ממשק) ל"איך" (קוד)

- הדגמה מתוך מטלה 0 לממשקים

ירושה

ירושה היא שיטה "למחזור קוד", בדומה לממשק מנגנון זה מאפשר להגדיר "התנהגות משותפת":

- מנגנון מקובל למחזור קוד בעיקרון "is a"
- מקרה פרטי של הכלה – ירושה יחידה בלבד!.
- אם יש קוד משמעותי שרוצים למחזר – ירושה בכל מקרה אחר - ממשקים

- דיון בדוגמא של נקודות, וצורות

עקרונות OOP

אבסטרקציה: ייצוג העולם ע"י מחלקות

- הכמסה: encapsulation: public / private: מינימום שיטות ציבוריות
- מחזור קוד: (קצת מסובך): ירושה,
- פולימורפיזם: (רב צורתיות) לאפשר מכנה משותף לוגי.

עקרונות תכנות בסיסיים

- KIS: Keep It Simple
- תכנון, ובדיקות: הם הכרחיים
- חשוב להכיר היטב את סביבת הפיתוח: Eclipse
  ו
- ריבוי איטרציות refactoring (תכתבו מחדש)
- להסתכל\להריץ\לשנות על קודים של אחרים

- תשקיעו הרבה – לומדים דרך האצבעות

טיפול בחריגים

נושא שדורש "הבנה" ויכולת תכנון

- חלק גדול מהקוד בעולם עוסק בטיפול במקרי קצה ושגיאות.
- דוגמאות: חריגה במערך, פתיחת קובץ לא קיים, חלוקה באפס
- כלל: נזרוק שגיאה: אם אנחנו מזהים בעיה ולא יודעים לטפל בה. (המילה השמורה throws)
- מילים שמורות: throw, throws, Exception
- דוגמא מקבצים

שימוש בקבצים

חלק גדול מהתוכנות בעולם מקבלות קובץ כקלט ומייצרות קובץ כפלט.

- כתיבה לקובץ טקסט דוגמא בסיסית
- קריאה מקובץ – ממש כמו כתיבה readline
- מומלץ לעשות "אבסטרקציה" לאפליקציות כמו: כתיבה ל"לוג", קריאת פרמטרים מקובץ.

דיון בהנדסת תוכנה

אוניברסיטת אריאל בשומרון

כיצד מתחילים לתכנן:

- חלוקה למחלקות, ממשקים
- כתיבת מערכת בסיסית (פשוטה ביותר עובדת), ושיפור מתמיד
- חייבים לבדוק!
- הקפידו לעבוד בתצורה איכותית – לא לכתוב קוד שלא מבינים, לא להתעלם מבאגים
- לא לפחד להתחיל לכתוב מחדש.
- דוגמאות מתוך הקוד של השיעור.

דיון במטלה 0

מידול מדול מערכת שמתזמנת מעליות חכמות בבניין:
מבנה נתונים + אלג'

- הבנת המודל
- הבנת הממשקים
- כתיבת Testers
- מקרי קצה – לפי מיטב ההבנה שלכם (יש "נכון" ו"לא נכון")
- לא לשכוח לתעד + כתיבת Readme

דיון במטלה 0

שאלות:

- הבנת הבעיה - לפחות 50% מהמטלה!
- הבנת הממשקים
- הגדרת המסגרת ובדיקות
- מקרי קצה – לפי מיטב ההבנה שלכם (יש "נכון" ו"לא נכון")
- לא לשכוח לתעד: כל מחלקה, כל שיטה ציבורית, כל אלג' שאינו טרוויילי.