## הערות ממהלך עשיית מבחנים:

**הורשה:**

- בנאים והורשה: תמונת המצב צריכה להיות כך, שאם במחלקת אם יש בנאי מפורש **עם משתנים**, גם ביורשיה צריכים להיות בנאים המפעילים super על אותם משתנים. אם אין בנאי מפורש או אין בנאי כלל, לא חייבים לשים בנאי מפורש/כלל ביורשים.
- לא ניתן לדרוס מתודה סטטית, לכן אם עבור טיפוס כלשהו נקראת מתודה סטטית **ישירות**, תופעל זו המופיעה **בטיפוס הסטטי**.
- **לעומת זאת**, אם נקראת מתודה סטטית **מתוך מתודה דינאמית**, אז בפועל תקרא המתודה הסטטית הראשונה שנתקל בה במעלה עץ הירושה **החל מהטיפוס הדינמי**, ולא נשלח ישירות אל המתודה הסטטית בטיפוס הסטטי.
- אם מתודה כלשהי אינה נמצאת **בטיפוס הסטטי** של עצם, אז הקוד **לא מתקמפל**.
- מתודות המופיעות במוריש וביורש עם אותה חתימה חייבות להיות סטטיות / לא סטטיות יחד.
- Overloading של מתודות **אינו אפשרי אם השוני היחיד בחתימה הוא הערך המוחזר.** אם גם הפרמטרים שונים, זה תקין.

**משתנים והרשאות:**

- משתנה final מסוג הפניה – אסור לשנות את ההפניה עצמה, אך מותר לשנות את שדות העצם אליו מצביע.
- אם משתנה הפניה final נשלח למתודה, במתודה נוצר משתנה הפניה חדש המצביע אל אותו עצם, אך משתנה ההפניה החדש המקומי של המתודה **אינו** final, ולכן ניתן לשנות אותו (כמובן שאת שדותיו, מההערה הראשונה).
- Casting "בטוח" (בלי חשש לאיבוד מידע) ניתן לעשות אוטומטית, כמו int ל-double.
- Casting לא חוקי הוא **שגיאת זמן ריצה** ולא שגיאת קומפילציה.
- חריגת מערך מגבולותיו הוא **שגיאת זמן ריצה** ולא שגיאת קומפילציה.

**מחלקות:**

- כל השדות של הבנאי שלא הוצב בהם ערך דיפולטי יאותחלו ל-0 או null בעת קריאה לבנאי (או כמובן שיוצבו בהם ערכים שמקבל הבנאי).
- על כל מתודה שכותבים לחשוב האם היא פר מופע או שהיא צריכה להיות static? במחלקות עזר לרוב יהיו מתודות סטטיות, לעומת מחלקות המתארות עצמים.
- **לא ניתן ליצור מערך מטיפוס גנרי!!** במקום זה, נשתמש ב-casting: T[] arr = (T[])(new Object[<some number>]);
- כאשר עושים override ל-equals חייבים לקבל כפרמטר **Object** (ולעשות לו casting).
- אם נדרשים לממש טיפוס, ובוחרים במימוש פנימי כלשהו, למשל מערך, ואחת המתודות שמתבקשים לכתוב היא להחזיר את ערכי העצם בצורת המימוש הפנימי, למשל להחזיר את תוכן העצם בצורת מערך, אז **אסור להחזיר את המימוש הפנימי**, כי אז עלולים לשנותו. צריך להחזיר אובייקט חדש מועתק.

**חוזים:**

- צורת כתיבת פונקציית הפשטה:

  AF:<שם המחלקה> → <הצגה מופשטת>

  AF(this) = <הצגה מופשטת בהקשר לנתונים>  e.g:

  AF:Triangle → (P1, P2, P3)

  AF(this) = (points[0], point[1], points[2])

- חוזה עם חריגים צריך להכיל @throws באופן הבא: @throws <cond> $implies throwing <exception>

**אחר וכללי:**

- אם מתבקשים לממש איטרטור בגבולות מתודת האיטרטור (בהינתן מנשק לאיטרטור ששמו xxxIterator), ניתן לכתוב בגוף המתודה:

return new xxxIterator() { מחלקה אנונימית המממשת את המנשק };

- מאזינים ואדפטרים ב-GUI:
  - אם נתבקשנו לכתוב מאזין, אז נשתמש בפקודה

someShell.addSomethingListener(new MyListener());

…

Class MyListener **implements** SomethingListener{…} – מימוש ישיר של מנשק המאזין

  - אם נתבקשנו להשתמש באדפטר, נעשה אותו דבר, רק שהמחלקה הנוספת תרחיב את האדפטר (שהוא כידוע מחלקה מופשטת המממשת את המאזין).

**חריגים:**

- אם טיפוס סטטי כלשהו זורק חריג, כל המתודות על העצם (לא משנה איזה טיפוס דינמי) צריכות לטפל באותו חריג או במי שמעליו בירושה (לא מספיק טיפול במי שתחתיו), ובכלל:
- מתודה המשתמשת במתודה הזורקת חריג מסוג כלשהו – **כלומר מופיע בחתימתה**, **חייבת להכריז על זריקת החריג בחתימתה או לעטוף את החלק הבעייתי ב-try-catch המטפל באותם חריגים או אבות אותם חריגים.**
- מתודה המכריזה על זריקת חריג מכל סוג שהוא ולא זורקת **כן מתקמפלת**.
- Unchecked Exceptions:
  - מתודה זורקת חריגים מסוג זה **לא חייבת** להצהיר בחתימה (או לעטוף ב-try-catch).
  - מסקנה: אם קיים מנשק שממממשות אותו מחלקות שחלקן בעלות מתודות שזורקות חריגים מסוג זה, וחלקן לא, המנשק **לא חייב להכריז בחתימות על זריקה.**
- Checked Exceptions:
  - מתודות זורקות חריגים מסוג זה **כן חייבות** להצהיר בחתימה (או לעטוף ב-try-catch).
  - מסקנה: אם קיים מנשק שממממשות אותו מחלקות שחלקן בעלות מתודות שזורקות חריגים מסוג זה, וחלקן לא, המנשק **כן חייב להכריז בחתימות על זריקה** (לא תהיה בעיה למתודות במימושים שלא זורקות בפועל).
- מבחינת חוזה למנשק: נרצה חוזה **מחמיר** שכן **לא כל המחלקות (אלו שלא זורקות חריגים) מטפלות במקרי הקצה!**