מנשקים (Interfaces)

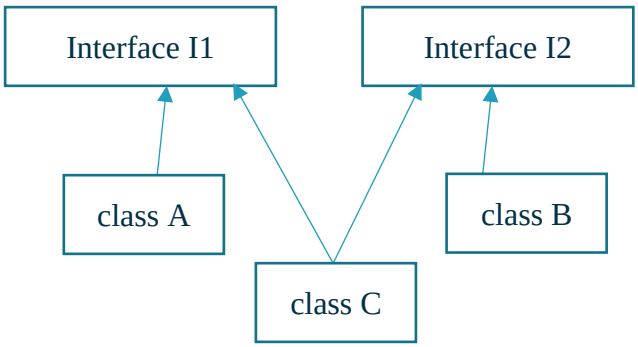
מה נכלל בעבודה עצמית זו?
- ✓ מנשקים
- ✓ הכללות

חלק א – מנשקים:

הורידו את הקוד אשר נמצא בקיצור הבא:
http://courses.cs.tau.ac.il/software1/2021a/lectures/tutorials/resources/class5.zip

והוסיפו אותו לתוך תיקיית src של הפרויקט עליו אתם עובדים. הקובץ מופיע ב Moodle יחד עם מסמך התרגול.

היררכיית המחלקות היא כזו:



עברו על הקוד של שני המנשקים ושלושת המימושים, וודאו שהוא ברור.

מספר דגשים:

א. במנשק I1 מוגדרת מתודה סטטית. מתודה סטטית, כזכור, לא צריכה מופע של המחלקה, ולכן ניתן לקרוא לה באופן הבא:

```
I1.func(true);
```

שימו לב ששירות זה אינו זמין מתוך המחלקות A ו C, ולכן לא ניתן לכתוב:

```
A.func(true);
```

ב. בתוך המנשק I2 מוגדרת מתודה דיפולטית. זוהי מתודת מופע שקיים לה מימוש בתוך המנשק. המימוש שלה ישתמש בשני המימושים של הפונקציות האבסטרקטיות f1 ו f2 שבהן היא משתמשת.

הריצו את הקוד הבא (בתוך מחלקה כלשהי עם main)

```
B b = new B();
C c = new C();
System.out.println(b.defFun(1, 2, 3));
System.out.println(c.defFun(1, 2, 3));
```

וודאו שהתוצאות ברורות לכם. מומלץ להריץ את הקוד ב debug ולעקוב אחרי הצעדים.

כל קריאה ל defFun מפעילה בעצם שלוש פונקציות. בשורה האחרונה, הקריאה ל defFun מפעילה את defFun הממומשת ב I2, והיא מפעילה את הפונקציות f1 ו f2. בתוך I2 אין מימוש לשתי הפונקציות האלה, המימוש שלהן מופיע בתוך C. למה הלכנו דווקא ל C? כי הפונקציה defFun הופעלה על c, שהוא אובייקט מטיפוס C.

ג. בתוך A ממומשת מתודה שאינה קשורה למנשק (המתודה foo). אין אם זה שום בעיה. A מחוייבת לממש את כל המתודות האבסטרקטיות במנשק, אבל אין שום מניעה להוסיף שירותים אחרים.

ד. המחלקה C מממשת שני מנשקים, I1 ו I2. מחלקה, כזכור, יכולה לממש כמה מנשקים שהיא רוצה. ומה קורה אם בשני המנשקים מוגדר אותו שירות אבסטרקטי? אין שום בעיה. גם I1 וגם I2 מגדירים את הפונקציה f1 עם חתימה זהה. C מממשת את הפונקציה הזו פעם אחת, וזה "מספק" את שני המנשקים.

ה. מה היה קורה אם בשני המנשקים היתה פונקציה דיפולטית זהה? פה המצב היה בעייתי יותר. אתם מוזמנים לנסות את זה בקוד ולראות מה קורה, נדבר על זה בכל מקרה בהמשך הקורס.

<u>חלק ב – הכללות:</u>

בשאנחנו מסתכלים על A, B, C אנחנו רואים של A ו ל C יש מכנה משותף, זהו I1, וגם ל B ול C יש מכנה משותף, זהו I2. ולא רק זה, A הוא בעצם סוג של I1, ו B הוא סוג של I2. כיצד ניתן לבטא קשר זה בקוד? הוסיפו את הקוד הבא ל main שלכם:

```
A a1 = new A();
I1 a2 = new A();
a1.f1(1, 2);
a1.foo();
a2.f1(1, 2);
a2.foo()
```

מה בעצם קורה בשורה השנייה? אנחנו מגדירים משתנה מטיפוס I1 להצביע על אובייקט מטיפוס A. האם זה חוקי? כן! כל אובייקט מטיפוס A מקיים יחס is-a עם הטיפוס I1, כלומר, כל A הוא סוג של I1.
אנחנו רואים פה בעצם התנהגות שהיא חדשה – מגדירים משתנה שלא תואם לטיפוס של האובייקט.
כשנסתכל על a2, אנחנו בעצם רואים שני טיפוסים:
הטיפוס הסטטי – הטיפוס של a2 שמוגדר פעם אחת, בעת הגדרת המשתנה a2. הטיפוס הסטטי של a2 יישאר I1 לנצח.
הטיפוס הדינאמי – הטיפוס של האובייקט עליו a2 מצביע. במקרה זה, הטיפוס הדינאמי הוא A. אם היינו עושים השמה ל new C() הטיפוס הדינאמי היה C. למה הוא דינאמי? כי במהלך ריצת התוכנית a2 יכול להצביע לאובייקטים שונים מטיפוסים שונים המממשים את I1.

עכשיו נתבונן על שורות 5 ו 6. שורה 6 אינה מתקמפלת. מדוע?
בשלב הזה, הקומפיילר רואה את המשתנה a2 ומסתכל רק על הטיפוס הסטטי שלו. הטיפוס הדינאמי לא נלקח בחשבון בשלב הקומפילציה. לכן, מבחינתו a2 הוא מצביע ל"משהו" שיש לו רק את התכונות של I1.
האם ל I1 יש את השירות foo? לא! לכן הקוד לא מתקמפל.
מדוע הקומפיילר מתעלם מהטיפוס הדינאמי? כי לא תמיד ניתן לדעת אותו.
דוגמת הקוד הבאה מבהירה את זה:

```
I1 a2;
if (Boolean.getBoolean(args[0])) {
	a2 = new A();
}
else {
	a2 = new C();
}
a2.f1(1, 2);
```

ניתן לראות כי בשורה האחרונה, אין לקומפיילר (וגם לנו) שום מושג אם a2 מצביע ל A או לא C, ולכן הדבר היחיד ש"מובטח" לנו הוא שהמימוש של f1 ו f3 קיים. מעבר לזה, לא ניתן להסתמך על קיומו של שום שירות אחר.
הערות:

א. האם a2 יכול להצביע על אובייקט מטיפוס B? התשובה היא שלא. השמה כזו תיכשל כיוון ש B לא מממש את המנשק I1.

ב. מה היה קורה אם גם ב C היתה ממומשת הפונקציה foo, בדיוק כמו ב A. לכאורה, אין שום מניעה לקרוא ל foo מתוך a2, כי גם A וגם C מממשות אותה. זה כמובן לא יעבוד, מדוע? כי זה לא משנה.

הוא פועל רק לפי הטיפוס הסטטי. אם היינו רוצים לקרוא ל foo מתוך a2, לא מספיק שגם A וגם C יממשו אותה, היא צריכה להופיע במנשק. זה יבטיח גם שכל מימוש אחר של המחלקה I1 יממש את הפונקציה הזו, ולא נצטרך "לעקוב אחר שרשרת ההשמות של a2" בשביל לוודא שהפונקציה הזו קיימת עבורו.

הדבר הבא שנוכל לעשות הוא לכתוב קוד כזה:

```
I1[] arr = new I1[2];
arr[0] = new A();
arr[1] = new C();
for (I1 i1 : arr) {
	System.out.println(i1.f3("abc"));
}
```

ראינו שמשתנה מטיפוס I1 יכול להצביע על אובייקטים מטיפוס A ו C, ולכן אין שום בעיה להכניס אותם לאותו המערך.
בשנכתוב את הלולאה על המערך, משתנה הלולאה חייב להיות מטיפוס I1, כיוון שזה הדבר היחיד שאנחנו יודעים על המערך. באיטרציה הראשונה תיקרא f3 של A ובשנייה של C, בהתאם לטיפוס האמיתי של האובייקטים. מומלץ להריץ את הקוד הזה בdebug ולראות כיצד הוא עובד.