ירושה למתקדמים

הסתרת שיטות

נניח שבמחלקת הבסיס יש פונקציה f המקבלת פרמטר bool.

במחלקה היורשת יש פונקציה f המקבלת פרמטר int.

לכאורה אלו שתי שיטות שונות - יש כאן העמסה ולא החלפה.

הבעיה היא, שהקומפיילר אוטומטית ממיר bool ל-int. לכן, אם ניצור עצם מהמחלקה היורשת ונקרא ל-f עם true, הקומפיילר ימיר אותו ל-1 ויקרא לשיטה המחליפה - השיטה המחליפה למעשה מסתירה את השיטה הבסיסית ולא מאפשרת לקרוא לה ישירות.

פתרון אפשרי הוא, במקום לקרוא לשיטה f לקרוא לשיטה Base::f, למשל:

```
d.Base::f(true);
```

הבעיה, שבמקרה כזה הקומפיילר תמיד יקרא לשיטה של מחלקת הבסיס - גם אם נעביר 5, הוא ימיר אותו ל-true ויקרא לשיטת הבסיס.

מה עושים כדי שהקומפיילר יבחר את השיטה המתאימה לפי סוג הארגומנט?
-- מוסיפים במחלקה היורשת את ההוראה:

```
using Base::f
```

זה אומר לקומפיילר, שאנחנו רוצים שבמחלקה היורשת יהיו שתי פונקציות בשם f באותה דרגת חשיבות, כך שהוא יבחר ביניהן רק לפי סוג הארגומנט.

ראו דוגמה בתיקיה 1.

ירושה כפולה

בשפת C++, בניגוד לג'אבה, מחלקה אחת יכולה לרשת שתי מחלקות או יותר.

דוגמה לשימוש: יש לנו מחלקה המייצגת קובץ קלט ומחלקה המייצגת קובץ פלט. אנחנו רוצים מחלקה המייצגת קובץ שיכול לשמש גם לקלט וגם לפלט.

איך עושים את זה? פשוט:

```
class IoFile: public InputFile, public OutputFile{...}
```

סדר הבניה והפירוק יהיה לפי הסדר שבו כותבים את המחלקות בשורה של הירושה (במקרה זה, InputFile ואז OutputFile).

מה קורה אם יש שיטה המופיעה בשתי המחלקות המורישות (עם אותה חתימה בדיוק)?

- במקרה זה הקומפיילר לא יאפשר לנו לקרוא לשיטה הזו - יש כאן אי-בהירות. בדיוק כמו שקורה כשמנסים לגשת לפונקציה מועמסת עם כמה אפשרויות והקומפיילר לא יודע לבחור ביניהן.

פתרון אפשרי הוא לקרוא ישירות לשיטה שאנחנו רוצים, בעזרת אופרטור ארבע נקודות. למשל, אם בשתי המחלקות המורישות יש שיטה בשם open, אנחנו יכולים לבחור בפירוש למי לקרוא:

```
f.InputFile::open();
```

ירושת יהלום וירושה וירטואלית

ירושת-יהלום היא מצב שבו יש מחלקת בסיס (למשל "קובץ"), שתי מחלקות שיורשות אותה (למשל "קובץ קלט", "קובץ פלט"), ומחלקה רביעית שיורשת את שתיהן ("קובץ קלט ופלט").

במחלקה הרביעית הזאת, יש **שני** עותקים של מחלקת הבסיס. זה אומר שכל השדות של מחלקת הבסיס מופיעים פעמיים, וכל השיטות מופיעות פעמיים

שוב נוצרת כאן אי-בהירות שאפשר לפתור כמו קודם - ע"י פניה ישירות לשדה או לשיטה המגיעים מאחד משני הצדדים. למשל, אפשר לשנות את שם הקובץ ע"י שינוי המשתנה

```
f.InputFile::name
```

אבל זה לא ישפיע על המשתנה

```
f.OutputFile::name
```

אלה שני משתנים שונים.

פתרון אחר הוא להגדיר את הירושות של מחלקת הבסיס כוירטואליות, באופן הבא:

```
struct InputFile : virtual public File {
  void read();
};
struct OutputFile : virtual public File {
  void write();
};
```

במקרה זה, במחלקה הרביעית, הקומפיילר ייצר רק עותק **אחד** של מחלקת הבסיס (File). לא תהיה אי-בהירות - יהיה רק עותק אחד מכל שיטה ומכל שדה.

בעיה נוספת בירושת יהלום היא כשבונים את המחלקה הרביעית. באופן טבעי, הבנאי קורא לשני הבנאים של המחלקה השניה והשלישית, וכל אחד מהם קורא לבנאי של מחלקת הבסיס. יוצא שמחלקת הבסיס נבנית פעמיים - שגיאה לוגית.

הפתרון: הבנאי של המחלקה הרביעית צריך לקרוא ישירות לבנאי של מחלקת הבסיס - לפני שהוא קורא לבנאים הקודמים. כך הקומפיילר בונה קודם-כל את מחלקת הבסיס, ואז את שתי המחלקות היורשות - ולא בונה שוב את מחלקת-הבסיס.

ירושה כפולה - כן או לא?

בגלל כל הבעיות הנגרמות ע"י ירושה כפולה, ישנן שפות רבות שאינן מאפשרות זאת, למשל, Java.

בעבר היו ויכוחים סוערים בין תומכי C++ לתומכי Java בשאלה אם ירושה כפולה היא טובה או לא. כיום הדעות מעט התמתנו:

- מצד אחד, מומחים ל-C++ ממליצים להשתמש בירושה כפולה רק כאשר רוב המחלקות המורישות הן וירטואליות-טהורות - בלי שדות. "ירושה כפולה" מסוג זה אפשרית גם בג'אבה - כל מחלקה יכולה לממש הרבה ממשקים.
- מצד שני, ב-Java החל מגירסה 8 יש אפשרות לשים בממשקים שיטות עם מימוש ברירת-מחדל (בעזרת מילת המפתח `default`). כך, כשמחלקה מממשת כמה ממשקים, היא נהנית מחלק מהיתרונות של ירושה כפולה.

סדר בניה

אם יש ירושה וירטואלית במקום כלשהו בהיררכיה - קודם-כל נבנות מחלקות הבסיס הוירטואליות. הן נבנות לפי הסדר שבו הן מופיעות בסיור עומק-תחילה (`depth-first search`), ובכל רמה - לפי סדר הכתיבה בשורת הירושה.

אחרי שכל מחלקות-הבסיס-הוירטואליות בנויות, מתחילים לבנות את שאר המחלקות - מהבסיסיות ליורשות - לפי סדר הכתיבה בשורת הירושה.

שאלה: האם אפשר להשתמש בקומפיילר כדי לפתור בעיות בתורת הגרפים? למשל, נתון גרף ורוצים לעבור עליו בסדר *DFS*. יוצרים עץ מחלקות המשקף את הגרף וקוראים לקומפיילר...

מקורות

- מצגות של אופיר פלא ומירי בן-ניסן.

סיכם: אראל סגל-הלוי.