שפות תכנות, 234319

חורף 2018-2019

תרגיל בית 4

חלק יבש

קראו על סוגי פולימורפיזמים השונים, וענו על הסעיפים הבאים:
על כל הסעיפים בהם צורפו דוגמאות קוד, הקוד נבדק ב-Online D Editor, שהוא מגרש המשחקים הרשמי של השפה.

1. אילו סוגי פולימורפיזם קיימים בשפת D? תנו דוגמה לכל אחד מהסוגים.
תשובה:

<u>פולימורפיזם אד הוק</u>

- Overloading - ניתן למשל להעמיס פונקציות. דוגמת קוד:

```
import std.stdio;

// function overloading
int foo(int i) {
    return i;
}

double foo(double i) {
    return i;
}


void main() {
    double almostPi = 3.14;
    int zero = 0;

    writeln("This time I print double: ", foo(almostPi));
    writeln("And this time int: ", foo(zero));
}
```

תוצאת הרצה:

```
> rdmd playground.d

This time I print double: 3.14
And this time int: 0
```

- Coercion - ניתן לבצע "כפייה". דוגמת קוד:

```
// coercion
import std.stdio;

double foo(double i) {
    return i;
}


void main() {
    double almostPi = 3.14;
    int zero = 0;

    writeln("This time I print double: ", foo(almostPi));
    writeln("And this time int: ", foo(zero));
}
```

ת

ת

תוצאת הרצה (בדומה לדוגמה הקודמת):

```
> rdmd playground.d
```

```
This time I print double: 3.14
And this time int: 0
```

פולימורפיזם אוניברסלי

- Polytype - ניתן לבצע polytype בעזרת "תבניות". דוגמת קוד לתבנית של פונקציה המשמשת להחזרה של ארגומנט מכל טיפוס. בדוגמה הפונקציה נקראת פעם עם טיפוס מספר שלם ופעם שנייה עם טיפוס מערך של מספרים שלמים:

```
// parametric / polytype
import std.stdio;

T returnGeneric(T)(T x)
{
    return x;
}
void main()
{
    writeln("Now I print int: ", returnGeneric!(int)(5));
    writeln("But I can also print any printable type," ~
            "such as arrays: ", returnGeneric!(int[])([1,2,3,4]));
}
```

תוצאת הרצה:

```
> rdmd playground.d
```

```
Now I print int: 5
But I can also print any printable type,such as arrays: [1, 2, 3, 4]
```

- Inclusion - ניתן לבצע inclusion בשפת D. למעשה, לפי התיעוד הרשמי, אם לא הוגדר אחרת, כל מחלקה שנוצרת יורשת ממחלקת Object.
להלן דוגמת קוד עם מחלקות בשם A,B,C, כאשר B ו-C יורשות מ-A ומכילות מטודות הדפסה נפרדות משלהן. בנוסף, יש מערך המכיל טיפוסים ממחלקות B ו-C, כאובייקטים מסוג A (ניתן לעשות את זה כי B ו-C הם "סוג של" A) ואז הדפסה של האובייקטים שנוצרו בלולאה, כך שכל מופע של אובייקט קורא למטודת הדפסה המתאימה לו, שזה בעצם dynamic binding שמתאפשר בזכות מנגנון הפולימורפיזם בשפה (הדוגמה כביכול ארוכה, אבל פשוטה מאוד):

```
// inclusion / inheritance
import std.stdio : writeln;

class A {
    // protected is just seen by inheriting classes
    protected string type;

    this(string type) {
        this.type = type;
    }

    abstract void print();
}

class B : A {
    private {
        int number;
    }

    // constructor
    this(int number) {
        // call base class constructor
        super("B");
        this.number = number;
    }

    override void print() {
        writeln("I'm a B object and my number is: ", number);
    }
}

class C : A {
    private {
        char c;
    }

    // constructor
    this(char c) {
        // call base class constructor
        super("C");
        this.c = c;
    }

    override void print() {
        writeln("I'm a C object any my char is: ", c);
    }
}

void main()
{
    A[] anys = [
        new B(10),
        new C('D')
    ];

    foreach (any; anys) {
        any.print();
    }
}
```

תוצאה הרצה:

```
> rdmd playground.d
```

```
I'm a B object and my number is: 10
I'm a C object any my char is: D
```

2. האם השפה מאפשרת למתכנת להעמיס (overload) אופרטורים? אם כן, תנו דוגמת קוד קצרה.

תשובה: כן, ניתן. דוגמת קוד של העמסת האופרטור האונרי מינוס על אובייקט מסוג class S:

```
// operator overloading
import std.stdio;

class S {
    int m;

    // constructor
    this(int m) { this.m = m; }

    // unary operator minus
    S opUnary(string s)() if (s == "-") {
        return new S(-m);
    }
}

void main() {
    S s = new S(5);
    S s_minus = -s;

    writeln("This is what s is: ", s.m);
    writeln("But with op- we get: ", s_minus.m);
}
```

תוצאת הרצה:

```
> rdmd playground.d
```

```
This is what s is: 5
But with op- we get: -5
```

3. האם השפה מאפשרת למתכנת להעמיס שמות של פונקציות ופרוצדורות? אם כן, תנו דוגמת קוד קצרה.

תשובה: ניתן להעמיס פונקציות לפי האתר הרשמי. נתתי דוגמת קוד כזו בסעיף 1 עבור overloading.

לעומת זאת, לא ניתן להעמיס פרוצדורות, כי באופן כללי פונקציות נבדלות אחת מהשנייה על ידי המזהה שלהן. אם המזהים זהים, אז מתבצעת התאמה לפי טיפוסי הארגומנטים וערך החזרה (או ההתאמה הטובה ביותר באמצעות coercion, אם קיים). היות ובפרוצדורות אין ערך חזרה ואין ארגומנטים, אין דרך להבדיל בין שתי פרוצדורות בעלות אותו מזהה.

חלק רטוב

תרגיל 1 - מילון

בתרגיל זה נגדיר טיפוס חדש עבור מבנה הנתונים מילון, והפעולות עליו. מילון או, מערך אסוציאטיבי, הוא מבנה נתונים המכיל קישורים בין מפתחות לערכים. כלומר בהינתן מילון d ומפתח k - ייתכנו בדיוק שתי אפשרויות: (1) המפתח k לא קיים במילון d או (2) הערך עבור המפתח k במילון d הוא v.

לאורך הדוגמאות יהיה שימוש באופרטור -- לשם יצירת זוגות של מפתח וערך, את הגדרתו יש לכלול בפתרון:

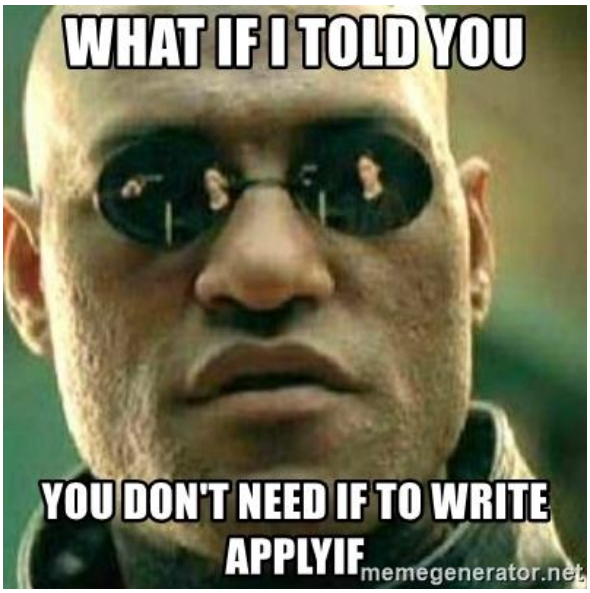
```
infix 2 --;
fun key -- value = (key, value);
```

1. השלימו את הגדרת הטיפוס, וכללו אותה בקובץ הפתרון שלכם:

```
datatype ('a, 'b) dictionary = dict of ('a * 'b) list;
exception ItemIsNotPresent;
```

אתנחתא

הנה מם שהיה אמור להיכנס בתרגיל הקודם, אבל נשכח:



והנה אחד אקטואלי יותר: