יום שישי 16 מאי 2008

תכנות מונחה עצמים (236703)

מבחן סוף סמסטר– סמסטר חורף 2007-2008, מועד א

מרצה : פרופ'/ח' יוסי גיל

מתרגלים : קרן לנץ, יבגניה אלפרין

הנחיות :

1. במבחן 6 שאלות שאינן שוות משקל וקושי. עליכם לענות על כל השאלות.
2. בכל שאלה, אם אינכם יודעים את התשובה, כתבו "לא יודע/ת" ותקבלו 20% מהנקודות.
3. ענו לענין. הוספת מידע לא רלוובנטי בעליל (אף אם הוא נכון) תפגע בציון הניתן על התשובה, ובמקרים קיצוניים אף למתן ניקוד שלילי.
4. כתבו תשובתכם בכתב ברור
5. התחילו כל תשובה בעמוד חדש.
6. אין להשתמש בכל חומר עזר.
7. במבחן 5 עמודים (כולל עמוד זה) וודאו כעת שקיבלתם את כולם.
8. משך הבחינה שעתיים וחצי. לא תינתן הארכה.

בהצלחה

שאלה 1 – 20 נקודות

שפת התכנות Sava הינה שפה הדומה לJava מבחינת תחביר, אך שונה מבחינת סמנטיקה בהיבטים הבאים: השפה היא structurally typed וחוקי התאימות שלה מאפשרים דריסת מתודה תוך שינוי קו-וריאנטי של טיפוס החזרה ושינוי קונטרה-וריאנטי של טיפוסי הפרמטרים.

א. נתונים הinterface והמחלקה הבאים:

```
interface I{
    I clone();
}

class C{
    C clone(){…}
}
```

האם ההשמה הבאה חוקית? נמק. אם ההשמה אינה חוקית תקן את הקוד כך שההשמה תהיה חוקית

```
I i = new C();
```

ב. נתונים הinterface והמחלקה הבאים:

```
interface I{
    boolean equals(I i){…}
}

class C{
    boolean equals(C c){…}}
```

האם ההשמה הבאה חוקית? נמק. אם ההשמה אינה חוקית תקן את הקוד כך שההשמה תהיה חוקית

```
I i = new C();
```

ג. בנוסף, שפת Sava מאפשרת דריסת שדות תוך שינוי הטיפוס בצורה קו-וריאנטית. אילו מבין התכונות הבאות חייבות להתקיים ע"י השדות על מנת לשמור על שלמות מערכת הטיפוסים?

a. כל השדות הינם private
b. כל השדות הינם public
c. כל השדות הינם read only
d. כל השדות הינם write only
e. כל השדות הינם read-write
f. כל השדות הינם non-null

נמק תשובתך.

פתרון

א. 6 נקודות

ההשמה חוקית. המתודה C.clone() *שונה מ*I.clone() *בטיפוס החזרה בלבד, והשינוי הינו קו-וריאנטי.*

תשובות שהתיחסו לכך שהמחלקה C *אינה מוצהרת כ*implements I *לא התקבלו מאחר ובהינתן* structural typing *אין משמעות להכרעה על יחסי* subtyping.

ב. 6 נקודות

ההשמה אינה חוקית. המתודה C.equals() *משנה את טיפוס הפרמטר*

בגורם קו-וריאנטית ולכן אינה תואמת ל()I.equals.

ט. 8 נקודות

בשפת Sava כל השדות חייבים להיות read only. בדוגמא שמתחישה זאת:

```
class C1{
        private Object o;
        public void f() {
                o = new Date();
        }
}

class C2{
        private String o;
        void g(){
                if(o != null)
                        o.charAt(0);
        }

        void main(){
                C2 c2 = new C2();
                C1 c1 = c2;
                c1.f();
                c2.g(); //compiles but results in a runtime error.
        }
}
```

שאלה 2 – 20 נקודות

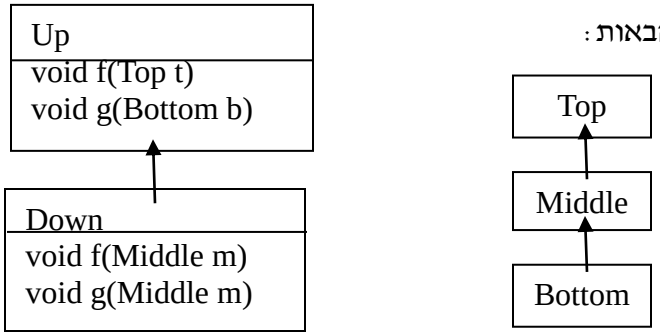
להלן ציטוט מתוך ***The Java Language Specification, Third Edition, section 8.4.9***:

> "If two methods of a class (whether both declared in the same class, or both inherited by a class, or one declared and one inherited) have the same name but signatures that are not override-equivalent, then the method name is said to be *overloaded*."

וציטוט נוסף מתוך ***The C++ Programming Language*** מאת ***Bjarne Stroustrup***:

> "Two function declarations of the same name refer to the same function if they are in the same scope and have equivalent parameter declarations. A function member of a derived class is *not* in the same scope as a function member of the same name in a base class. *Example:*
>
> ```
> class B {
> public:
>  int f(int);
> };
> ```
>
> ```
> class D : public B {
> public:
>  int f(char*);
> };
> ```
>
> Here `D::f(char*)` *hides* `B::f(int)` rather than overloading it."

נתונות היראַרכיות הטיפוסים הבאות:



א. פרש את ההיראַרכיות כאילו הן בשפת Java. נתונה הפונקציה הבאה:

```
1  void bar(){
2     Down d = new Down();
3     d.f(new Top());
4     d.f(new Bottom());
5     d.g(new Bottom());
6  }
```

לכל אחת מהקריאות בשורות 3-5, ציין מהי חתימת הפונקציה שתקרא (במידה ושורה מסוימת תגרום לשגיאת קומפילציה יש לציין זאת ולהתייחס לשורות האחרות).

ב. פרש את ההיראַרכיות כאילו הן בשפת C++ ועיין בשכתוב הפונקציה bar ב-C++:

```
1  void bar(){
2     Down* d = new Down();
3     d->f(new Top());
4     d->f(new Bottom());
5     d->g(new Bottom());
6  }
```

לכל אחת מהקריאות בשורות 3-5, ציין מהי חתימת הפונקציה שתקרא (במידה ושורה מסוימת תגרום לשגיאת קומפילציה יש לציין זאת ולהתייחס לשורות האחרות).

פתרון

א. שורה 3 (נקרא מן) – Up.f

שורה 4 (נקרא מן) – Down.f

שורה 5 (נקרא מן) – Up.g

ב. שורה 3 (נקרא מן) – compilation error

שורה 4 (נקרא מן) – Down.f

שורה 5 (נקרא מן) – Down.g

שאלה 3 – 15 נקודות

א. נניח כי בשפה בעלת תחביר הדומה לlittle smalltalk אשר ממומשת כמערכת בת 5 רמות, המחלקה Z יורשת מY אשר יורשת מ X.
מה יהיו תוצאות הביטויים הבאים? נמק תשובותיך.

```
Z class class class class
Z new class superClass class superClass
```

ב. נניח כעת כי השפה ממומשת כמערכת בת 3 רמות. מה יהיו כעת תוצאות הביטויים?

פתרון

א. ביטוי ראשון (4 נקודות): Meta-class *(הוא הרמה הרביעית בהיררכיה). הסבר:*

Z class = Z-class
Z class class = meta-class
Z class class class = meta-class-class
Z class class class class = meta-class

ביטוי שני (5 נקודות): X-class. *הסבר:*

Z new class = Z
Z new class superClass = Y
Z new class superClass class = Y-class
Z new class superClass class superClass = X-class

ג. ביטוי ראשון (3 נקודות): Class. *הסבר:*

Z class = Class
Z class class = Class
Z class class class = Class
Z class class class class = Class

ביטוי שני (3 נקודות): Object. *הסבר:*

Z new class = Z
Z new class superClass = Y
Z new class superClass class = Class
Z new class superClass class superClass = Object

ד.

שאלה 4– 10 נקודות

נתונה המחלקה הבאה (++C):

```
class Conjunction {
      list<char>* literals;
public:
      Conjunction(char l){
            literals = new list<char>();
            literals->push_front(l);
      };
      ~Conjunction() {
            delete literals;
      }
      bool compare(Conjunction c) const {
            return literals == c.literals;
      }
};
```

ונתונה התכנית הבאה:

```
int main() {
```

```
        Conjunction ca('a');
        Conjunction cb('b');
        cout << ca.compare(cb) << endl;
}
```

א. הסבר את מקור שגיאת זמן הריצה שתתקבל עבור התכנית.
ב. שנה את הקוד כדי לתקן את בעיה זו.

פתרון

א. 7 נקודות

מקור הבעיה: הפרמטר לcompare() מתקבל by value, ולכן כאשר הפונקציה נקראת מופעל copy constructor שמעתיק את cb לאובייקט זמני. מאחר ולא מוגדר copy constructor במחלקה, מופעל default copy constructor אשר מבצע העתקה שטחית, כלומר, השדות literal של האובייקט הזמני cb מצביעים על אותו אובייקט.

אובייקט זה משוחרר בסיום הפונקציה, עם סיום הscope בו מוגדר האובייקט הזמני והפעלת הdestructor שלו. בסיום פונקציית המain מופעל הdestructor של cb וזה מנסה לשחרר שוב את הזיכרון שכבר שוחרר, ומפה שגיאת זמן הריצה.

ב. 3 נקודות

שינוי הקוד הינו בחתימת הפונקציה compare:

```
bool compare(Conjunction& c) const
```

שאלה 5– 20 נקודות

א. צייר את הגרף המתאר אובייקט ממחלקה D כאשר נתונות ההגדרות הבאות:

```
struct A {
      virtual void f() {cout << "A" << endl;};
};
struct B : virtual A {
      virtual void f() {cout << "B" << endl;};
};
struct C : A {
      virtual void g() {cout << "C" << endl;};
};
struct D : B, C {};
```

ב. בהתייחס למחלקות מהסעיף הקודם, האם התכנית הבאה תעבור קומפילציה? אם כן, ציין מהו הפלט. אם לא, נמק מדוע.

```
int main() {
```

```
        D d;
        d.f();
        return 0;
}
```

ג. האם ניתן להוסיף מתודה אחת לאחת המחלקות כך שתיוצר שגיאת קומפילציה בשורה

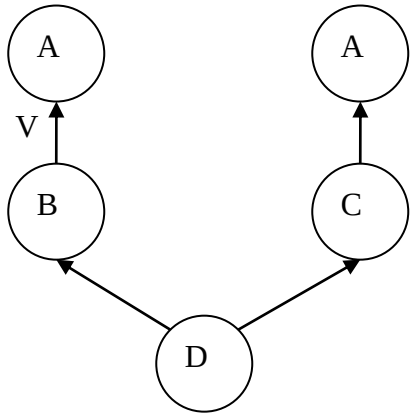
```
struct D : B, C {};
```

אם כן, ציין את שם המחלקה ושם המתודה, אם לא, הסבר מדוע.

פתרון:

א. 5 נקודות

תשובה:



ניקוד:

- *מצוייר לא טרף של אובייקט אלא היררכית ירושה (A משותף) – 1 נקודה קנס*
- *לא סומן v – 2 נקודות קנס*
- *ציור במקום טרף – 1 נקודה קנס. קנס נוסף על טעויות בציור.*

ב. 6 נקודות

תשובה: לא.

יש שתי פונקציות f: בתת אובייקט של C ובתת-אובייקט של B. לכן תהיה שגיאת קומפילציה בשורה d.f() – ambiguity *בקריאה.*

ניקוד:

- *שגיאת זמן ריצה במקום שגיאת קומפילציה – 1 נקודה קנס*

ט. 9 נקודות

תשובה: לא.
שטיאת קומפילציה בשורה זו מופיעה כשאי אפשר לבנות טבלה ויטואלית של אובייקט D.
בשביל זה צריך שיהיו שני מצביעים לטבלה ויטואלית בשדה המתאר את האובייקט, שבכל אחד מהם יש overriding של f (באותה רמה)
אזי קומפיילר לא יודע איזה כתובת של f לשים בטבלה הויטואלית ויש דו-משמעות ביצירת אובייקט.

במקרה שלנו לאף טבלה וויטואלית אין שני מצביעים (לB ולC יש תת-אובייקט של A נפרד ולכן טבלה וויטואלית נפרדת). לכן אי אפשר ליצור שטיאת קומפילציה מסוג זה.

שאלה 6– 15 נקודות

להלן מספר קריאות בLittle Smalltalk והפלט שלהן:

```
> True respondsTo: #not
false
> True respondsTo includesKey: #not
true
> True respondsTo: #print
true
> True respondsTo includesKey: #print
true
```

א. הסבר את השוני במימוש של שתי המתודות שגורם לשוני בפלט.
שים לב: המתודה not שייכת למחלקה True

ב. הראה שתי דרכים שונות (בעזרת `respondsTo:` ובעזרת `respondsTo`) לבדוק האם הפעלת מתודה `select:` על אובייקט מסוג `List` הינה חוקית.

פתרון:

א. 9 נקודות

תשובה:

*מתודה `respondsTo` מוגדרת במחלקה `Class`, ע"א היא צריכה להיות מופעלת על אובייקט מסוג **מחלקה** ופלט שלה הינו `Dictionary` של*

מתודות המוגדרות במחלקה בו ובאבות שלה.

מתודה respondsTo: מוגדרת במחלקה Object, כ"א היא צריכה להיות מופעלת על **אובייקט** והפעם שלה הינו true אם האובייקט יכול לקבל את ההודעה שמוגדרת ל-respondsTo: כפרמטר (או במילים אחרות אם מתודה המוגדרת כפרמטר מוגדרת במחלקה של האובייקט או באחד האבות שלה).

הסבר של דוגמאות:

```
> True respondsTo: #not
false
```

respondsTo: מופעל על אובייקט ממחלקה Class, ברור שבמחלקה Class אין מתודה #not, לכן פלט הינו false.

```
> True respondsTo includesKey: #not
true
```

מחלקה True מכילה מתודה #not, לכן פלט הינו true.

```
> True respondsTo: #print
true
```

respondsTo: מופעל על אובייקט ממחלקה Class, מחלקה Class יורשת ממחלקה Object בה מתודה print מוגדרת לכן פלט הינו true.

```
> True respondsTo includesKey: #print
true
```

מחלקה True יורשת מתודת print מ-Object, לכן פלט הינו true.

ג. 6 נקודות (respondsTo: 4, respondsTo 2)

<u>תשובה:</u>

```
List respondsTo includesKey: #select:
List new respondsTo: #select:
```