מבוא לתכנות מונחה עצמים

מועד א'

ד"ר אופיר פלא, גברת אליזבט איצקוביץ

תשע"ד סמסטר א'

משך הבחינה:

הנחיות:

1. ניתן להעזר בכל חומר כתוב / מודפס או מצולם.
2. ענו על כל השאלות
3. אסור להשתמש בכלי חישוב כלשהוא.
4. אין להשתמש בעט אדום.
5. אם ברצונכם לבטל שאלה, חלק ממנה, או טיוטה, מחקו באופן ברור שאינו משאיר מקום לפירושים.
6. הקפידו על סגנון תכנותי נאות: כתבו באופן מודולרי ,אל תכפילו קוד וכו'.
7. תעדו כל תכנית כך שניתן יהיה להבינה בנקל (מותר לתעד בעברית).
8. פתרון מיטבי לשאלה יחשב פתרון פשוט ויעיל ככל האפשר .תוכנית מסורבלת או מסובכת לא תתקבל כתשובה מלאה, גם אם היא נכונה!

שאלות

1. (1 נקודה) תן הגדרה לאובייקט.
2. (1 נקודה) תן הגדרה ל-polymorphism.
3. (1 נקודה) מהו ההבדל בין interface לבין abstract class?
4. (1 נקודה) מהי משמעות המילה this בג'אווה?
5. (1 נקודה) הסבר מושג של serialization.

6. (36 נקודות) כתבו קוד שישמש למשחק תפקידים. במשחק יש גיבורים (Hero) שיכולים להיות או לוחמים (Warrior) או קוסמים (Magician).
לכל הגיבורים יש "נקודות חיים", מספר שלם (יכול להיות שלילי) אשר אפשר לקבלו ע"י השיטה lifePoints().
כל גיבור יכול לתקוף גיבור אחר ע"י השיטה attack.
ללוחם יש בנאי שמקבל את נקודות החיים שלו כמספר שלם ראשון ואת נקודות התקיפה שלו כמספר שלם שני.
כאשר לוחם תוקף גיבור אחר הוא מוריד מנקודות החיים של האחר את נקודות התקיפה שלו.
לקוסם יש בנאי שמקבל את נקודות החיים שלו כמספר שלם ראשון ואת נקודות הקסם שלו כמספר שלם שני.
כאשר קוסם תוקף הוא מחלק את מספר נקודות החיים של האחר במספר נקודות הקסם שלו (חילוק שלמים).
בלוגיקה של המשחק לא צריך להיות מצב שגיבור יתקוף את עצמו. אם זה קורה, אז זה באג בתוכנה. עליכם לבדוק זאת ובמידה שזה קורה עליכם לזרוק `SelfAttackException`. עליכם גם לרשום את המחלקה SelfAttackException ושהיא תירש מ RuntimeException.
שימו לב: כמו בכל שאר השאלות עליכם לכתוב קוד נקי וקל להרחבה ושומר בפני טעויות ככל הניתן.
להלן דוגמת main שצריך להתקמפל עם הקוד שלכם והפלט שהוא צריך לתת.
הקוד:

```
public static void main(String args[]) {
      Hero heroes[]= new Hero[3];
      heroes[0]= new Warrior(10,5);
      heroes[1]= new Warrior(15,2);
      heroes[2]= new Magician(15,2);

      for (int i=0; i<heroes.length; ++i) {
            for (int j=i+1; j<heroes.length; ++j) {
                  heroes[i].attack(heroes[j]);
                  heroes[j].attack(heroes[i]);
            }
      }

      for (int i=0; i<heroes.length; ++i) {
            System.out.println(heroes[i].lifePoints());
      }

      try {
            heroes[0].attack(heroes[0]);
      } catch (SelfAttackException e) {
            System.out.println("We successfully prevented the first
hero to attack himself");
      }

}
```

הפלט:

```
4
5
8
We successfully prevented the first hero to attack himself
```

7. (9 נקודות) נתון הקובץ A.java הבא:

```
1) class A {
2)    private int _x;
3)    public A() {
4)          _x= 0;
5)    }
6)    public int x() {
7)          return _x;
8)    }
9)    private class B {
10)         B() {_x= 42;}
11)   }
12)   void foo() {B b= new B();}
13) }
```

כמו כן, נתון קובץ Main.java הבא:

```
14) public class Main {
15)   public static void main(String args[]) {
16)         A a= new A();
17)         System.out.println(a.x());
18)         a.foo();
19)         System.out.println(a.x());
20)   }
}
```

הקבצים מקומפלים (הקמפול תקין) והתוכנית רצה.
מה יהיה פלט התוכנית ?
באילו שורות נקרא בנאי ?
באילו שורות מוגדר בנאי ?

8. (50 נקודות) כתבו מחלקה גנרית MyArrayGraph . פונקציונאליות המחלקה (שימו לב שסעיף ז אתם לא צריכים לממש, אתם מניחים כי הוא קיים):
א. בנאי חסר פרמטרים שבונה גרף ריק.
ב. פונקציה addVertex המוסיפה קודקוד לגרף. האינדקס של הקודקוד הוא מספר הפעמים שהשתמשו ב addVertex לפני הקריאה הנוכחית (כלומר, 0 לאיבר הראשון שמוסף, 1 לבא אחריו,...).
ג. פונקציה getVertex מקבלת אינדקס של קודקוד ומחזירה את המידע שלו.
ד. מחלקה פנימית MyArrayGraphEdge אשר מייצגת צלע כיוונית בגרף ומכילה אינדקס from ואינדקס to עם בנאי שמקבל את 2 הפרמטרים הללו.
ה. פונקציה hasEdge אשר בודקת אם אובייקט MyArrayGraphEdge כבר נמצא בגרף. שימו לב: הכוונה לבדוק לא שיוויון בין רפרנסים אלא לבדוק שיש אובייקט זהה לוגית בגרף (ראו גם שימוש למטה).
ו. פונקציית addEdge אשר מוסיפה צלע לגרף. אם הצלע כבר קיימת בגרף הפונקציה לא עושה כלום.
ז. הניחו כי קיימת פונקציה ()toString המחזירה ייצוג String של המחלקה כך שדוגמת main למטה עובדת כראוי. אין צורך לכתוב את הפונקציה.
ח. אין צורך לבדוק תקינות קלט אשר לא הוגדרה בשאלה (לדוגמא, אין צורך לבדוק אינדקסי הקודקודים הניתנים הם חוקיים).
ט. מותר ואף רצוי להשתמש במחלקה Vector בשיטות שמפורטות בסוף המבחן עם התיעוד שלהם.
י. לשם הפשטות אתם יכולים להניח שלכל שדה שלכם יש פונקציות get ו- set. אין צורך לכתוב אותם אך אתם יכולים להשתמש בהם.
להלן קובץ ה main והפלט הנדרש:

```
public class Main {
      public static void main(String args[]) {
            MyArrayGraph<String> g= new MyArrayGraph<>();
            g.addVertex("C");
            g.addVertex("C++");
            g.addVertex("Java");
            g.addVertex("C#");
            g.addVertex("Lisp");
            g.addEdge(g.new MyArrayGraphEdge(0, 1));
            g.addEdge(g.new MyArrayGraphEdge(0, 2));
            // should do nothing
            g.addEdge(g.new MyArrayGraphEdge(0, 2));
            g.addEdge(g.new MyArrayGraphEdge(2, 3));
            assert( g.hasEdge(g.new MyArrayGraphEdge(0,1)) );
            assert( !g.hasEdge(g.new MyArrayGraphEdge(0,3)) );
```

```
            System.out.println(g);
            System.out.println("------------------");
            System.out.println(g.getVertex(2)); // Java
      }
}
```

הפלט הנדרש:

```
0(C)-> 1(C++), 2(Java)
1(C++)->
2(Java)-> 3(C#)
3(C#)->
4(Lisp)->
------------------
Java
```

שיטות של Vector<E> שאתם יכולים להשתמש בהם ותיעוד שלהם:

```
Vector()
Constructs an empty vector

boolean add(E e)
Appends the specified element to the end of this Vector.
always returns true

E get(int index)
Returns the element at the specified position in this Vector.

int size()
Returns the number of components in this vector.
```

הערה: מי שמכיר syntax של foreach יכול להשתמש בו עם Vector אם זה יותר נוח לו, אך זה לא חובה.

בהצלחה !!!