בחינה בתוכנה 1 (2157 - 0368)
ובתוכנה 1א (2004 - 0368)

ליאור וולף ואורנית דרור

סמסטר ב' תשס"ז
מועד ב' , 8 באוקטובר 2007

משך הבחינה שלוש שעות.

יש לענות על כל השאלות. בשאלות שבהן יש צורך לנמק, תשובה ללא נימוק לא תזכה באף נקודה.

יש לענות על כל השאלות ב**גוף הבחינה במקום המיועד לכך**. המקום המיועד מספיק לתשובות מלאות. יש לצרף את טופס המבחן למחברת הבחינה. מחברת ללא טופס עזר תפסל. תשובות במחברת הבחינה לא תיבדקנה.

יש למלא מספר סידורי (מספר מחברת) ומספר ת"ז על כל דף של טופס הבחינה.

אסור השימוש בחומר עזר כלשהו, כולל מחשבונים או כל מכשיר אחר פרט לעט.

בהצלחה!

לשימוש הבודקים:

1	**א**	**ב**						
2	**א**	**ב**	**ג**					
3	**א**	**ב**	**ג**	**ד**	**ה**	**ו**		
4	**א**	**ב**	**ג**	**ד**	**ה**	**ו**		
						סה"כ		

שאלה 1

בשאלה זו נדון במחלקות שמייצגות מספרים רציונליים, כגון $1/3$ או $99/100$.
תזכורת מתמטית קצרה: כדי לצמצם שברים צריך לחלק את המונה והמכנה בגורם המשותף הגדול ביותר (GCD באנגלית). למשל, הגורם המשותף הגדול ביותר של 42 ו-56 הוא 14, ולכן

$$\frac{42}{56} = \frac{3 \cdot 14}{4 \cdot 14} = \frac{3}{4}.$$

כדי לחבר או לחסר שברים, צריך למצוא את המכנה המשותף (LCM באנגלית). למשל, המכנה המשותף של 21 ו-6 הוא 42, ולכן

$$\frac{2}{21} + \frac{1}{6} = \frac{4}{42} + \frac{7}{42} = \frac{11}{42},$$

בשאלה הזו נניח שהמונה (numerator) והמכנה (denominator) מיוצגים תמיד על ידי שדות או משתנים מטיפוס int. בכל המחלקות שנדון בהן, המונה מיוצג על ידי שדה מופע בשם num והמכנה על ידי שדה מופע בשם denom.

א. המחלקות שמייצגות מספרים רציונלים יממושו תוך שימוש במחלקת עזר MathExtra שמכילה שירותי מחלקה שממממשים אלגוריתם לחישוב מכנה משותף של שני שלמים (שם השירות lcm) ואלגוריתם לחישוב הגורם המשותף הגדול ביותר של שני שלמים (שם השירות gcd).

הגדר/הגדירי את מחלקת העזר הזו, השאר/י את גוף השירותים הללו ריק (כלומר אין צורך לכתוב קוד שממממש את האלגוריתמים הללו).

{ _____

_____

_____

_____

_____

}

ב. הגדירו מחלקה Rational, שזורקת חריג כאשר המכנה מתאפס. החריג שנזרוק יהיה java.lang.ArithmeticException. הניחו שלא קיים אף אובייקט מסוג Rational אשר יש לו מכנה אפס.

הטקסט הבא מועתק מתוך התיעוד של הספריה הסטנדרטית:

java.lang
Class ArithmeticException

java.lang.Object
  └java.lang.Throwable
    └java.lang.Exception
      └java.lang.RuntimeException
        └**java.lang.ArithmeticException**

All Implemented Interfaces:
Serializable

---

public class **ArithmeticException**
extends RuntimeException

Thrown when an exceptional arithmetic condition has occurred. For example, an integer "divide by zero" throws an instance of this class.

Since:
JDK1.0

הגדר/י את המחלקה ואת החריגים שהיא זורקת.

```
public class Rational {
  private int denom;
  private int num;



  public Rational (int denom, int num) {

  _____

  _____

  }
```

```
public void add(Rational r) {
_____
_____
_____
_____
_____
}

public void divideBy(Rational r) {
_____
_____
_____
_____
_____
_____
}
```

```
public void multiply(Rational r) {
_____
_____
_____
_____
_____
}

public boolean equals(Rational r) {
_____
_____
_____
_____
_____
_____
  }
}
```

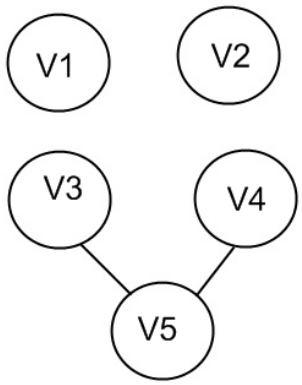
שאלה 2

בשאלה זו נממש מחלקה אבסטרקטית של גרפים לא מכוונים (להלן גרף). גרף הוא מבנה שיש לו קבוצת קודקודים V וקבוצת קשתות E. לצורך השאלה נניח שכל קודקוד הוא מופע של מחלקה בשם Vertex (ראו למטה), וכל קשת היא זוג לא סדור Pair של קודקודים.

בדוגמא זו:

V = { V1,V2,V3,V4,V5 }

E = { {V3,V5}, {V5, V4} }



המחלקה לייצוג קודקוד מוגדרת בפשטות:

```
public class Vertex{}
```

א. הגדירו מחלקת עזר גנרית (generic class) בשם Pair אשר מתאימה לייצג זוגות <u>לא סדורים</u> של עצמים מאותו טיפוס. על המחלקה לכלול:
   1. בנאי המקבל שני עצמים מאותו טיפוס
   2. מתודה toArray המחזירה מערך בן שני אברים המכיל את אברי הזוג
   3. מתודת equals להשוואה לזוג לא סדור אחר (השוואה מבוססת תוכן)

ב. הגדירו מחלקה אבסטרקטית של גרף, Graph, על המחלקה לכלול:
1. מתודה אבסטרקטית vertices אשר מחזירה את אוסף כל הקודקודים בגרף.
2. מתודה אבסטרקטית neighbors אשר בהינתן קודקוד קיים מחזירה את אוסף כל הקודקודים בגרף אשר יש קשת בינם ובין הקודקוד הנתון.
3. מתודה connected שאינה אבסטרקטית אשר מקבלת כארגומנט קודקוד ומחזירה את אוסף הקודקודים אשר יש מסלול מקודקוד זה אליהם. על מתודה זו להיות יעילה במובן שאסור לה לקרוא למתודה neighbors של קודקוד מסוים יותר מפעם אחת.

לנוחותכם, להלן תיעוד המנשק Set:

java.util
Interface Set<E>

All Superinterfaces:
Collection<E>, Iterable<E>
Some Implementing Classes:
HashSet, LinkedHashSet, TreeSet

```
public interface Set<E>
extends Collection<E>
```

A collection that contains no duplicate elements…

Method Summary	
boolean	**add**(E o) Adds the specified element to this set if it is not already present (optional operation).
boolean	**addAll**(Collection<? extends E> c) Adds all of the elements in the specified collection to this set if they're not already present (optional operation).
void	**clear**() Removes all of the elements from this set (optional operation).
boolean	**contains**(Object o) Returns true if this set contains the specified element.
boolean	**containsAll**(Collection<?> c) Returns true if this set contains all of the elements of the specified collection.
boolean	**equals**(Object o) Compares the specified object with this set for equality.
int	**hashCode**() Returns the hash code value for this set.
boolean	**isEmpty**() Returns true if this set contains no elements.
Iterator<E>	**iterator**() Returns an iterator over the elements in this set.
boolean	**remove**(Object o) Removes the specified element from this set if it is present (optional operation).

boolean	**removeAll**(Collection<?> c) Removes from this set all of its elements that are contained in the specified collection (optional operation).
boolean	**retainAll**(Collection<?> c) Retains only the elements in this set that are contained in the specified collection (optional operation).
int	**size**() Returns the number of elements in this set (its cardinality).
Object[]	**toArray**() Returns an array containing all of the elements in this set.
<T> T[]	**toArray**(T[] a) Returns an array containing all of the elements in this set; the runtime type of the returned array is that of the specified array.

השתמשו במקום זה כדי לענות על סעיף ב':

ג. ממשו במלואה את המחלקה MatrixGraph שהיא הרחבה של Graph. המחלקה מקבלת בבנאי שלה מטריצה ריבועית, משולשית עליונה עם אלכסון אפס, כמערך דו מימדי של int (אין צורך לבדוק זאת). הגרף המתואר ע"י המטריצה הוא הגרף בו יש קודקוד המשויך לכל שורה במטריצה, וקשת לכל איבר במטריצה אשר שונה מאפס. כך, למשל, אם בשורה i ובעמודה j ישנו ערך שונה מאפס אזי יש קשת (i,j) בגרף.

לדוגמא, המטריצה הזו:

0	3	0	-5
0	0	0	19
0	0	0	0
0	0	0	0

וגם זו:

0	23	0	54
0	0	0	1092
0	0	0	0
0	0	0	0

הן שתיים מהמטריצות המשויכות לגרף הבא:

V = { V1,V2, V3, V4 }
E = { {V1,V2}, {V1,V4}, {V2,V4} }

(שימו לב, עליכם לאתחל את הקודקודים בעצמכם, שמות המופעים הינם שרירותיים)

שאלה 3

נתונות המחלקות הבאות:

```
public class Base {
	protected int i = 0;
	public Base(int i) { this.i = i; }
	public Base(Base b) { this(b.i); }
	public Base foo() { return new Base(this); }
}

public class Sub extends Base {
	public Sub(int i) { super(i); }
	public Sub(Sub s) { super(s.i * 2); }
	public Base foo() {    return this; }
	public boolean equals(Object o) { return ((Sub) o).i == this.i; }

	public static void main(String[] args) {
		Base b1 = new Base(1);
		Base b2 = b1;
		Base b3 = b2.foo();
		Base b4 = new Sub(1);
		Base b5 = b4.foo();

		// HERE
	}
}
```

עבור כל אחת משורות הקוד הבאות בדקו האם ניתן להגדירה בפונקצית ה-main של המחלקה Sub במקום ההערה // HERE. סמנו את האפשרות המתאימה: האם תהיה שגיאת קומפילציה (אם כן, ציינו מה השגיאה), או שתהיה תעופה בזמן ריצה (אם כן, ציינו מאיזה סיבה) או שהקוד ירוץ בצורה תקינה (אם כן, ציינו מה יהיה פלט ההדפסה). נמקו בקצרה.

א.

```
System.out.println(b1 == b3);
```

שגיאת קומפילציה (מהי?) / תעופה בזמן ריצה (מה הסיבה) / קוד תקין (מה מודפס)

_____

_____

ב.

```
System.out.println(b1.equals(b3));
```

שגיאת קומפילציה (מהי?) / תעופה בזמן ריצה (מה הסיבה) / קוד תקין (מה מודפס)

_____

_____

ג.

```
System.out.println(b1.equals(b4));
```

שגיאת קומפילציה (מהי?) / תעופה בזמן ריצה (מה הסיבה) / קוד תקין (מה מודפס)

_____

_____

ד.

```
System.out.println(b4.equals(b1));
```

שגיאת קומפילציה (מהי?) / תעופה בזמן ריצה (מה הסיבה) / קוד תקין (מה מודפס)

_____

_____

ה.

```
System.out.println(b5 == b4);
```

שגיאת קומפילציה (מהי?) / תעופה בזמן ריצה (מה הסיבה) / קוד תקין (מה מודפס)

_____

_____

ו.

```
System.out.println(b5.equals(b4));
```

שגיאת קומפילציה (מהי?) / תעופה בזמן ריצה (מה הסיבה) / קוד תקין (מה מודפס)

_____

_____

שאלה 4

נתונות המחלקות הבאות הנמצאות באותה חבילת תוכנה:

```
public class Base {
   protected String bar = "Base";
   Base (String s) { bar = s; }
   void foo() {}
}


public class Sub extends Base {
        // HERE
}
```

עבור כל אחת משורות הקוד הבאות בדקו האם ניתן להגדירה במחלקה Sub במקום ההערה // HERE. נמקו.

א.

```
public String bar = "Sub";
```

_____

_____

ב.

```
private String bar = "Sub";
```

_____

_____

ג.

```
Sub (String s) { bar = s; }
```

_____

_____

ד.

```
Sub (String s) {
        super();
        bar = s;
}
```

_____

_____

ה.

public void foo() {}

_____

_____

ו.

private void foo() {}

_____

_____

ושוב, בהצלחה!