

**פתרון הבחינה בתוכנה 1 (2157 - 0368)**  
**ובתוכנה 1א (2004 - 0368)**

סיון טולדו, אוהד ברזילי, מיכל עוזרי-פלאטו, ליאור שפירא

סמסטר א' תשס"ז  
 מועד ג', 21 בדצמבר 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 בגוף 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 מחשבוניו או כל מכשיר אחר פרט לעט. בהצלחה!

לשימוש הבודקים:

|     |      |     |     |      |      |      |     |   |
|-----|------|-----|-----|------|------|------|-----|---|
| 31  |      |     |     |      | 5 ג  | 20 ב | 6 א | 1 |
| 31  |      |     |     | 10 ד | 10 ג | 5 ב  | 6 א | 2 |
| 18  |      | 3 ו | 3 ה | 3 ד  | 3 ג  | 3 ב  | 3 א | 3 |
| 20  |      |     |     |      |      |      | 20  | 4 |
| 100 | סה"כ |     |     |      |      |      |     |   |

במבחן 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

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

### שאלה 1, 31 נקודות

א. בין המתכנתים ביל ואומה התפתח ויכוח האם **טרם** לכתובת המחלקה המייצגת מספרים רציונליים יש לכתוב **מנשק** המתאר את הפונקציונליות של המחלקה. ביל טען כי המחלקה פשוטה דיה וכי אין הצדקה לתאר את אותו הדבר גם ע"י מנשק וגם ע"י מחלקה. ציינו 3 טיעונים **נגדיים** המצדדים בשימוש במנשק. נמקו או הדגימו בקצרה את טיעוניכם:

1. כתיבת מנשק מאפשרת להפריד את הדיון על השרותים הנחוצים במנשק והחזרה שלהם ממימוש הטיפוס
2. כתיבת מנשק מאפשרת לפתח רכיבים המשתמשים בטיפוס החדש במקביל למימוש הטיפוס החדש
3. כתיבת מנשק מאפשרת גמישות בהחלפת מימוש עתידי ללא שינוי קוד לקוח המשתמש בשרותים אלו
4. שימוש במנשק ממזער את התלות בין חלקים שונים של המערכת היוצר מערכת: קלה יותר להבנה, בעלת הסתרת מידע טובה יותר, קלה לתחזוקה ושינויים ומתקמפלת במהירות

ב. לאחר שביל השתכנע בנחיצות המנשק, החל דיון בינו ובין אומה על נחיצות השרותים הבאים. עבור כל אחד מהם ציינו האם יש לו מקום במנשק, במחלקה, בשניהם או באף אחד מהם. נמקו בקצרה:

- add - לחיבור שני מספרים רציונליים

זוהי פעולה יסודית וטבעית עבור מספרים רציונליים – צריכה להופיע גם במנשק וגם במחלקה

- subtract – לחיסור שני מספרים רציונליים

זוהי פעולה יסודית וטבעית עבור מספרים רציונליים – צריכה להופיע גם במנשק וגם במחלקה

- multiply - להכפלת שני מספרים רציונליים

זוהי פעולה יסודית וטבעית עבור מספרים רציונליים – צריכה להופיע גם במנשק וגם במחלקה

- divide – לחלוקת שני מספרים רציונליים

זוהי פעולה יסודית וטבעית עבור מספרים רציונליים – צריכה להופיע גם במנשק וגם במחלקה

- equals - להשוואת שני מספרים רציונליים

זוהי פעולה יסודית עבור כל טיפוס ב Java המופיעה במחלקה Object ולכן היא אינה צריכה להופיע במנשק אלא רק במחלקה (מימוש שידרוס את מימוש ברירת המחדל)

- gcd – לחישוב הגורם המשותף הגדול ביותר של שני שלמים

זוהי פעולה יסודית עבור שלמים ולא רציונליים ולכן היא אינה צריכה להופיע במנשק או במחלקה. מקומה הטבעי הוא כפונקציה סטטית במחלקת עזר לשרותים מתמטיים.

- lcm – לחישוב מכנה משותף של שני שלמים

זוהי פעולה יסודית עבור שלמים ולא רציונליים ולכן היא אינה צריכה להופיע במנשק או במחלקה. מקומה הטבעי הוא כפונקציה סטטית במחלקת עזר לשרותים מתמטיים.

- normalize – להבאת שבר פשוט למצב מצומצם

זוהי פעולה עבור שברים – ייצוג רציונלי כשבר פשוט נתונה לבחירת המממש (ניתן לייצג מספר רציונלי גם כשבר עשרוני למשל). לכן היא אינה צריכה להופיע במנשק. פעולה זו עשויה להופיע במחלקה, אולי כשרות פרטי, כתלות בבחירת הייצוג הפנימי.

- toString – לייצוג המספר כמחרוזת של שבר פשוט (למשל לצורכי הדפסה)

זוהי פעולה יסודית עבור כל טיפוס ב Java המופיעה במחלקה Object ולכן היא אינה צריכה להופיע במנשק אלא רק במחלקה (מימוש שידרוס את מימוש ברירת המחדל)

- toStringDecimal – לייצוג המספר כמחרוזת של שבר עשרוני (למשל לצורכי הדפסה)

בשונה מ- toString השרות הזה אינו קיים במחלקה Object ולכן צריך להופיע גם במנשק וגם במחלקה (משיקולי סימטריה היה רצוי להוסיף עוד שרות toStringFraction שימומש ע"י toString)

- ג. מה לגבי השרותים: getNumerator ו- getDenominator להחזרת המונה והמכנה של השבר בהתאמה? ביל ואומה הסכימו כי הדבר תלוי בהקשרי השימוש של הטיפוס החדש. ציינו באילו הקשרים יש הצדקה להכללת השרותים במנשק ובאילו אין. נמקו או הדגימו את תשובתכם:

הכללת השרותים getNumerator ו- getDenominator תלויה בצורך של לקוח המחלקה.

לדוגמא: יישום מתמטי אשר מעוניין לבצע חישובים על מספרים רציונלים אינו "מעוניין" בייצוג הפנימי שלהם אלא רק בתוצאה. מאידך יישום גרפי אשר מעוניין לצייר רציונלים על המסך (כדוגמת equation או TeX) צריך לדעת בנפרד את ערכם של המונה או המכנה.

## שאלה 2, 31 נקודות

ביל ואומה מתלבטים לגבי הקבעון (mutability) של הטיפוס החדש. טיפוס מקובע (immutable) הוא טיפוס שלא ניתן לשנות את ערכי העצמים שלו לאחר שנוצרו. ראינו בתרגול את הטיפוסים String ו- StringBuffer המממשים גרסא מקובעת ושאיינה מקובעת למחרוזות.

א. ציינו את היתרונות שיש לטיפוס רציונלי מקובע ואת היתרונות שיש לטיפוס רציונלי שאינו מקובע:

יתרונות הטיפוס הרציונלי המקובע (immutable):

העצמים משמשים כליטרלים, ניתן להצביע על אותו עצם ממספר מקומות (aliasing, sharing) ללא חשש שאחת ההפניות תשנה את ערכו של העצם.

יתרונות הטיפוס הרציונלי שאינו מקובע (mutable):

סכנון בזכרון: אין צורך לייצר עצם חדש עבור כל שינוי במצבו של הטיפוס.

ב. באיזה מקרה יהיה ההבדל בין מימוש השרות equals במחלקות מקובעות לעומת מימושו במחלקות שאינן מקובעות? הסבירו במילים, אין צורך לכתוב קוד (רמז: המחלקה (String).

הדבר תלוי בסמנטיקה (במשמעות) של הפעולה equals.

- האם שוויון של שני עצמים פירושו זהות או שתוכנם שווה (השוואת תוכן לעומת השוואת מצביעים. בשפת Scheme: eq? vs. equals?).
- האם שוויון של שני עצמים הוא מצב קבוע או רגעי

כאשר טיפוס הוא מקובע אין הבדל בין מצב קבוע לרגעי ולעומת זאת בטיפוס שאינו מקובע עצמים שני עצמים שתוכנם שווה בזמן מסוים יכולים לא להיות שווים בהמשך. במחלקה String הקבעון ומנגנון ה- internig מאפשרים לממש את equals בעזרת ==

ג. הגדירו את המנשק MutableRational לתאור טיפוס רציונלי שאינו מקובע. על המנשק להכיל את החתימות המלאות של כל השרותים שציינתם בשאלה 1 סעיף ב' שהם חלק מהמנשק. יש לציין את המצב המופשט (abstract state) של הטיפוס במקום המיועד לכך מעל הגדרת המנשק. כמו כן, עבור כל אחד מהשרותים יש לציין את משמעותו בעזרת המצב המופשט (יצויין בהערה מעל חתימת השרות).

בתאור המצב המופשט ניתן להשתמש במבנים הבאים:

- this – המצב הקונקרטי של העצם לאחר ביצוע השרות
- $\langle \text{exp} \rangle$  - AF( $\langle \text{exp} \rangle$ ) – המצב המופשט של  $\langle \text{exp} \rangle$
- $\langle \text{exp} \rangle$  – \$prev( $\langle \text{exp} \rangle$ ) – מצב \ ערך לפני ביצוע השרות
- \$ret – הערך המוחזר של השרות

כדי להימנע מהורדת נקודות על טעויות נגררות מסעיף 1'ב' אנו מציינים את המצב המופשט של כל השרותים, כולל כאלו שלא צריכים להופיע במנשק מלכתחילה.

```
/** @abst: n/d    s.t. n,d ∈ Z */

public interface MutableRational {

    /** AF(this) = AF($prev(this)) + AF(other) */
    public void add(MutableRational other);

    /** AF(this) = AF($prev(this)) - AF(other) */
    public void subtract(MutableRational other);

    /** AF(this) = AF($prev(this)) * AF(other) */
    public void multiply(MutableRational other);

    /** AF(this) = AF($prev(this)) / AF(other) */
    public void divide(MutableRational other);

    /** $ret = (AF(this) = AF(other)) */
    public boolean equals(Object other);

    /** gcd(d,n) = 1 */
    public void normalize();

    /** $ret = "n/d" */
    public String toString();

    /** $ret = "x.y" ; s.t. AF(this) = x.y , */
    public String toDecimalString();

    /** $ret = n */
    public int getNumerator();

    /** $ret = d */
    public int getDenominator();

}
```

ד. הגדירו את המנשק `ImmutableRational` לתאור טיפוס רציונלי מקובע. על המנשק להכיל את החתימות המלאות של כל השרותים שציינתם בשאלה 1 סעיף ב' שהם חלק מהמנשק. יש לציין את **המצב המופשט** (`abstract state`) של הטיפוס במקום המיועד לכך מעל הגדרת המנשק. כמו כן, עבור **כל אחד מהשרותים** יש לציין את משמעותו בעזרת המצב המופשט (יציין בהערה מעל חתימת השרות).

בתאור המצב המופשט ניתן להשתמש במבנים הבאים:

- `this` – המצב הקונקרטי של העצם לאחר ביצוע השרות
- `AF(<exp>)` – המצב המופשט של `<exp>`
- `$prev(<exp>)` – מצב \ ערך `<exp>` לפני ביצוע השרות
- `$ret` – הערך המוחזר של השרות

```
/** @abst: n/d    s.t. n,d ∈ Z */

public interface ImmutableRational {

    /** $ret = AF(this) + AF(other) */
    public ImmutableRational add(ImmutableRational other);

    /** $ret = AF(this) - AF(other) */
    public ImmutableRational subtract(ImmutableRational other);

    /** $ret = AF(this) * AF(other) */
    public ImmutableRational multiply(ImmutableRational other);

    /** $ret = AF(this) / AF(other) */
    public ImmutableRational divide(ImmutableRational other);

    /** $ret = (AF(this) = AF(other)) */
    public boolean equals(Object other);

    /** gcd(d,n) = 1 */
    public void normalize();

    /** $ret = "d/n" */
    public String toString();

    /** $ret = "x.y" ; s.t. AF(this) = x.y , */
    public String toDecimalString();

    /** $ret = n */
    public int getNumerator();

    /** $ret = d */
    public int getDenominator();
}
```

## שאלה 3, 18 נקודות

המתכנת סטיב מימש את המחלקה SimpleRational לייצוג טיפוס רציונלי ובה נכלל קוד ההשוואה הבא:

```
public class SimpleRational implements ImmutableRational {
    private int num;
    private int denom;

    public SimpleRational(int num, int denom) {
        this.num = num;
        this.denom = denom;
    }

    public boolean equals(SimpleRational other) {
        return num==other.num && denom==other.denom;
    }

    // the rest of the class...
}
```

המחלקה אינה מכילה מימושים נוספים של השרות equals כדי לבדוק את נכונות המימוש כתב סטיב את מחלקת הבדיקה הבאה:

```
public class TestRationalEquality {
    public static void main(String[] args) {
        SimpleRational r1 = new SimpleRational (1,2);
        SimpleRational r2 = new SimpleRational (1,2);
        ImmutableRational ir1 = new SimpleRational (1,2);
        ImmutableRational ir2 = new SimpleRational (1,2);

        // HERE
    }
}
```

עבור כל אחת משורות הקוד הבאות ציינו האם ניתן להגדירה בפונקציה ה-main של המחלקה TestRationalEquality 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) או שהקוד ירוץ אך יחשוף באג במימוש השרות equals (אם כן, ציינו מה יהיה פלט ההדפסה, מה אמור היה להיות פלט ההדפסה ומה סיבת הבאג).

א. System.out.println(r1.equals(r2));

שגיאת קומפילציה (מהי?) / תעופה בזמן ריצה (מה הסיבה) / קוד תקין (מה מודפס) / קוד תקין החושף באג (מה מודפס? מה אמור להיות מודפס? איפה הבאג?)

[קוד תקין. מודפס true](#)

System.out.println(ir1.equals(ir2)); ב.

שגיאת קומפילציה (מהי?) / תעופה בזמן ריצה (מה הסיבה) / קוד תקין (מה מודפס) / קוד תקין החושף באג (מה מודפס? מה אמור להיות מודפס? איפה הבאג?)

קוד תקין החושף באג. מודפס false במקום true. השרות equals הוא שרות מועמס ולא דורס.

System.out.println(r1.equals(ir1)); ג.

שגיאת קומפילציה (מהי?) / תעופה בזמן ריצה (מה הסיבה) / קוד תקין (מה מודפס) / קוד תקין החושף באג (מה מודפס? מה אמור להיות מודפס? איפה הבאג?)

קוד תקין החושף באג. מודפס false במקום true. השרות equals הוא שרות מועמס ולא דורס.

System.out.println(ir1.equals(r1)); ד.

שגיאת קומפילציה (מהי?) / תעופה בזמן ריצה (מה הסיבה) / קוד תקין (מה מודפס) / קוד תקין החושף באג (מה מודפס? מה אמור להיות מודפס? איפה הבאג?)

תלוי בחתימת הפונקציה equals במנשק:

2. אם המנשק אינו מכיל את equals כלל (או מכיל equals המקבל Object כארגומנט) אז הקוד לעיל הוא קוד תקין החושף באג. מודפס false במקום true. השרות equals הוא שרות מועמס ולא דורס.
3. אם המנשק מכיל את equals בחתימה זהה לחתימה כאן, דבר שהוא שגוי לכשעצמו, אז זהו קוד תקין המדפיס true כי השרות equals הוא שרות דורס.
4. שימו לב כי לא יתכן שהמנשק מכיל שרות equals המקבל כארגומנט ImmutableRational כי אז המחלקה SimpleRational לא היתה מתקמפלת

ה.

r1 = ir1;  
System.out.println(r1.equals(ir1))

שגיאת קומפילציה (מהי?) / תעופה בזמן ריצה (מה הסיבה) / קוד תקין (מה מודפס) / קוד תקין החושף באג (מה מודפס? מה אמור להיות מודפס? איפה הבאג?)

שגיאת קומפילציה  
ההשמה אינה חוקית

ו.

ir1 = r1;  
System.out.println(r1.equals(ir1));

שגיאת קומפילציה (מהי?) / תעופה בזמן ריצה (מה הסיבה) / קוד תקין (מה מודפס) / קוד תקין החושף באג (מה מודפס? מה אמור להיות מודפס? איפה הבאג?)

קוד תקין. מודפס true (מימוש ברירת המחדל עובד בפוקס!)

**שאלה 4, 20 נקודות**

בשאלה זו נכתוב איטרטור מסוג `UniqueIterator<T>`, אשר מחזיר מכל רצף של איברים זהים רק איבר בודד.

למשל:

עבור הרצף 4 1 1 1 5 5 5 3 3 2 2 2 1 1 1  
הוא יחזיר ערכים כאילו הרצף הוא 4 1 5 3 2 1

האיטרטור יוגדר כמחלקה **פנימית** גנרית בתוך המחלקה `MyList<T>` אשר מממשת את הממשק `MyList.Iterable<T>`. היא רשימה מקושרת של אברים מהטיפוס `Cell <T>` כפי שראינו בהרצאה.

השלימו את הקוד החסר במחלקות `MyList` ו- `UniqueIterator` שבעמוד הבא. יש להשאיר את מימוש השרות `remove` של האיטרטור ריק.

```
public class Cell <T> {
    private T    cont;
    private Cell <T> next;

    public Cell (T cont, Cell <T> next) {
        this.cont = cont;
        this.next = next;
    }

    public T cont() {
        return cont;
    }

    public Cell <T> next() {
        return next;
    }

    public void setNext(Cell <T> next) {
        this.next = next;
    }
}
```

```
public class MyList<T> implements Iterable<T> {  
    private Cell<T> head;  
  
    public MyList(T... elements) {  
        this.head = null;  
        for (int i = elements.length - 1; i >= 0; i--) {  
            head = new Cell<T>(elements[i], head);  
        }  
    }  
  
    public Iterator<T> iterator() {  
        return new UniqueIterator(head);  
  
    class UniqueIterator implements Iterator<T> {  
  
        public UniqueIterator(Cell<T> cell) {  
            this.curr = cell;  
        }  
  
        public boolean hasNext() {  
            return curr != null;  
        }  
  
        public T next() {  
            T result = curr.cont();  
            while(curr != null && curr.cont().equals(result))  
                curr = curr.next();  
            return result;  
        }  
  
        public void remove() {} // must be implemented  
  
        private Cell<T> curr;  
    }  
}
```

**ושוב, בהצלחה!**