מטלת מנחה (ממ"ן) 12

הקורס: 20441 - מבוא למדעי המחשב ושפת Java

חומר הלימוד למטלה: יחידות 3 - 4 **נושאי המטלה:** שימוש במחלקות נתונות וכתיבת מחלקות

מספר השאלות: 3 **משקל המטלה:** 4 נקודות

סמסטר: 2022ב **מועד אחרון להגשה:** 9.4.2022

(ת)

מטרת מטלה זו היא להקנות לכם את עיקרי התכנות מונחה-העצמים. תתבקשו לממש מחלקות שונות המייצגות זמן וטיסה. כדי לעמוד על ההבדל בין המימוש לממשק של מחלקה, תתבקשו לכתוב שני מימושים שונים למחלקה המייצגת זמן.

שאלה 1 - 35 נקודות

המחלקה Time1 מייצגת זמן -

למחלקה Time1 יש את התכונות הפרטיות (instance variables) הבאות:

- int _hour – שמייצגת את השעה (בין 0 ל- 23);
- int _minute – שמייצגת את הדקה (בין 0 ל- 59).

למחלקה Time1 הוגדרו שני בנאים (constructors):

- האחד - בנאי המקבל שני פרמטרים (שעה ודקה)

```
public Time1(int h, int m)
```

אם אחד הפרמטרים שהתקבל אינו בתחום הנכון, הוא צריך להיות מאותחל ל- 0.

- השני - בנאי העתקה המקבל זמן אחר, ומעתיק את ערכיו.

```
public Time1 (Time1 other)
```

בנוסף הוגדרו במחלקה השיטות:

- שיטות **האחזור**:
  - int getHour() המחזירה את השעות.
  - int getMinute() המחזירה את הדקות.
- השיטות **הקובעות**:
  - void setHour (int num) המשנה את ערכה של השעה להיות num. אם num הוא לא בתחום 0-23, הערך של _hour לא משתנה.
  - void setMinute (int num) המשנה את ערכה של הדקה להיות num. אם num הוא לא בתחום 0-59, הערך של _minute לא משתנה.

- השיטה toString() שמחזירה את תוכן האובייקט כמחרוזת תווים לפי הייצוג המקובל - hh:mm. כך, המחרוזת "07:30" מייצגת את הזמן של שבע שעות ושלושים דקות. **שימו לב לדייק במחרוזת לפי הכתוב כאן, ללא רווחים וללא תווים נוספים.**

יש להקפיד שהשיטה תחזיר מחרוזת בת 5 תווים בדיוק.

```
public String toString()
```

- השיטה minFromMidnight שמחזירה כמה דקות עברו מאז חצות הלילה. למשל, אם הזמן המאוחסן באובייקט הוא 07:30, יוחזר הערך 450.

```
public int minFromMidnight()
```

- השיטה equals המקבלת כפרמטר זמן מסוים ובודקת אם הוא **זהה** בערכיו לזמן שמיוצג על ידי האובייקט עליו מופעלת השיטה. אם כן, השיטה תחזיר true ואם לא, יוחזר false.

```
public boolean equals (Time1 other)
```

- השיטה before המקבלת כפרמטר זמן מסוים ובודקת אם האובייקט שעליו מופעלת השיטה **קודם** בזמן לאובייקט שמתקבל כפרמטר. אם כן, השיטה תחזיר true ואם לא, יוחזר false. למשל, 07: 30 הוא לפני 13: 24.

```
public boolean before (Time1 other)
```

השיטה after המקבלת כפרמטר זמן מסוים ובודקת אם האובייקט שעליו מופעלת השיטה **מאוחר** בזמן לאובייקט שמתקבל כפרמטר. אם כן, השיטה תחזיר true ואם לא, יוחזר false. **השיטה הזו משתמשת אך ורק בשיטה before שהוגדרה לעיל. אסור להשתמש בשום אופרטור אחר או שיטה אחרת.**

```
public boolean after (Time1 other)
```

- השיטה difference המקבלת כפרמטר זמן מסוים ומחזירה את ההפרש בדקות בין האובייקט שעליו מופעלת השיטה לאובייקט שמתקבל כפרמטר.

שימו לב, אתם יכולים להניח שהאובייקט שעליו מופעלת השיטה מייצג זמן מאוחר יותר מהזמן שבאובייקט שהתקבל כפרמטר. כמו כן אפשר להניח ששני האובייקטים מייצגים זמנים באותה יממה.

```
public int difference(Time1 other)
```

- השיטה addMinutes המקבלת כפרמטר מספר שלם num המייצג מספר דקות, ומוסיפה אותו לזמן המיוצג על-ידי האובייקט עליו מופעלת השיטה (this). השיטה מחזירה אובייקט חדש מהמחלקה Time1 המייצג את הזמן החדש. האובייקט עליו מופעלת השיטה addMinutes לא משתנה. אם הפרמטר num הוא שלילי, השיטה מחסירה את מספר הדקות הזה מהאובייקט . שימו לב לדאוג לכך שאם ישנה גלישה ליום הבא (או יותר) או אם יש גלישה אחורה בזמן ליום אחד או יותר, עדיין התכונות צריכות להיות בטווחים של 0-23 ו-0-59. **אין צורך לציין באיזושהי צורה את העובדה שעבר יום.**

```
public Time1 addMinutes(int num)
```

עליכם לכתוב את המחלקה Time1 לפי ההגדרות לעיל.

הגדרות מדויקות לפי API תמצאו באתר הקורס ביחידה 4 בתת-פרק העוסק במטלה 12.

שימו לב שאינכם יכולים להגדיר תכונות נוספות על התכונות _hour ו- _minute.
מותר להגדיר קבועים נוספים למחלקה.
אתם יכולים להגדיר שיטות פרטיות נוספות על אלו שהוגדרו לעיל, אבל לא שיטות ציבוריות ולא תכונות נוספות.

שאלה 2 - 35 נקודות

המחלקה Time2 מייצגת זמן, לפי מספר הדקות שעברו מאז חצות הלילה ועד לזמן שמייצג האובייקט.

לדוגמא, אם הזמן שמיוצג על ידי האוביקט הוא: 07:30 (כלומר 7 בבוקר, 30 דקות), הוא ייוצג על-ידי הערך 450 שכן, 7 * 60 + 30 = 450

למחלקה Time2 יש, אם כן, תכונה פרטית (instance variable) אחת, והיא מספר הדקות הזה.
int _minFromMid

עליכם לכתוב את המימוש של המחלקה Time2.

ה- API של שתי המחלקות Time1 ו- Time2 זהה לחלוטין! רק הייצוג הפנימי של האובייקטים (התכונות) שונה.

Time2 (int h, int m)	בנאי המקבל שני פרמטרים (שעה ודקה). אם אחד הפרמטרים שהתקבל אינו בתחום הנכון, הוא צריך להיות מאותחל ל- 0.
Time2 (Time2 other)	בנאי העתקה המקבל זמן אחר, ומעתיק את ערכיו.
int getHour()	שיטות מאחזרות
int getMinute()	
void setHour (int num)	שיטות קובעות בשיטות הקובעות אם הפרמטר אינו תקין יש להשאיר את התכונה ללא שינוי
void setMinute (int num)	
int minFromMidnight ()	שיטה המחזירה כמה דקות עברו מאז חצות הלילה
boolean equals(Time2 other)	שיטה הבודקת האם הזמנים שווים
boolean before(Time2 other)	השיטה בודקת האם הזמן שעליו הופעלה השיטה מוקדם מהזמן שהתקבל כפרמטר

boolean after(Time2 other)	השיטה בודקת האם הזמן שעליו הופעלה השיטה מאוחר לזמן שהתקבל כפרמטר (השיטה הזו יכולה להשתמש אך ורק בשיטה *before* שהוגדרה במחלקה *Time2* ולא בשום אופרטור אחר או שיטה אחרת).
int difference(Time2 other)	שיטה המחזירה את ההפרש בדקות בין האובייקט שעליו מופעלת השיטה לאובייקט שמתקבל כפרמטר. אתם יכולים להניח שהאובייקט שעליו מופעלת השיטה מייצג זמן מאוחר יותר מהזמן שבאובייקט שהתקבל כפרמטר. כמו כן אפשר להניח ששני האובייקטים מייצגים זמנים באותה יממה.
String toString()	שיטה המחזירה את תוכן האובייקט כמחרוזת תווים לפי הייצוג המקובל - hh:mm. כך, המחרוזת "07:30" מייצגת את הזמן של שבע שעות ושלושים דקות. שימו לב לדייק במחרוזת לפי הכתוב כאן, ללא רווחים וללא תווים נוספים. יש להקפיד שהשיטה תחזיר מחרוזת בעלת 5 תווים בדיוק.
Time2 addMinutes(int num)	שיטה המקבלת כפרמטר מספר שלם num המייצג מספר דקות, ומוסיפה אותו לזמן המיוצג על-ידי האובייקט עליו מופעלת השיטה (this). השיטה מחזירה אובייקט חדש מהמחלקה Time2 המייצג את הזמן החדש.

שימו לב שאינכם יכולים להגדיר תכונות נוספות על התכונה minFromMid_.

מותר להגדיר קבועים נוספים למחלקה.

במילים אחרות, חתימות השיטות של המחלקה Time1 זהות לחלוטין לאלו של Time2, לבד מהמקרים בהם מתקבל זמן כפרמטר לשיטה, ואז במקום שכתוב Time1 צריך להיות Time2.

שימו לב, גם השיטה toString של המחלקה Time2 צריכה להיות לפי זו של המחלקה Time1, כלומר להדפיס את הזמן לפי שעות ודקות ולא לפי מספר הדקות מאז חצות.

אין להשתמש בשיטות ובבנאים של המחלקה Time1 במחלקה Time2. מדובר במימושים חלופיים למחלקה של הזמן.

שאלה 3 - 30 נקודות

המחלקה Flight מייצגת טיסה.

למחלקה Flight התכונות הפרטיות (instance variables) הבאות:

- String _origin – שם העיר ממנה ממריאה הטיסה.
- String _destination – שם העיר בה נוחתת הטיסה.
- Time1 _departure – זמן ההמראה של הטיסה.
- int _flightDuration – משך זמן הטיסה בדקות.
- int _noOfPassengers – מספר הנוסעים בטיסה.
- boolean _isFull – האם הטיסה מלאה?
- int _price – מחיר לכרטיס טיסה לאדם.

כמו כן קיים במחלקה קבוע שלם MAX_CAPACITY המציין את המספר המקסימלי של נוסעים על טיסה - 250.

למחלקה Flight יש שני בנאים:

- בנאי אחד שמקבל כפרמטרים: שם עיר ההמראה, שם עיר הנחיתה, שני מספרים שלמים המהווים את זמן ההמראה של הטיסה (שעות ודקות), מספר שלם המייצג את משך זמן הטיסה בדקות, מספר שלם המייצג את מספר הנוסעים בטיסה ומספר שלם המייצג מחיר כרטיס טיסה לאדם.
  שימו לב להנחות הבאות:
  - אם מספר הנוסעים גדול מהמספר המקסימלי של נוסעים האפשרי, הערך שיינתן לתכונה של מספר הנוסעים יהיה המספר המקסימלי ולא הפרמטר.
  - אם מספר הנוסעים קטן מ- 0, ינתן הערך 0.
  - את התכונה הבוליאנית צריך לקבוע לפי מספר הנוסעים והקבוע המציין את מספר הנוסעים המקסימלי האפשרי.
  - אם הפרמטר המייצג את משך זמן הטיסה בדקות קטן מ- 0, הערך שיינתן לתכונה של זמן הטיסה יהיה 0.
  - אם הפרמטר המייצג את מחיר כרטיס טיסה קטן מ- 0, הערך שיינתן לתכונה זו יהיה 0.
  - אם המספרים המהווים את זמן ההמראה של הטיסה שגויים, הטיפול בכך יתבצע במחלקה Time1.
- בנאי העתקה המקבל טיסה אחרת, ומעתיק את ערכיה.

במחלקה הוגדרו פעולות get ו- set לפי השמות המקובלים. ראו פרטים מדוייקים ב- API וראו שם גם הנחיות לטיפול במקרי קצה.

כמו כן הוגדרו השיטות הבאות:

- equals שיטה המקבלת טיסה אחרת כפרמטר ומחזירה true אם הטיסה שעליה השיטה מופעלת והטיסה שהתקבלה כפרמטר זהות. הזהות תיקבע לפי שמות הערים של ההמראה והנחיתה וזמן ההמראה.
- getArrivalTime שיטה המחשבת את זמן הנחיתה של הטיסה ומחזירה זמן זה.
- addPassengers שיטה בוליאנית המקבלת מספר של נוסעים num, ומוסיפה אותם לטיסה, אם יש בה מקום. אם יש, היא מחזירה true, אם אין מקום לכולם, היא לא מוסיפה אף אחד, ומחזירה false. שימו לב שצריך לעדכן גם את התכונה הבוליאנית isFull במקרה והיא אמורה להשתנות. ניתן להניח שהשיטה מקבלת מספר חיובי.
- isCheaper שיטה המקבלת טיסה אחרת כפרמטר ומחזירה true אם כרטיס לטיסה עליה מופעלת השיטה הוא זול יותר מכרטיס לטיסה שהתקבלה כפרמטר, אחרת יוחזר false.
- totalPrice שיטה המחשבת את התשלום הכולל שהתקבל מכל נוסעי הטיסה, ומחזירה ערך זה.
- landsEarlier שיטה המקבלת טיסה אחרת כפרמטר ומחזירה true אם הטיסה עליה מופעלת השיטה נוחתת בזמן מוקדם יותר מאשר זמן הנחיתה של הטיסה שהתקבלה כפרמטר, אחרת יוחזר false.
- toString שיטה המחזירה מחרוזת ובה נתוני הטיסה הבאים (בלבד), לפי הפורמט הבא:

  Flight from _*origin* to _*destination* departs at _*departure*. Flight is full.

  אם הטיסה לא מלאה, יודפס .Flight is not full.

הערות כלליות:

שימו לב, אסור להוסיף תכונות פרטיות.

מותר להוסיף שיטות פרטיות.

אין להשתמש במספרים בקוד. יש להוסיף קבועים (final) עבור כל מספר קבוע ולהשתמש בקבוע בקוד.

בכל השיטות במטלה שמקבלות אובייקט כפרמטר אפשר להניח שמתקבל אובייקט שאותחל ואינו שווה ל- null.

הגדרות מדויקות לבנאים ולשיטות הנדרשות לפי API תמצאו באתר הקורס.

שימו לב ששמנו טסטרים לשלוש המחלקות באתר הקורס. חובה שטסטרים אלו ירוצו ללא שגיאות קומפילציה עם המחלקות שלכם. אם יש שיטה שלא כתבתם, כתבו חתימה והחזירו ערך סתמי כדי שהטסטרים ירוצו עם המחלקות ללא שגיאות קומפילציה. אם הטסטרים לא ירוצו ללא שגיאות קומפילציה הציון במטלה יהיה אפס.

שימו לב לא לבצע aliasing במקומות המועדים.

עליכם לתעד את כל המחלקות שתכתבו ב- API וגם בתיעוד פנימי. אפשר כמובן להשתמש בהערות ה-API שנמצאות באתר.

הגשה

1. הגשת הממ"ן נעשית בצורה אלקטרונית בלבד, דרך מערכת שליחת המטלות.
2. הקפידו ששמות המחלקות והשיטות יהיו בדיוק כפי שמוגדר בממ"ן. **אחרת יורדו לכם הרבה נקודות!**
3. חובה להריץ את הטסטרים שנמצאים באתר הקורס על המחלקות שכתבתם. שימו לב שהטסטרים לא מכסים את כל האפשרויות, ובפרט לא את מקרי הקצה. הם רק בודקים את השמות של השיטות במחלקות. מאד מומלץ להוסיף להם בדיקות. שימו לב שאם הטסטרים לא יעברו קומפילציה מול המחלקות שכתבתם, לא יקבלו נקודות בכלל. אם יש שיטה שאתם מעוניינים לדלג עליה, עלכם לרשום את חתימת השיטה ולהחזיר ערך סתמי על מנת שהטסטרים יעברו קומפילציה.
4. את התשובות לשאלות יש להגיש בשלושה קובצי Java הבאים: Time2.java, Time1.java, Flight.java
5. ארזו את כל הקבצים בקובץ zip יחיד ושלחו אותו בלבד.

ב ה צ ל ח ה

מטלת מנחה (ממ"ן) 13

הקורס: 20441 - מבוא למדעי המחשב ושפת Java

חומר הלימוד למטלה: יחידות 5 – 6 **נושא המטלה:** לולאות ומערכים

מספר השאלות: 1 **משקל המטלה:** 5 נקודות

סמסטר: 2022ב **מועד אחרון להגשה:** 23.4.2022

(ת)

במטלה זו אנו משתמשים במחלקות Time1 ו- Flight שכתבנו בממ"ן 12.

אתם יכולים להשתמש במחלקות Time1 ו- Flight **שכתבתם** או בקבצים Time1.class ו- Flight.class שיהיו באתר ביחידה 6 בצמוד למטלה 13. נשים את הקבצים האלו באתר רק אחרי ההגשה של מטלה 12.

שימו לב שהקבצים שיש באתר הם קובצי class ולא הקוד ב- Java. אי אפשר לפתוח אותם אלא להשתמש בהם בלבד. לצידם יש קובצי html שהם ה- API של המחלקות Time1 ו- Flight.

אנא קראו את הכתוב במדריך **מדריך עזר קצר לשימוש בקובצי מחלקות (class) הקיימים בפרויקט שאתם יוצרים ב-Blue**. המדריך נמצא בלשונית "מדריכי עזר" ביחידה 3 באתר הקורס. כך תדעו איך להשתמש במחלקה שכבר כתובה, וניתנת לכם כקובץ class ללא הקוד.
שמנו באתר טסטר בסיסי לבדיקה ראשונית של המטלה. **חובה** להריץ את המטלה מול הטסטר ולבדוק שאין טעויות קומפילציה.

שאלה 1 - להרצה (100%)

המחלקה Airport מייצגת את לוח הטיסות בשדה התעופה בימה.

הייצוג נעשה על-ידי מערך ששומר את רשימת הטיסות. התכונות במחלקה הן:

- מערך של הטיסות Flight [] _flightsSchedule
- מספר הטיסות בלוח הטיסות (במערך) int _noOfFlights
- שם העיר בה נמצא שדה התעופה String _city

כמו כן קיים במחלקה קבוע שלם MAX_FLIGHTS המציין את המספר המקסימלי של טיסות בימה– 200.

הטיסות (כלומר האובייקטים מהמחלקה Flight) נמצאים במערך ברצף, **ללא "חורים" מתחילת המערך. המערך צריך להישאר כך (ללא חורים) לאחר כל פעולה.**

עליכם לכתוב את המימוש ב- Java של המחלקה Airport. מימוש המחלקה כולל את הסעיפים שלהלן:

1. הגדרת הקבוע של המחלקה.
2. הגדרת התכונות של המחלקה.
3. בנאי שמקבל את שם העיר בה נמצא שדה התעופה ומאתחל את תכונות המחלקה כך שמערך הטיסות יהיה בגודל מקסימלי ומספר הטיסות הוא 0.
4. שיטה (addFlight) בוליאנית המוסיפה טיסה ללוח הטיסות. השיטה מקבלת את **הטיסה** כפרמטר. השיטה מחזירה ערך true אם ההוספה התבצעה כשורה, אם לא, השיטה תחזיר false. שימו לב שהמקור או היעד של הטיסה חייבים להיות העיר בה נמצא שדה התעופה. אתם יכולים להניח שהטיסה הזו לא קיימת כבר בלוח הטיסות. אין צורך לבדוק זאת.
5. שיטה (removeFlight) בוליאנית המוחקת טיסה מלוח הטיסות. השיטה מקבלת את **הטיסה** כפרמטר. השיטה מחזירה ערך true אם המחיקה התבצעה כשורה, אם לא, השיטה תחזיר false.
6. שיטה (firstFlightFromOrigin) המקבלת עיר כלשהי place, מחזירה את הזמן בו ממריאה הטיסה הראשונה באותו יום מהמקום place. אם אין אף טיסה באותו יום מהמקום place יוחזר null.
7. שיטה (howManyFullFlights) המחזירה מספר האומר כמה טיסות מלאות יש באותו יום.
8. שיטה (howManyFlightsBetween) המקבלת עיר place ומחזירה מספר האומר כמה טיסות יש באותו יום הממריאות משדה התעופה city ונוחתות ב- place, או ממריאות מ- place ונוחתות ב- city .
9. שיטה (mostPopularDestination) המחזירה את העיר הכי פופולרית באותו יום (כלומר העיר בה נוחתות הכי הרבה טיסות). אם אין טיסות בכלל בלוח הטיסות, יוחזר null. אם יש כמה ערים שהן פופולריות ביותר באותה מידה, תוחזר העיר הפופולרית הראשונה שנמצאה בלוח הטיסות.
10. שיטה (mostExpensiveTicket) המחזירה את **הטיסה** שהכרטיס שלה הוא היקר ביותר. אם אין טיסות בכלל בלוח הטיסות, יוחזר null. אם יש כמה טיסות שהן יקרות ביותר באותה מידה, תוחזר הטיסה היקרה ביותר הראשונה שנמצאה בלוח הטיסות.
11. שיטה (longestFlight) המחזירה את **הטיסה** הארוכה ביותר במערך הטיסות. אם אין טיסות בכלל בלוח הטיסות, יוחזר null. אם יש כמה טיסות שהן ארוכות ביותר באותה מידה, תוחזר הטיסה הארוכה ביותר הראשונה שנמצאה בלוח הטיסות.
12. שיטה (toString) המחזירה מחרוזת המתארת את כל הטיסות במערך הטיסות כסדרן לפי המערך (אין צורך למיין לפי זמנים או משהו אחר). כל טיסה תהיה בשורה נפרדת. ובתחילה תהיה כותרת. אם אין טיסות בכלל בלוח הטיסות, יוחזר null.

ראו את הדוגמא הבאה:

The flights for airport Tel-Aviv today are:

Flight from Tel-Aviv to London departs at 12:00. Flight is full.

Flight from New York to Tel-Aviv departs at 10:50. Flight is full.

Flight from Tel-Aviv to Paris departs at 11:35. Flight is not full.

בכל השיטות לעיל, אם מועבר אובייקט כפרמטר, אפשר להניח שהוא אינו null.

שימו לב לא לבצע aliasing במקומות המועדים.

תזכורת – השוואה בין אובייקטים (ומחרוזות) צריכה להיעשות בעזרת השיטה equals ולא על-ידי סימן השוויון ==.

מותר להוסיף שיטות נוספות (פרטיות בלבד), לפי ראות עיניכם, אבל אי אפשר להוסיף תכונות נוספות. כמו כן, כל התכונות חייבות להיות פרטיות!

לפניכם רשימת החתימות של הבנאי ושיטות המחלקה:

- `public Airport(String city)`
- `public boolean addFlight(Flight f)`
- `public boolean removeFlight(Flight f)`
- `public Time1 firstFlightFromOrigin (String place)`
- `public String toString()`
- `public int howManyFullFlights()`
- `public int howManyFlightsBetween (String city)`
- `public String mostPopularDestination()`
- `public Flight mostExpensiveTicket()`
- `public Flight longestFlight()`

אתם צריכים לכתוב בעצמכם API למחלקה, לבנאים ולשיטות לפי הנהוג בכתיבת API. כמו כן, עליכם לתעד בתיעוד פנימי כל מה שדורש הבהרה ואינו פשוט.

שימו לב,

באתר הקורס תמצאו גם טסטר לבדיקת האיות והפרמטרים של השמות של השיטות והמחלקה שאתם צריכים לכתוב. חובה עליכם לבדוק את המחלקה שכתבתם בטסטר זה, ולהגיש אותה רק אם הטסטר עובר קומפילציה. שימו לב שהטסטר לא מכסה את כל האפשרויות, ובפרט לא את מקרי הקצה. הוא רק בודק את השמות של השיטות במחלקות כלומר שגיאות קומפילציה. מאד מומלץ להוסיף לו בדיקות. **אם יש שיטה שלא כתבתם, כתבו חתימה והחזירו ערך סתמי כדי שהטסטר ירוצו עם המחלקה ללא שגיאות קומפילציה. אם הטסטר לא ירוץ ללא שגיאות קומפילציה הציון במטלה יהיה אפס.**

הגשה

1. הגשת הממ״ן נעשית בצורה אלקטרונית בלבד, דרך מערכת שליחת המטלות.
2. הקפידו לתעד בתיעוד פנימי וב- API את כל השיטות שיש במחלקות השונות.
3. הקפידו ששמות השיטות יהיו בדיוק כפי שכתוב במטלה. וכן שההדפסות יהיו בדיוק כפי שמופיע במטלה.
4. עליכם להגיש את הקובץ Airport.java, עטפו אותו בקובץ zip ושלחו. אין לשלוח קבצים נוספים.