מבוא למדעי המחשב – סמסטר א' תשע"ט

עבודת בית מספר 4 (תכנות מונחה עצמים)

מתרגלת אחראית: נועה בן-דוד

תאריך פרסום: 24/5/2019

תאריך הגשה: 7/6/2019, 14:00 בצהריים.

הוראות מקדימות

הגשת עבודות בית

1. קראו את העבודה מתחילתה ועד סופה לפני שאתם מתחילים לפתור אותה. ודאו שאתם מבינים את כל השאלות. רמת הקושי של המשימות אינה אחידה: הפתרון של חלק מהמשימות קל יותר התחילו בהן.
2. ניתן להגיש את העבודה לבד או בזוג. ההגשה מתבצעת דרך מערכת ההגשות (Submission System) המחלקתית. אין לפתור את תרגילי הבית עם כל אדם אחר.
3. אין לשנות את שמות הקבצים.
4. אין להגיש קבצים נוספים.
5. שם קובץ ה-ZIP יכול להיות כרצונכם, אך באנגלית בלבד. בנוסף, הקבצים שתגישו יכולים להכיל טקסט המורכב מאותיות באנגלית, מספרים וסימני פיסוק בלבד. טקסט אשר יכיל תווים אחרים (אותיות בעברית, יוונית וכד'..) לא יתקבל.
6. קבצים שיוגשו שלא על פי הנחיות אלו לא ייבדקו.
7. את קובץ ה-ZIP יש להגיש ב-Submission System.
8. אין להשתמש ב- **packages**. אם תעשו בהן שימוש עבודתכם לא תתקבל על ידי מערכת ההגשות. בידקו כי המילה **package** אינה מופיעה בקובצי ההגשה שלכם.

בדיקת עבודות הבית

9. עבודות הבית נבדקות באופן ידני וכן באופן אוטומטי. הבדיקה האוטומטית מתייחסת לערכי ההחזרה של הפונקציות/שיטות או לפעולות אשר הן מבצעות וכן לפלט התכנית המודפס למסך (אם קיים). לכן, יש להקפיד על ההוראות ולבצע אותן במדוייק. כל הדפסה אשר אינה עונה בדיוק על הדרישות המופיעות בעבודה (כולל שורות, רווחים, סימני פיסוק או כל תו אחר - מיותרים, חסרים או מופיעים בסדר שונה מהנדרש), לא תעבור את הבדיקה האוטומטית ולכן תגרור פגיעה בציון.
10. סגנון כתיבת הקוד ייבדק באופן ידני. יש להקפיד על כתיבת קוד ברור, **הימנעות משכפול קוד**, מתן שמות משמעותיים למשתנים, על הזחות (אינדנטציה), ועל הוספת הערות בקוד המסבירות את תפקידם של מקטעי הקוד השונים. אין צורך למלא את הקוד בהערות סתמיות, אך חשוב לכתוב הערות בנקודות קריטיות, המסבירות קטעים חשובים בקוד. הערות יש לרשום אך ורק באנגלית. כתיבת קוד אשר אינה עומדת בדרישות אלו תגרור הפחתה בציון העבודה.

עזרה והנחיה

11. לכל עבודת בית יש מתרגל האחראי עליה. ניתן לפנות למתרגל בשעות הקבלה.

הערות ספציפיות לעבודת בית זו

12. לעבודה זו מצורפים קבצי Java עם השמות הנדרשים כמפורט בכל משימה. צרו תיקייה חדשה והעתיקו את קובצי ה- Java לתוכה. עליכם לערוך את הקבצים האלו בהתאם למפורט בתרגיל ולהגישם כפתרון, מכווצים כקובץ ZIP יחיד. שימו לב: עליכם להגיש רק את קובצי ה-Java הנדרשים.
13. בעבודה זו ניתן להגדיר פונקציות (עזר) נוספות, לפי שיקולכם. פונקציות אלו ייכתבו בתוך קובצי המשימה הרלוונטיים.
14. שימו לב, כחלק מהעבודה קיבלתם מימושים של המחלקות Pair, Sorter, StudentInfo, LinkedList, DynamicArray, והממשקים List. כמו כן, קיימים מימושים חלקיים למחלקות, Student, Course. **אין לשנות את המימושים שקיבלתם.**
15. בעבודה זו אנו משתמשים ב2 ממשקים עיקריים –
    15.1. List אשר נלמד בכיתה, בעבודה הרחבנו את הממשק במספר שיטות נוספות אשר לא נלמדו בכיתה, ניתן לראות את מימושן בקובץ LinkedList.java, DynamicArray.java.
    15.2. Comparable אשר ילמד בשיעור 17. קוד הממשק נמצא בקובץ Comparable.java . דוגמה למימוש הממשק ניתן למצוא בקובץ ComparbleExample.java.
16. בנוסף ניתנת לכם המחלקה Sorter המשמשת למיון רשימות, ניתן לראות שימוש בה בComparableExample.java.

יושר אקדמי

הימנעו מהעתקות! ההגשה היא ביחידים או בזוג. אם תוגשנה שתי עבודות עם קוד זהה או אפילו דומה - זוהי העתקה, אשר תדווח לאלתר לוועדת משמעת. אם טרם עיינתם בסילבוס הקורס אנא עשו זאת כעת.

מערכת ניהול סטודנטים
בגישת תכנות מונחה עצמים

בעבודה 3 כתבתם שלושה מחלקות אשר מהוות את הבסיס למערכת לניהול רישום של סטודנטים, בעבודה זו אתם תשלימו את המערכת, ותייצרו מערכת ניהול רישום לקורסים.

מבנה הייצוג החדש:
ישנם שלושה מרכיבים עיקריים במערכת הרישום –

- מערכת התארים: בחלק זה אתם תבנו מערכת שמאפשרת להגדיר תארים, כל תואר יכיל את כל המידע הרלוונטי – כמה נק"ז התואר, מה הם קורסי חובה הנדרשים לסיומו, איזה קורסי בחירה ניתן לבחור, מהו קהל היעד של התואר.
- מערכת הרישום: בחלק זה תסיימו את ייצוג מערכת הרישום, אתם תגדירו מערכת שלמה אשר כוללת בתוכה את כל הסטודנטים, הקורסים, התארים, ודרכה עושים את כל הפעולות של המערכת (רישום לקורסים, סגירת תואר, מתן ציונים, הוספת סטודנטים וכו...)
- אחזור מידע: בחלק זה של העבודה אתם תוסיפו פונקציונליות למערכת הרישום, ותאפשרו לה יכולות של ניתוח נתונים ואחזור מידע למשתמש.

הנחיות למשימות:

- לא ניתן לשנות את חתימת השיטות הציבוריות במחלקה.
- פרט לשיטות אותן נדרשתם להשלים, מותר לכם (ואף מומלץ בחום רב) להוסיף שיטות עזר, אם תזדקקו להן. שיטות אלו יהיו בעלות הרשאה private.
- לכל השדות יהיה מאפיין הרשאה private.
- במידה ולא צוין אחרת, אין הנחות על תקינות הקלטים לפונקציות\שיטות.

בדיקה עצמית:

לעבודה מצורף קובץ בשם Assignment4Tester , קובץ זה מכיל מס' רב של בדיקות, אך שימו לב **שהוא לא בודק הכל**, מעבר של כל הבדיקות בקובץ לא מבטיח 100 בעבודה!

אנו ממליצים לכם להרחיב את הבדיקות בעצמכם, ולכתוב בדיקות מקיפות אשר יכסו את כלל המקרים!

משימה 1: מימוש מערכת התארים (30 נקודות)

במשימה זו עליכם להשלים את מימוש המחלקות Course, Degree, BachelorDegree, MasterDegree.

המחלקה Course: (10 נקודות)

במערכת שלנו, קורסים מיוצגים על ידי המחלקה Course, הממשת את Comparable<Course>.
יש להגדיר את הטיפוס Course במחלקה בשם Course בקובץ Course.java.

לקורסים, לכל סוגיהם, ישנן שלוש תכונות המגדירות אותם: שם, מספר סידורי ומספר נק"ז. לכל קורס מספר סידורי ייחודי משלו אך שם הקורס לא חייב להיות ייחודי. במחלקה Course נייצג תכונות אלו באמצעות השדות הבאים:

- שם הקורס: נייצג באמצעות משתנה מטיפוס String.
- מס' הקורס: נייצג באמצעות Integer
- נק"ז: נייצג באמצעות מספר שלם (ניתן לבחור טיפוס כרצונכם).
- קורסי קדם: רשימה של קורסי קדם ישירים (לדוגמה, "SPL" הוא קדם ישיר ל"אלגוריתמים", אבל "מבוא למדעי המחשב" הוא לא קדם ישיר ל"אלגוריתמים", למרות שהוא כן **קורס קדם** ל"אלגוריתמים").

חלק מהשיטות כבר נתונות לכם, יש להשלים את השיטות הבאות:

1. **public** boolean isPreliminaryCourse(Course other): השיטה מקבלת קורס other ובודקת האם הקורס הנוכחי הוא קורס קדם שלו. לדוגמה, הקריאה:

   introToCs.isPreliminaryCourse(algorithms)

   תחזיר true, שכן הקורס "מבוא למדעי המחשב" הוא קודם לקורס "אלגוריתמים", ולא ניתן לקחת את הקורס "אלגוריתמים" לפני שהשלמתם בהצלחה את הקורס "מבוא למדעי המחשב".

2. **public int** compareTo(Course other) : השיטה של הממשק Comparable מחילה יחס סדר על הקורסים בצורה הבאה: במידה וקורס A הוא קורס קדם לקורס B, $B > A$, במידה ואין יחס קדם/לא קדם ביניהם, אז ההשוואה תתבצע ע"פ מספר הקורס.

3. **public** List<Course> getAllPreliminaryCourses() השיטה מחזירה את רשימת קורסי הקדם, וכל קורסי הקדם שלהם (**לעומק** כלל שידרש, לדוגמא הקורס "מבוא למדעי המחשב" הינו קדם לקורס "אוטומטים", הקורס "אוטומטים" הינו קדם לקורס "אלגוריתמים". כאשר נפעיל את השיטה על הקורס המייצג את "אלגוריתמים" תחזור רשימה עם הקורסים "מבוא למדעי המחשב" ו"אוטומטים").

4. **public** void addPreliminaryCourse(Course course) השיטה מקבלת קורס, ומוסיפה אותו לרשימת קורסי הקדם.

5. **public** void addPreliminaryCourses(List<Course> courses) השיטה מקבלת רשימה של קורסים, ומוסיפה אותם לרשימת קורסי הקדם.

שימו לב: על כל השיטות המחזירות רשימות של קורסים, להחזירן ממוינות בסדר עולה (סדר עולה < מוגדר על ידי השיטה compareTo).

המחלקה Degree: (10 נקודות)

המחלקה Degree מייצגת תואר, ויורשות אותה המחלקות MasterDegree ו-BachelorDegree,
יש להגדיר את הטיפוס Degree במחלקה בשם Degree בקובץ Degree.java.

השדות במחלקה:

- שם התואר, ייצוג ע"י String, לדוגמה "Computer Science"
- קוד התואר, ייצוג ע"י מספר שלם, מימוש ע"פ טיפוס לבחירתכם.
- רשימה של קורסי חובה
- רשימה של קורסי בחירה
- נק"ז הנדרש לסיום התואר

יש להשלים את הבנאי: **public** Degree(String name, int degreeCode, int requiredCredits)
הבנאי מקבל שם וקוד תואר, ומאתחל את שדות העצם.

יש להשלים את השיטות הבאות:

1. **public** String getDegreeName() מחזיר את שם התואר.

2. **public** int getDegreeCode() מחזיר את קוד התואר.

3. **public** List<Course> getMandatoryCourses() מחזיר את רשימת קורסי החובה בסדר כלשהו.

4. **public** List<Course> getElectiveCourses() מחזיר את רשימת קורסי הבחירה בסדר כלשהו.

5. **public** boolean addCourse(Course course, boolean mandatory) השיטה מוסיפה קורס חובה או בחירה לתואר.
אם הקורס הוא **חובה**, יש להוסיף את כל קורסי הקדם שלו כקורסי חובה. הוספת הקורס תצליח אם סך הנק"ז שנוסף בעקבות הקורס וקורסי הקדם שלו אינו חורג מסך הנק"ז הדרוש לתואר.
שימו לב: ייתכן כי חלק מקורסי הקדם כבר קיימים בתואר. במקרה זה לא נספור את הנק"ז שלהם פעם נוספת.
אם הקורס הוא **בחירה**, הוספתו תמיד מצליחה. כמו כן, אין להוסיף את קורסי הקדם שלו.
השיטה תחזיר true אם הוספת הקורס (בחירה/חובה) הצליחה, ו-false אחרת.

6. **public** int getRequiredCredits() מחזיר את כמות הנק"ז הנדרשת להשלמת התואר.

7. **public** int getMandatoryCredits() מחזיר את כמות הנק"ז של קורסי החובה בלבד, ערך זה נגזר מרשימת קורסי החובה.

המחלקה BachelorDegree: (5 נקודות)

המחלקה BachelorDegree מייצגת תואר ראשון, והיא יורשת את המחלקת Degree.
יש להגדיר את הטיפוס BachelorDegree במחלקה בשם BachelorDegree בקובץ BachelorDegree.java.

על מנת לסיים את התואר הראשון נדרש להשלים 20 נק"ז סך הכל (מיוצג כשדה סטאטי REQUIRED_CREDITS במחלקה).

יש להשלים את הבנאי:
public BachelorDegree(String name, int degreeCode)

המחלקה MasterDegree: (5 נקודות)

המחלקה MasterDegree מייצגת תואר שני, והיא יורשת את המחלקת Degree.
יש להגדיר את הטיפוס MasterDegree במחלקה בשם MasterDegree בקובץ MasterDegree.java.

על מנת לסיים את התואר השני נדרש להשלים 10 נק"ז סך הכל (מיוצג כשדה סטאטי REQUIRED_CREDITS).
בנוסף קיימת האפשרות ללמוד לתואר שני ללא תזה. תכונה זו תצויין על ידי שדה בוליאני המציין האם התואר הוא עם/בלי תזה.

יש להשלים את הבנאי:
public MasterDegree(String name, int degreeCode, boolean withThesis)

יש להשלים את השיטה:
1. **public** Boolean getWithResearch () מחזיר את הערך של עם/בלי תזה.

משימה 2: ייצוג מונחה עצמים של מערכת הרישום (50 נקודות)

בעבודת הבית השלישית בניתם ייצוג פשוט יחסית של מערכת הרישום. במשימה זו נגדיר ייצוג מורכב יותר, בעל יותר יכולות.

המחלקה Grade: (5 נקודות)

מחלקה זו מייצגת ציון של סטודנט בקורס.
יש להגדיר את הטיפוס Grade במחלקה בשם Grade בקובץ Grade.java.

למחלקה Grade יש את השדות הבאים:
- קורס: נייצג באמצעות משתנה מטיפוס Course.

ציון: נייצג באמצעות מספר שלם בטווח 0 עד 100 (ניתן לבחור טיפוס כרצונכם).

יש להשלים את הבנאי: **public** Grade(Course course, int grade)
הבנאי מקבל את הפרמטרים ומאתחל את האובייקט, יש לבדוק:
- הפרמטר Course אינו יכול להיות null.
- הציון בטווח 0 עד 100.

אם אחד מהתנאים הללו לא מתקיים על הבנאי לזרוק חריגה מטיפוס IllegalArgumentException.

1. **public** Course getCourse(): השיטה אינה מקבלת פרמטרים ומחזירה את הקורס.
2. **public** int getGrade(): השיטה אינה מקבלת פרמטרים ומחזירה את הציון בקורס.
3. **public** int setGrade(int grade): השיטה מקבלת ציון עדכני ומעדכנת את ערך השדה ציון ומחזירה את הציון טרם העדכון.
4. **public** String toString(): השיטה אינה מקבלת פרמטרים ומחזירה מחרוזת המייצגת את מצב העצם (מתארת את הציון). אתם יכולים להגדיר את המחרוזת כרצונכם.
5. **public boolean** equals(Object other) השיטה מגדירה שוויון בין שני עצמים מטיפוס Grade. שני עצמים כאלו שווים, כאשר מתקיימים שני תנאים: ראשית הם צריכים לייצג ציון באותו קורס, ושנית הערכים המספריים של הציון צריכים להיות שווים. השיטה מקבלת משתנה מטיפוס Object ומחזירה true אם other הוא מהטיפוס Grade, ושווה לאובייקט המפעיל את השיטה.

המחלקה Student: (15 נקודות)
המחלקה Student מייצגת סטודנט, וממשת את הממשק Comparable<Student>.
יש להגדיר את הטיפוס Student במחלקה בשם Student בקובץ Student.java.

עליה לכלול את השדות הבאים:
- פרטי הסטודנט , ע"י טיפוס מסוג StudentInfo.
- סוג תואר, ע"י טיפוס מסוג Degree.
- רשימת הקורסים אליהם הסטודנט רשום כרגע, רשימה של אובייקטים מסוג Course.
- רשימת הציונים של הסטודנט, רשימה של אובייקטים מטיפוס Grade.
- שנה נוכחית של הסטודנט. (לדוג' שנה א', ב', וכו')
- שנת סיום הלימודים של הסטודנט. (לדוג' 2016)

חלק מהשיטות כבר נתונות לכם, יש להשלים את השיטות הבאות:

1. **public** void increaseYear(): מגדיל את שנת הלימודים של הסטודנט בשנה אחת.

2. **public** void closeDegree(int year): משנה לסטודנט את שנת הלימודים ל-0, וקובע מה השנה בה הסטודנט סיים את הלימודים.

3. **public** boolean isRegisteredTo(Course course):
השיטה מקבלת משתנה מטיפוס Course ומחזירה true אם ורק אם הסטודנט רשום לקורס. ניתן להניח כי הקלט שונה מ- null.

4. public boolean isCompleted(Course course, int passGrade): השיטה מקבלת קורס וציון עובר עבורו, ומחזירה true אם ורק אם הסטודנט השלים את הקורס עם ציון עובר ומעלה. כלומר, קיימת רשומה ברשימת הציונים עבור הקורס עם ציון הגדול שווה מהציון העובר. ניתן להניח כי הקלט שונה מ-null.

5. **public** boolean registerTo(Course course):
השיטה מקבלת כפרמטר ערך מטיפוס Course ומוסיפה אותו אל רשימת הקורסים של הסטודנט אם הדבר אפשרי. ניתן לרשום סטודנט לקורס מסוים רק אם הסטודנט אינו רשום כבר לקורס. על השיטה להחזיר true אם ההוספה התבצעה בהצלחה ו- false אחרת.
ניתן להניח כי הקלט שונה מ- null.

6. **public** double averageGrade():
השיטה אינה מקבלת פרמטרים. על השיטה לחשב את הממוצע המשוקלל של הסטודנט בכלל הקורסים בהתאם למספר הנק"ז של כל קורס. אם לסטודנט אין ציון באף קורס, הממוצע המשוקלל הוא 0.

7. **public** boolean addGrade(Course course, int grade):
השיטה מקבלת כפרמטר קורס וציון ומנסה להוסיף את הציון של הסטודנט בקורס לרשימת הציונים. אם הסטודנט רשום לקורס, הציון נוסף לרשימה, הקורס מוסר מרשימת הקורסים אליהם הסטודנט רשום ומוחזר הערך true. אם הסטודנט אינו רשום לקורס הציון לא יתווסף לרשימה והשיטה תחזיר את הערך false.
ניתן להניח כי הפרמטר course שונה מ- null.

8. **public** int setGrade(Course course, int grade):
השיטה מקבלת כפרמטר קורס וציון ומנסה לעדכן את הציון של הסטודנט בקורס. ניתן לעדכן ציון של סטודנט בקורס רק אם כבר קיים לו ציון בקורס זה. אם תנאי זה מתקיים, על השיטה לעדכן את הציון ולהחזיר את הערך הקודם של הציון.
אם אין לסטודנט ציון בקורס, יש לזרוק חריגה מסוג IllegalArgumentException.
ניתן להניח כי הפרמטר course שונה מ- null.

9. **public** Grade getCourseGrade(Course course):
מחזיר את הציון של סטודנט בקורס מסוים, מחזיר null במידה והסטודנט לא השלים את הקורס.

10. **public int** compareTo(Student other) : השיטה של הממשק Comparable<Student> מחילה יחס סדר על הסטודנטים, ע"פ הציון הממוצע של הסטודנט (כלומר אם הממוצע של סטודנט A גדול מהממוצע של סטודנט B, $A > B$), אם הציון הממוצע שווה, ההשוואה תתבצע על פי תעודת הזהות.

המחלקה StudentManagementSystem: (25 נקודות)

יש להגדיר את הטיפוס StudentManagementSystem במחלקה בשם StudentManagementSystem בקובץ StudentManagementSystem.java.

על מערכת ניהול הסטודנטים לאחסן את רשימות התארים, הקורסים, הסטודנטים ועוזרי ההוראה הקיימים במערכת. מספריהם של הסטודנטים, הקורסים, התארים ועוזרי ההוראה במערכת **אינם מוגבלים**. כדי לממש תכונות אלו, נשתמש בשדות הבאים:

- רשימת הסטודנטים List<Student>
- רשימת התארים List<Degree>
- רשימת הקורסים List<Course>
- שדה שמציין את הציון המינימלי שיש להשיג על מנת לקבל ציון עובר בקורסים.

נשים לב שבאמצעות מחלקה זו אנו מבצעים את כל הפעולות של המערכת.
הוסיפו למחלקה ארבעה שדות מתאימים.

יש להשלים את הבנאי: **public** StudentManagementSystem(int failThreshold). הבנאי מקבל את ציון הסף עבור כלל הקורסים במערכת, ומאתחל את שדות המחלקה.

יש להשלים את השיטות הבאות:

1. **public** List<Student> getStudents():
השיטה מחזירה את רשימת הסטודנטים.

2. **public** List<Degree> getDegrees():
השיטה מחזירה את רשימת התארים.

3. **public** List<Course> getCourses():
השיטה מחזירה את רשימת הקורסים.

4. **public** int getFailThreshold():
השיטה מחזירה את סף המעבר.

5. **public** boolean addStudent(Student student):
השיטה מקבלת כפרמטר סטודנט ומוסיפה אותו אל רשימת הסטודנטים במערכת השיבוץ, אם לא קיים במערכת סטודנט השווה (עפ"י הגדרת השיטה equals) לסטודנט שהתקבל כפרמטר. בנוסף המערכת בודקת האם קיים במערכת תואר מהסוג אליו הסטודנט שייך, אם לא, ההוספה תיכשל.
על השיטה להחזיר true אם ההוספה התבצעה בהצלחה ו- false אחרת.
ניתן להניח כי הקלט שונה מ- null.

6. **public** boolean addCourse(Course course):
השיטה מקבלת כפרמטר ערך מטיפוס Course ומוסיפה אותו אל רשימת הקורסים במערכת השיבוץ, אם לא קיים במערכת קורס השווה (עפ"י הגדרת השיטה equals) לקורס שהתקבל כפרמט. בנוסף, המערכת מוודאת כי כל קורסי הקדם כבר קיימים במערכת..
על השיטה להחזיר true אם ההוספה התבצעה בהצלחה ו- false אחרת.
ניתן להניח כי הקלט שונה מ- null.

7. **public** boolean addDegree(Degree degree):
השיטה מקבלת כפרמטר ערך מטיפוס Degree ומוסיפה אותו אל רשימת התארים במערכת השיבוץ, אם הוא לא קיים במערכת. במידה וישנם קורסים בתואר (חובה ובחירה) שאינם קיימים במערכת, השיטה תיכשל.
ניתן להניח כי הקלט שונה מ- null.

8. **public** List<Course> getMissingPreCourses(Course course, int studentId):
השיטה מקבלת קורס ותעודת זהות של סטודנט ומחזירה את רשימת קורסי-הקדם שהסטודנט נדרש להשלים על מנת להירשם לקורס בהצלחה. על הרשימה להיות ממוינת ע"פ ה-Comparable שהוגדר ב-Course.
ניתן להניח כי הקלט שונה מ-null.

9. **public** boolean register(int studentId, Course course):
השיטה מקבלת שני פרמטרים, תעודת זהות של סטודנט וקורס ומוסיפה את הקורס לרשימת הקורסים אליהם רשום הסטודנט. על השיטה לבדוק שהקורס והסטודנט קיימים במערכת, כמו כן עליה לוודא כי הסטודנט עבר את כל קורסי הקדם לקורס ועליה לוודא כי הקורס קיים בתואר שהסטודנט עושה (בתור קורס בחירה או קורס חובה).
בנוסף, לפני ההרשמה לקורס, המערכת תבדוק אם הסטודנט אינו עשה כבר את הקורס. אם הוא כבר עשה את הקורס, המערכת תאפשר להירשם אם יש לו ציון נכשל (ע"פ ה-failTreshold המוגדר) בקורס.
על השיטה להחזיר true אם הרישום התבצע בהצלחה ו- false אחרת.
ניתן להניח כי הקלט שונה מ- null.

10. **public** boolean addGrade(Course course, int studentId, int grade):
הפונקציה מקבלת Course, תעודת זהות של סטודנט, וציון, ומזינה לסטודנט את הציון. במידה והסטודנט לא קיים / לא רשום לקורס / הקורס לא קיים / הסטודנט כבר קיבל ציון עובר, המערכת תחזיר false. במידה והמערכת הצליחה להזין ציון, יחזור true.

11. **public** List<Student> closeCourse(Course course, List<Pair<Integer, Integer >> grades):

הפונקציה מקבלת Course ורשימה של Pair (המחלקה Pair מצורפת לעבודה), כאשר כל זוג מכיל משתנה int ומשתנה int, שכוללים ת"ז וציון של סטודנט. על המערכת לתת לכל סטודנט את הציון ע"פ הרשימה. השיטה תחזיר רשימה של כל הסטודנטים שנכשלו בקורס.

12. **public** boolean closeDegree(int studentId, int year):

השיטה מקבלת תעודת זהות של סטודנט, ושנת סיום. ומנסה לסגור לו את התואר עם השנה הנתונה, על מנת לסגור את התואר, היא קודם בודקת את הדברים הבאים:

- הוא ביצע את כל קורסי החובה ע"פ דרישות התואר.
- הוא ביצע מספיק קורסי בחירה כדי להגיע לכמות נק"ז הדרושה בתואר.
- הוא לא רשום כרגע לשום קורס.

שימו לב, בשביל שקורס ייספר (לנק"ז, או לבדיקה אם עשה) על הסטודנט להיות בעל ציון עובר (ע"פ failTreshold) לפחות בקורס (מותר למשל לעשות קורס בחירה ולהיכשל, כל עוד יש קורסי בחירה אחרים שמספקים את הנק"ז).

אם אחד מהתנאים לא התקיים, או שהסטודנט לא קיים במערכת, הפונקציה תחזיר false.

במידה והשיטה הצליחה לסגור לו את התואר, שנת הסיום של הסטודנט תתעדכן ל-year. והיא תחזיר true.

משימה 3: הוספת פונקציונליות לתשאול, אחזור וניתוח מידע. (20 נקודות)

השלימו מחלקה StudentManagementSystem את השיטות הבאות:

1. **public** List<Student> getFirstKStudents(Degree degree, int year, int k):
השיטה מקבלת 3 פרמטרים: סוג התואר, שנת סיום תואר ו-k. על השיטה להחזיר רשימה ממוינת ע"פ יחס הסדר שנקבע על ידי השיטה CompareTo. הרשימה תכיל את k הסטודנטים המצטיינים בתואר degree שסיימו בשנת הלימודים year. סיימו כלומר סגרו את התואר. שימו לב שאם יש פחות מk סטודנטים, יש להחזיר את כל הרשימה.
אם k אינו מספר טבעי, year אינו מספר טבעי או ש-degree הוא null, על השיטה לזרוק חריגה מהטיפוס IllegalArgumentException.

2. **public** List<Student> getFailStudents(Course course) :
השיטה מקבלת קורס. על השיטה להחזיר רשימה של סטודנטים שנכשלו בקורס זה, כאשר ציון נכשל הוא ציון המוגדר במחלקה (ע"פ הfail threshold המוגדר).

3. **public** List<Student> getRegisteredStudents(Course course) :
השיטה מקבלת קורס ומחזירה רשימה המכילה את הסטודנטים הרשומים אליו.

4. **public** List<Course>nextAvailableCourses(int studentId) :
השיטה מקבלת תעודת זהות של סטודנט ומחזירה את הרשימה של כל הקורסים שהסטודנט יכול ללמוד (אם כל קורסי הקדם שלהם נלקחו על ידי הסטודנט, או שאין להם קורסי קדם כלל) כחלק מתכנית הלימודים של הסטודנט. על הרשימה להיות ממוינת ע"פ הגדרת Comparable ב-Course.

5. **public** List<Course>getMissingCourses(int studentId):
השיטה מקבלת תעודת זהות של סטודנט, ומחזירה את רשימת קורסי החובה שנשאר לסטודנט ללמוד.

עבודה נעימה :)