תוכנית לימודים במקצוע מדעי המחשב
לחטיבה העליונה
בכל המגזרים
Ver. 2.62

מבוא לתכנית הלימודים

מדעי המחשב הינו תחום מרכזי בעל השלכות רבות על תחומים אחרים. מדעי המחשב נתפס כשפת הטכנולוגיה המשמשת לתיאור והבנת מבני מידע וידע, קשרים ותהליכים. כשפה, מדעי המחשב מאפשרת פתרון בעיות, ייצוג ידע ופורמליזציה של תהליכים ולכן תומכת בהבנת טכנולוגיה ובפיתוח מדעי-טכנולוגי ועקרונותיה משמשים כיום לעבודה מדעית והנדסית בתחומים רבים.

בנוסף לאתגר המחשבתי שמציבים לימודי מדעי המחשב והפוטנציאל שלהם לקידום כישורי חשיבה מסדר גבוה, למידת מדעי המחשב בחטיבת הביניים עשויה להעמיק את הבנתם של התלמידים תופעות שונות להם הם עדים בחייהם, ולתרום ללימודיהם בתחומים אחרים. בנוסף, היות ומדעי המחשב ניצב בבסיסן של רבות מההתפתחויות הטכנולוגיות להן אנו עדים בעשורים האחרונים, הבנת התחום והשלכותיו עשויה להעלות גם את מודעותם החברתית, האתית והתרבותיות של תלמידים הלומדים אותו.

הגדרה כללית של מיומנויות ייחודיות ורלבנטיות לתחום הדעת
פתרון בעיות והיכולת לממשן באמצעות שפת מחשב תוך תכנון, בניית ובדיקת אלגוריתם הן מהמיומנויות הייחודיות לתחום מדעי המחשב.
מידול והפשטה הן מיומנויות נדרשות ונרכשות בתחום מדעי המחשב והתלמידים יוכלו ליישמם גם בתחומי ידע רבים מחוץ למדעי המחשב.

הגדרת מטרות התכנית בתחום אסטרטגיות החשיבה
אחת ממטרות הלימוד במדעי המחשב בחטיבה העליונה, הינה חשיפת התלמידים לתהליכי פתרון בעיות החל מניתוחן ועד למימושן באמצעות תכנית מחשב. לשם כך על התלמידים ליישם תהליכי חשיבה הן ברמת הפשטה גבוהה, המתייחסים להבנת הבעיה, והן ברמות הפשטה נמוכות יותר בדרך לפתרונה של הבעיה. בכך, מאפשרת תוכנית הלימודים פיתוח כישורי חשיבה מסדר גבוה, מודעות לרמות הפשטה וכן כישורי רפלקציה להבנת תהליך הפתרון וחשיבה ביקורתית לצורך הערכת הפתרון.

פיתוח תכנית הלימודים
פיתוח תכנית הלימודים נעשה על-ידי ועדות המקצוע למדעי המחשב בראשות פרופ׳ יהודית גל עזר, פרופ׳ שמעון שוקן וועדת התכנית בראשות פרופ׳ אורית חזן.

יסודות

(יחידות לימוד 2+1)

שתי יחידות הלימוד הראשונות מחולקות לשישה פרקים:

	פרק מספר	שם הפרק	שעות לימוד
יסודות: יחידות לימוד 1+2	1	מבוא	10
	2	מושגי יסוד	30
	3	ביצוע מותנה	10
	4	ביצוע חוזר	40
	5	מבני נתונים סדרתיים	46
	6	תכנות מונחה עצמים	44
		סה"כ	180

כשליש משעות הלימוד מוקדשות לתרגול במעבדה.

פרק 1: מבוא

<u>יעדים</u>: הכרה ראשונית (בפרט לתלמידים שלא למדו מדעי המחשב בחטיבת הביניים) של תחום מדעי המחשב והשפעתו על תחומי ידע אחרים. חשיפה ראשונית לחשיבה אלגוריתמית וכתיבת תכניות. הכרה ראשונית של מושג העצם ותכנות מבוסס עצמים.

<u>תכנים</u>

- הדגמת חשיבותו ומקומו של מקצוע מדעי המחשב ע"י דיון באתגרים חישוביים מתחומי ידע שונים.
- הכרת משימות חישוביות פשוטות: ניתוח המשימה, ניסוח אלגוריתמי של פתרון אפשרי.
- הכרת המושג שפת תכנות.
- הכרת מושג התכנית: קריאה, כתיבה, הרצה, בדיקה, ותיקון תכניות פשוטות.
- מחלקה ופעולת main כמסגרת בסיסית לכתיבת תכנית.
- הכרת מושג העצם.
- קריאה והבנה של ממשק פשוט של מחלקה קיימת, לצורך יצירת עצמים וזימון פעולות (methods) על עצמים.
- יצירת עצמים באמצעות פקודת new.
- זימון וביצוע פעולות על עצמים.

<u>מטרות ביצועיות</u>[1]

- התלמיד יקרא תכנית פשוטה ויסביר את דרך פעולתה במילים שלו.
- התלמיד יקבל משימה מילולית פשוטה; התלמיד יתכנן ויכתוב אלגוריתם שפותר אותה בשפה כללית (כלומר סיפור – השתלשלות האלגוריתם).
- התלמיד יממש את האלגוריתם ע"י כתיבת תכנית והרצתה.
- התלמיד יתאר את המבנה הבסיסי של תכנית: מחלקה, פעולת main.
- התלמיד יכתוב תוכנית שיוצרת ומפעילה עצמים פשוטים ע"י תכנות בעזרת ממשק מחלקה נתון.
- התלמיד יפעיל פעולות בסיסיות של סביבת הפיתוח בה משתמשים בתכנית הלימודים ויוכל לפתח ולהריץ בה תכנית פשוטה (פתיחת קובץ תכנית, שמירת תכנית, עריכה, הרצה פקודה אחר פקודה).

מכן ואילך, המונח *"התלמיד יממש"* כוונתו *"התלמיד יכתוב, יתקן, וירץ תכנית מחשב"*.

<u>דרכי הוראה</u>

מטרת הפרק הראשון בתכנית הלימודים היא לתת הכרה ראשונית עם תחום מדעי המחשב, וליצור גירוי ורצון להמשך הלמידה. לכן, יש להציג את החומר באופן מושך ומסקרן, תוך חשיפה מינימלית לפרטים טכניים ומושגי תכנות (מימוש אלגוריתמים) שיידונו בפרקים הבאים.

תכנית הלימודים של "יסודות" (2 י"ל) מתחלקת לששה פרקים שרק האחרון שבהם נקרא "תכנות מונחה עצמים". סגנון העבודה בחמשת הפרקים הראשונים בתכנית הוא "תכנות מבוסס עצמים". זוהי אבחנה חשובה: בתכנות מבוסס-עצמים אנו עובדים עם מחלקות קיימות תוך שימוש בממשקי המחלקות, ללא

[1] ככלל, הנושאים שמופיעים בסעיף "מטרות ביצועיות" מתארים שאלות אפשריות בבחינות.

התייחסות כלשהי לעיצוב או למימוש המחלקות הללו. בפרק 6, שכותרתו כאמור "תכנות מונחה עצמים", נפתח את ה"קופסאות השחורות" הללו ונדון במימוש מחלקות.

ככלל, אפשרי אך לא חובה, לעבוד עם עצמים גרפיים קיימים, כגון הצב Turtle, הרובוט Karel, או מחלקת Bucket בהקשר של העברת נוזלים בין דליים. העבודה עם עצמים גרפיים מאפשרת הדגמת וכתיבת קטעי קוד קצרים ופשוטים שתוצאותיהם נראות באופן מיידי וחזותי. למשל, הזזה של צב או רובוט על המסך תוך ביצוע מתוכנן של פעולות שונות. זוהי דרך טובה ומהנה לממש ולהמחיש אלגוריתמים פשוטים – נושא שנעסוק בו בפרקים הבאים בתכנית. אפשר כמובן לדון גם בעצמים אחרים. ככלל, יש לבחור עצמים שמעניין להפעיל אותם, לדוגמא כדי לפתור חידה או לצייר ציור על המסך.

יש להדגיש את הבנת מושג האלגוריתם ודרך מימושו. לדוגמא, תכנון ומימוש אלגוריתם פשוט שמשתמש בצב כדי לצייר על המסך צורות גיאומטריות פשוטות, ציור שם התלמיד בעברית או באנגלית, וכדומה. הדגש צריך להיות יותר על איך להפעיל את הצב בצורה מושכלת ומתוכננת כדי לפתור את המשימה הנתונה, ופחות על הצב עצמו. הצב הוא מושג מופשט, וכך צריך להתייחס אליו. לכן, בשלב זה אין לייחס לעצם ולמחלקה ממנה העצם נוצר חשיבות מיוחדת.

את מושג העצם, פעולת new, ופעולות אחרות על עצמים יש להציג כמושגים מופשטים שניתן בהחלט להשתמש בהם בלי להבין כיצד הם ממומשים. זוהי דוגמא ראשונית טובה להפרדה בין שימוש למימוש, עיקרון שנדון בו בהרחבה בהמשך התכנית.

כדי לעבוד מול עצם קיים בצורה מושכלת, התלמיד חייב לקרוא ולהבין את ממשק המחלקה הרלבנטית. כלומר, במקום להסביר בצורה חלקית ולא מדויקת מה העצם יכול או לא יכול לעשות, יש להציג ממשק מחלקה כתוב ומדויק. יחד עם זאת, בשלב זה אין לחשוף את התלמיד אל API קונבנציונלי שמכיל הרבה מושגים ופעולות שלא נעשה בהן שימוש בשלב זה. במקום זאת, יש לכתוב על הלוח או על מסמך שיינתן לתלמיד רשימה פשוטה של כל הפעולות הרלבנטיות שניתן להפעיל על העצם, ולהקפיד שהרשימה תימצא מול עיני התלמיד בכל מהלך הדיון והעבודה. הכוונה לתיעוד לא פורמלי שיש להעלים ממנו בשלב זה מושגים כגון הרשאות גישה, ירושה ושאר פרטים שמופיעים ב- API's רגילים שבשלב זה של התכנית ייצרו רק בלבול.

תיעוד תכניות הוא כמובן נושא חשוב, אך אין סיבה לכפות אותו על קטעי קוד פשוטים וברורים מאליהם. אנו ממליצים לא לדון בתיעוד עד פרק 3, שבו מתחילים לכתוב תכניות שרמת מורכבותן מצדיקה תיעוד.

יש לערוך סקירה קצרה של עולם שפות התכנות, ולהתמקד בשפת התכנות בה נשתמש: היסטוריה קצרה של התפתחות השפה, והשימוש הנרחב בה בתעשייה ובאקדמיה.

לכל אורך תכנית הלימודים יש להשתמש בשפות העברית והאנגלית בצורה מושכלת ומשולבת. ראו הערה בנושא זה בנספח "הערות דידקטיות כלליות".

דרכי הערכה[2]

בחינה במעבדה:

- כתיבת תכנית פשוטה והרצתה בתוך מסגרת נתונה של מחלקה ופעולת main.
- יצירת והפעלת עצמים קיימים ע"י זימון פעולות תוך שימוש בתיעוד ממשק נתון.

בחינה עיונית:

- הבנה וכתיבה של אלגוריתם פשוט ותכנית פשוטה, בלי שימת דגש על תחביר השפה.
- מעקב אחר תכנית פשוטה שיוצרת ומפעילה עצם בעזרת ממשק מחלקה (לא פורמלי) נתון.

חלוקת שעות

נושא	שעות
הדגמת חשיבותו ומקומו של מדעי המחשב	1
הבנת אלגוריתמים ותכניות	2
כתיבת תכניות	7
סה"כ שעות:	10

[2] ככלל, "דרכי הערכה" מתייחסות לפעולות פורמליות שהן מעבר להכנת שיעורי בית ובחנים תקופתיים

פרק 2: מושגי יסוד בתכנות

יעדים: הכרת מושגי יסוד בתכנות: משתנים, טיפוסי נתונים, ביטויים חשבוניים, קלט / פלט, חלוקת קוד לפעולות עזר. המשך עבודה עם עצמים ממחלקות מוכנות.

תכנים

- אלגוריתם
- הכרת המושג משתנה/תכונה.
- הכרת המושג מחלקה, תכונות ופעולות
- טיפוסי נתונים בסיסיים: שלם (int), ועשרוני (double).
- תחומי הערכים של טיפוסי הנתונים לעיל.
- פקודות השמה.
- הגדרת ואתחול משתנים.
- אופרטורים חשבוניים: חיבור, חיסור, כפל, חילוק, שארית.
- פעולות מתמטיות בסיסיות: abs, max, min, pow, round, sqrt,random
- ביטויים חשבוניים: סדר קדימויות האופרטורים ותפקיד הסוגריים.
- המרה בסיסית בין טיפוסי הנתונים int ו- double
- פעולות ככלי לעידון וחלוקת משימות.
- הגדרת וכתיבת פעולת עזר (private method).
- הגדרת פרמטרים, והעברה לפי ערך (call by value).
- ערך מאוחזר (return value).
- שגיאות לוגיות, שגיאות תחביר, שגיאות בזמן-ריצה.
- פעולות קלט / פלט פשוטות ושרשור מחרוזות.
- טבלת מעקב.

מטרות ביצועיות

- התלמיד יסביר את המושג "אלגוריתם" והקשר שלו לכתיבת תכנית.
- התלמיד יגדיר את סוגי ושמות המשתנים/תכונות שמתאימים למשימה מילולית נתונה.
- התלמיד יכתוב ויעקוב אחר פעולות השמה.
- התלמיד יסביר את הצורך והשוני בין טיפוסי הנתונים int ו- double.
- התלמיד יחשב את הערך של ביטויים חשבוניים נתונים.
- התלמיד יתרגם תיאור משימה חישובית מילולית לביטוי חשבוני תקין.
- התלמיד יממש משימות חישוב שכוללות ניסוח ביטויים חשבוניים.
- התלמיד יסביר (באופן בסיסי בלבד) את הצורך בהמרה בין טיפוסי נתונים שונים.
- התלמיד יסביר את הצורך בחלוקת משימה מורכבת למשימות קטנות.
- התלמיד יסביר את הצורך בפעולות ככלי לעידון וחלוקה למשימות.

- התלמיד יקבל קטע קוד מסורבל וידע לחלקו לפעולת עזר אחת או יותר.
- התלמיד ידע להסביר את המונח "העברה לפי ערך".
- התלמיד יסביר את מנגנון הזימון של פעולות: העברת פרמטרים, איחזור ערך, השתלבות הערך המאוחזר בביטוי שכולל את זימון הפעולה.
- התלמיד יעקוב אחר ערכי משתנים בזמן ריצה.
- התלמיד יקבל ממשק של מחלקה, ותוכנית היוצרת עצמים מהמחלקה הנתונה ומשתמשת בהם. התלמיד ידע לבצע מעקב על תכנית זו ולציין את הפלט המתאים.
- התלמיד יקבל ממשק של מחלקה, ותוכנית היוצרת עצמים מהמחלקה הנתונה ומשתמשת בהם. התלמיד ידע לבצע שינויים בתוכנית לצורך קבלת פלט נדרש.
- התלמיד יבצע פעולות קלט פלט

<u>דרכי הוראה</u>

פרק זה כולל מושגים חדשים ופרטים טכנים רבים. מומלץ להציג את הנושאים הללו בהקשר של משימה מעשית כוללת ומוכרת. בפרט, חשוב לקשר את החומר הנלמד לדוגמאות אותן פגשנו בפרק 1, כמו למשל הפעלה של הצב לציור צורות גיאומטריות שמבוססות על חישוב ביטויים חשבוניים. כמובן שניתן ורצוי להשתמש ביותר מדוגמא יישומית אחת. לדוגמא, דיון במערכת מידע לניהול מידע על תלמידים מוביל באופן טבעי לצורך בטיפוסי נתונים שונים, ביטויים חשבוניים, קלט/פלט, וכדומה. במקום לדון במושגים הללו באופן תאורטי, רצוי כמובן להציגם כמענה לצורך יישומי ברור, ולהציג את השימוש בהם בהקשר של פתרון בעיות מוכרות.

ערכים: יש לדון במושג המופשט "ערך", ולציין שלכל ביטוי חשבוני (או תת-ביטוי בתוך ביטוי נתון) יש ערך. יש להציג ביטויים חשבוניים פשוטים הכוללים קבועים בלבד (ללא משתנים) ולחשב את הערכים שלהם. יש להסביר שבניגוד למספרים במתמטיקה, גודל הערכים שניתן לייצג במחשב מוגבל בגלל מגבלות פיזיות של גודל זיכרון המחשב.

טיפוסי נתונים: יש להציג את טיפוסי הנתונים שלם (int) ועשרוני (double) ולדון בתחומי הערכים הרלבנטיים שלהם. בגלל שתחומי הערכים שהמחשב מסוגל לייצג מוגבלים, יכולים להיווצר אי-דיוקים וטעויות בייצוג ערכים מספריים מסוימים. בשלב זה לא ננתח את אי הדיוקים הללו, אך יש להיות מודעים אליהם.

משתנים: בפרק זה התלמיד מתוודע לראשונה למושג המשתנה. יש להדגיש שטיפוס ושם המשתנה נקבעים עם הגדרתו ונשארים ללא שינוי לאורך זמן ; לעומת זאת, ערך המשתנה משתנה בדרך כלל במהלך ביצוע התכנית, ומכאן שמו. יש להסביר באופן כללי את מודל האחסון של משתנה בזיכרון המחשב, כדי לחדד את ההבנה של משתנה כתא שמכיל ערך שיכול להשתנות.

השמה: יש לחדד את מושג ההשמה כסוג של העתקה ולא של העברה. למשל, אפשר להסביר שהמטפורה של העברת נוזל מכלי לכלי אינה מתאימה להשמה.

המרה: יש לדון במקרים בהם מתבצעת המרה "אוטומטית" של טיפוסי נתונים, כמו למשל חיבור מספר int עם מספר double.

ביטויים חשבוניים: יש להציג את המושג "ביטוי חשבוני" כבנוי מרכיבים בסיסיים (בשלב זה, משתנים וקבועים) עליהם ניתן להפעיל פעולות חשבוניות כגון חיבור, חיסור, כפל, חילוק. יש להדגיש שכל ביטוי חשבוני מחשב ומייצג ערך חשבוני כלשהו.

פעולות מתמטיות: יש לדון ולהדגים שימוש בפעולות הבאות: abs, max, min, pow, random, round, sqrt. יש לדון בביצוע חלוקה של מספרים שלמים: מנה ושארית. יש להרחיב את הגדרת המושג "ביטוי חשבוני" כביטוי שבנוי ממשתנים, קבועים, פעולות חשבוניות, ופעולות מתמטיות.

פעולות ספרייה: לטובת השימוש בפעולות המתמטיות שהוזכרו לעיל, יש לדון בקצרה בתחביר זימון הפעולה הסטטית ClassName.methodName. יחד עם זאת, יש לעשות זאת בלי להזכיר את המושגים "סטטי" או "ציבורי", שהם חסרי משמעות בשלב זה. במקום זאת ניתן לומר שהשמות המלאים של הפעולות איתן נעבוד בשלב זה מורכבים משם המחלקה ושם הפעולה. בהקשר הזה, יש להתייחס למחלקת Math כאל ספריה שמאגדת ומציעה גישה לאוסף של פעולות מתמטיות שימושיות.

העברת פרמטרים: יש לדון בהעברת פרמטרים לפי ערך (call by value), ולהתייחס לפרמטרים הללו כאל הערכים עליהם אנו מפעילים את הפעולה. יש להדגיש ולהדגים שפרמטר מטיפוס מסוים יכול לקבל כל ביטוי שערכו הוא מאותו טיפוס. תבנית העברת פרמטרים לפי הפנייה (call by reference) תילמד בפרקים מאוחרים יותר בתכנית, ומומלץ לא להזכירה או לדון בה בשלב זה.

פעולות מחלקה: השימוש בפעולות מתמטיות, כמו גם משימות ציור שונות בעזרת הצב, מספקים מסגרת טובה לפתיחת דיון בפעולות (methods): כל פעולה במחלקת Math מממשת פעולה מתמטית מסוימת. יש לחזור ולהדגים פעולה כלשהי, כגון ()sqrt, ולהסביר שהשימוש בה הוא למעשה זימון פעולה שמתכנת אחר כתב במחלקה אחרת. יש להסביר שגם אנו יכולים לכתוב פעולות כאלה, כרצוננו. מכאן ניתן לעבור לדיון ראשוני ביתרונות הגישה המודולרית ותרומתה לתכנון, שימוש חוזר, הבנה, בדיקה, ותחזוקת קוד.

פעולות: כאמור, "פעולה" היא רכיב תכנותי שנועד לאפשר עידון וחלוקת קוד למשימות. יש לדון בתחביר של פעולות: חתימה, כולל הסיווג static שאנו משתמשים בו בלי להסבירו בשלב זה, סוגריים מסולסלים, וגוף הפעולה. בשלב זה של תכנית הלימודים, כל הפעולות שנממש תהיינה מסוג static. יש להציג את הסיווג הזה כפתיח נתון לכתיבת פעולות שמשמעותו תילמד בעתיד. יש לדון בבחירה מושכלת של שם הפעולה, שמות הפרמטרים, טיפוסי הפרמטרים, טיפוס הנתונים של הערך המאוחזר (כולל פעולות מטיפוס void), ופקודת return. יש להדגיש את שהשימוש בפעולות חשוב הן לצורך עידון והן לצורך שימוש חוזר.

מחרוזות: כדי למנוע בלבול, טיפוסי הנתונים char ו- String אינם כלולים בתכנית הלימודים של יחידות 1-2. נושאים אלה יידונו ביחידת הלימודים הרביעית. יחד עם זאת, הפרק הנוכחי כולל הכרה ראשונית של טיפוס הנתונים String ופעולת השרשור + של מחרוזות. חומר זה נלמד באופן מינימלי בלבד שמטרתו לאפשר קליטה והדפסה של ערכים.

קלט / פלט: יש ללמד את פעולת הפלט הפשוטה ביותר לצורך הדפסת ערכי משתנים ומעקב אחר ביצוע פעולות חשבוניות. באופן דומה, יש ללמד דרך פשוטה ככל האפשר לביצוע פעולות קלט. זוהי משימה בעייתית, כי בשפות הנלמדות בתכנית הלימודים פעולות קלט/פלט נעשות ע"י עבודה מול מחלקות, והתלמיד עדיין לא בשל להבין את המבנה ודרך הפעולה של המחלקות הללו. אי לכך, יש להתייחס לפעולות קלט / פלט כאל קופסאות שחורות, ולהסביר שבשלב זה נשתמש בהן בלי להתעמק באופן בו הן ממומשות.

טבלת מעקב: יש להגדיר מהי טבלת מעקב: כלי עזר שמאפשר לעקוב אחר קטע קוד ולבדוק את נכונותו. הטבלה כוללת מספר עמודות, אחת עבור כל משתנה פשוט (מסוג int או double) אחריו אנו מעוניינים לעקוב, עמודה עבור הפלט ובהמשך, לכשילמדו תנאים יתווספו לטבלה עמודות בעבור התנאים השונים.

שורות הטבלה מציינות את השתנות ערכי המשתנים אחריהם אנו עוקבים. בנוסף, עבור כל משתנה מסוג עצם תוגדר הפנייה לעצם ויפורטו תכונות העצם וערכיו.

דרכי הערכה

בחינה במעבדה:

- כתיבת תכנית שכוללת עבודה עם משתנים וביטויים חשבוניים והרצתה.
- כתיבת תכנית שכוללת חלוקה לפעולות פרטיות והרצתה.
- זיהוי שגיאות תחביר ותיקונן בקטעי קוד נתונים.

בחינה עיונית:

- ניתוח ומעקב אחר קטעי קוד נתונים שמכילים את החומר הנלמד.
- תרגום משימות חישוב מילוליות לאלגוריתמים וקטעי קוד שכוללים את החומר הנלמד.

חלוקת שעות

נושא	התנסות	עיונית	סה"כ שעות
משתנים, טיפוסי נתונים			3
ביטויים חשבוניים			6
פעולות ממחלקת Math			6
מודולריות וכתיבת פעולות			12
קלט / פלט			3
סה"כ שעות:			30

פרק 3: ביצוע מותנה

<u>יעדים:</u> הבנת ביטויים בוליאניים, מושג התנאי, הצורך בביצוע מותנה, מבנה הבקרה if, ותפקידם בהקשר הכללי של משימה חישובית ומימושה; העמקת ההבנה של עבודה עם משתנים.

<u>תכנים</u>

- טיפוס הנתונים boolean.
- פעולות בוליאניות
- יחסים: שווה, שונה, גדול, קטן, גדול או שווה, קטן או שווה.
- בעייתיות השימוש ביחסים == ו- =! בהקשר של ערכים עשרוניים.
- פעולות בוליאניות (not, and, or) וטבלאות האמת שלהן.
- ביטויים בוליאנים פשוטים, מורכבים, וסדר הפעולות הבוליאניות.
- ביצוע מותנה: if
- ביצוע מותנה: if .. else
- תקינות קלט, מסננת קלט פשוטה הכוללת תנאי בלבד.

<u>מטרות ביצועיות</u>

- התלמיד ייחשב את הערך של ביטויים בוליאניים נתונים ויבחין בסדר הפעולות הבוליאני.
- התלמיד יתרגם תיאור מצבים מילוליים לתנאים בוליאניים פשוטים ומורכבים. תנאי מורכב הוא תנאי שבנוי מכמה קשרים לוגיים עם או בלי סוגריים.
- התלמיד יתרגם משימות מותנות מילוליות לקטעי קוד שכוללים תצורות שונות של מבני if ו- if ... else.
- התלמיד יגדיר / יכתוב משפט תנאי מקונן בשלוש רמות לפחות.
- התלמיד יפצל משפטי תנאי ויצור תנאי עקיף הכולל או וגם ושלילה.
- התלמיד ישתמש בפעולות בוליאניות המוגדרות על עצמים
- התלמיד יכתוב פעולות בוליאניות וישתמש בערך החוזר מהן בתוך משפט if
- התלמיד יממש מסננת קלט פשוטה.

<u>דרכי הוראה</u>

יישומים: כמו בפרק הקודם, יש לדון במושגים שמוצגים בפרק זה בהקשר יישומי של משימה מעשית כלשהי. לדוגמא – עבודה עם עצם הצב, ייצוג פונקצית מדרגות, חישובי מס, חישוב מחיר פעילות קבוצתית כלשהי כפונקציה של מספר המשתתפים, גילאיהם, זמן הפעילות, וכדומה.

טיפוס הנתונים boolean וטבלאות אמת: יש להסביר שטיפוס הנתונים "בוליאני" מייצג שני ערכים בלבד שנהוג לכנותם true ו- false. יש להציג את טבלאות האמת של הפעולות not, and, or.

ביטויים בוליאנים בנויים גם מרכיבים לא בוליאנים, כגון המשתנה x בביטוי הבוליאני

(x > 15) || (x == 15). יחד עם זאת, חשוב להדגיש שביטוי בוליאני מחשב ומייצג ערך בוליאני. יש להשתמש בטבלאות אמת כדי לחשב את הערכים של ביטויים בוליאנים שמכילים מספר פעולות בוליאניות ושימוש בסוגריים.

השוואה של ערכים עשרוניים: חשוב להסביר את תופעת אי-הדיוק של ייצוג מספרים עשרוניים המאוחסנים בזיכרון המחשב. כתוצאה מאי-הדיוק הזה, השוואת מספרים עשרוניים בעזרת == או =! מובילה כמעט תמיד לשגיאה. הדיון הזה חשוב פחות מבחינה טכנית ויותר לצורך העמקת ההבנה של מוגבלות הייצוג של מספרים במחשב. בפרט, יש לחדד את השוני בין ערך חשבוני המיוצג במחשב למספר מופשט כפי שהוא מופיע במתמטיקה או בקוד של תכנית מחשב הכתובה בשפה עילית.

פעולות בוליאניות: יש להציג ולהסביר את השימוש בפעולות בוליאניות נתונות על עצמים על-ידי שימוש בעצמים המכילים פעולות בוליאניות. יש ללמד לכתוב פעולות סטטיות בוליאניות ולהשתמש בערכים החוזרים מפעולות אלו. יש לשלב במשפטי תנאי ערך בוליאני המתקבל משימוש בפעולה.

דוגמאות קוד: יש להציג ולהסביר משימות מילוליות וקטעי קוד שמבוססים על ביצוע מותנה. למשל, עבודה עם קטע קוד שמבצע קלט וניתוח מסלולי הריצה של הקוד עבור קלטים שונים. רצוי להתחיל עם תנאים בוליאניים פשוטים, ללא שלילה, ולעבור לתנאים מורכבים יותר בהדרגה.

יש לדון במקרים בהם צריך להשתמש ב if לעומת מקרים בהם נזקקים ל if ... else.

יש לשלב את הדיון בביטויים בוליאניים וביצוע מותנה בחומר הלימוד שנלמד בפרק הקודם. בפרט, יש להציג ולכתוב תכניות שעוסקות בקלט למשתנים, חישוב מותנה של ביטויים חשבוניים שונים, ופלט של התוצאה. לדוגמא, בסיום פרק זה יהיה לתלמיד כל הידע הדרוש לכתיבת תכנית שמחשבת את הפתרונות של משוואה ריבועית. זוהי דוגמא טובה לתרגיל תכנות שמסכם את רוב הידע שנצבר עד כה.

תנאים מקוננים: יש להציג ולהסביר את הצורך בקינון משפטי תנאי, כך שלא יתבצעו בדיקות מיותרות.. יש להציג בעיה כדוגמת מדרגות מס ולפתור אותה ע״י סדרה של משפטי תנאי, לעקוב אחר התכנית ולהדגיש את הבדיקות המיותרות המתבצעות בדרך פתרון זה. דיון זה צריך להוביל את התלמידים לצורך בקינון משפטי תנאי.

מסננת קלט: יש להציג את הדרישה לסינון קלט. יש לתת דוגמה בה עבור קלטים המקיימים תנאי מסויים מתבצעת סדרת הוראות אחת לעומת סדרת הוראות אחרת עבור קלטים שאינם מקיימים תנאי זה (לדוגמה: פלט שונה כתגובה לקלטים חיוביים / שליליים). כמו כן יש לתת דוגמה בה עבור קלט המקיים תנאי מסויים לא ניתן להמשיך בביצוע המשימה הנדרשת (למשל: חלוקה ב-0)

תיעוד: בפרק זה אנו עוסקים לראשונה בכתיבת קטעי קוד לא טריוויאלים. זהו שלב מתאים להציג את הצורך בתיעוד, ולהתחיל לדרוש מהתלמיד לתעד תכניות. התיעוד חייב להיות ברמת פירוט סבירה, ולא להפוך למעמסה בעיני התלמיד.

טבלת מעקב: יש להציג את דרך המעקב אחר ביצוע מותנה. יש להדגיש את הצורך במעקב אחר ערך התנאי בכל הוראת ביצוע מותנה. לצורך זה יש להגדיר עמודה עבור כל תנאי בקטע הקוד אחריו עוקבים.

<u>דרכי הערכה</u>

בחינה במעבדה:

- כתיבת תכנית פשוטה שכוללת קלט למשתנים, חישוב ביטוי חשבוני או בוליאני, ופלט של התוצאה והרצתה.
- כתיבת תכנית הכוללת תנאי מורכב ו/או משפטי תנאי מקוננים ו/או מסננת קלט והרצתה.

בחינה עיונית:

- חישוב ביטויים בוליאנים.

- תרגום תיאור בעיה מילולית לאלגוריתם שכולל ביצוע מותנה של פקודות.
- ניתוח ומעקב אחר אלגוריתם הכולל ביצוע מותנה ומסננת קלט.

<u>חלוקת שעות</u>

נושא	התנסות	עיונית	סה"כ שעות
ביטויים בוליאניים			2
ביצוע מותנה			8
סה"כ שעות:			10

פרק 4: ביצוע חוזר

<u>יעדים</u>: הבנה ומימוש של אלגוריתמים בסיסיים לביצוע חוזר; לתרגל ביצוע חוזר ככלי לעידון אלגוריתמים; להבחין בין כתיבה אלגוריתמית של לולאה לבין מימושה בשפת תכנות; הכרה בסיסית של המושגים נכונות ויעילות של אלגוריתמים; העמקת הידע של כתיבת, תיעוד, ותיקון תכניות; להשתמש בלולאות לאלגוריתמים המצריכים מנייה או צבירה.

<u>תכנים</u>

- ביצוע חוזר.
- לולאת for.
- לולאת while.
- מסננת קלט עקשנית
- משימות חישוב טיפוסיות: מונים, צוברים, ערכי קיצון.
- תנאי סיום.
- ביצוע אינסופי.
- ניתוח נכונות בעזרת טבלת מעקב.
- ניתוח יעילות ע״י ספירת איטרציות.
- קינון ושילוב מבני if, for, while
- תיעוד.

<u>מטרות ביצועיות</u>

- התלמיד יזהה בעיות שפתרונן דורש ביצוע חוזר.
- התלמיד יזהה בעיות שפתרונן מבוסס על צבירה, מניה, וחישוב ערכי קיצון (מינימום ומקסימום).
- התלמיד יבחין בין בעיה שפתרונה דורש ביצוע חוזר באורך ידוע מראש לבין בעיה שפתרונה דורש ביצוע חוזר מותנה.
- התלמיד יסביר את הצורך במבנה הבקרה for ואת דרך פעולתו.
- התלמיד יסביר את הצורך במבנה הבקרה while ואת דרך פעולתו.
- התלמיד יתרגם תיאור משימות מילוליות לאלגוריתמים שמבוססים על ביצוע חוזר.
- התלמיד יעקוב אחר קטעי קוד של ביצוע חוזר בעזרת טבלת מעקב.
- התלמיד יסביר את הצורך בתנאי הסיום של לולאה מותנית.
- התלמיד יזהה ויסביר לולאות בעלות ביצוע חוזר אינסופי.
- התלמיד יסביר במילים שלו את נכונותו של אלגוריתם נתון לביצוע חוזר.
- התלמיד יכתוב, יבדוק, ויריץ תכניות שמבוססות על מבני הבקרה for, while
- התלמיד יפתור בעיות הדורשות קינון ושילוב לולאות מסוגים שונים.
- התלמיד יתעד את התכניות שהוא כותב. על התיעוד לכלול הסבר לקטעי קוד מורכבים כגון לולאות.

<u>דרכי הוראה</u>

יש לפתוח בהצגת משימות ביצוע חוזר בנוסח מילולי. למשל: "בצע לכל איבר בקבוצה", "בצע n פעמים", וכדומה. חשוב לתאר ולפתור משימות כאלה על הלוח, ללא שימוש במחשבים. יש להדגיש שבאלגוריתם לביצוע חוזר, טקסט האלגוריתם נשאר קבוע, אך הוא משרה תהליכי ביצוע שאורכם (כולל אורך 0 או אינסוף) תלוי בגודל הקלט.

מומלץ להדגים את החומר הנלמד בפרק זה באמצעות **עבודה עם עצמים גרפיים מוכנים**. למשל, הזזת עצם מטיפוס Turtle על המסך כפועל יוצא של לולאות for ו- while. זוהי דרך מצוינת להמחיש באופן חזותי את משמעות הביצוע החוזר. בכל מקרה יש להדגים את החומר בשילוב עבודה עם עצמים. במקביל, יש להציג ולהסביר משימות חישוביות טיפוסיות כגון צבירה, מנייה, וחישוב ערכי קיצון. כמו כן מומלץ להציג ולממש תכניות שעוסקות בחישוב איברים וסכומים של סדרות חשבוניות וגיאומטריות שונות – נושאים שנלמדים במקביל בשיעורי מתמטיקה. יש להראות כי בעזרת הנוסחאות של סכום סדרה ואיבר כלשהו בסדרה, ניתן לחשב צבירה, מנייה וערכי קיצון בסדרה, ללא שימוש בלולאות.

נושא המערכים מוצג רק בפרק הבא, ששמו "מבני נתונים סדרתיים". יחד עם זאת, מורים שמעוניינים בכך יכולים בהחלט ללמד כבר עכשיו מבוא לעבודה עם מערכים. הכרה בסיסית של מערכים לא דורשת הבנה טכנית עמוקה, ומאפשרת כר נרחב לדיון באלגוריתמים לביצוע חוזר. בכל מקרה, גם אם חושפים את התלמידים למערכים, חשוב למקד את הלימוד בפרק זה יהיה ביצוע חוזר, ולא עבודה עם מערכים.

ביצוע חוזר באורך קבוע ולולאת for: יש להציג משימות אלגוריתמיות שפתרונן מבוסס על ביצוע חוזר באורך קבוע. לדוגמא, משימות צבירה, מניה, וחישוב ערכי סדרה חשבונית כלשהי. יש להציג מבני for לביצוע פקודה או פקודות מספר פעמים ידוע מראש. יש להסביר את תחביר ומשמעות מבנה הלולאה, ובפרט את תפקיד המציין (אינדקס). יש לדון בתנאי העצירה של לולאת for, ובערכי הקיצון של המציין. יש להסביר את סדר ביצוע שלושת החלקים של לולאת for ביחס לגוף הלולאה.

יש לתת דוגמאות של לולאות for משני סוגים: (א) המציין משמש מונה שסופר את צעדי הלולאה "מאחורי הקלעים", כמו למשל בהדפסת תבנית כלשהי n פעמים, (ב) המציין מופיע באופן מפורש בגוף הלולאה, כמו למשל בהדפסת סדרת הערכים n ... 0. חשוב להדגיש שהמציין מונה את פעולת הלולאה בכל מקרה, גם אם הוא לא בא לידי ביטוי מפורש בגוף הלולאה. יש להסביר שהמציין הוא למעשה משתנה לכל דבר ועניין.

ביצוע חוזר מותנה ולולאת while: יש להציג ולהסביר משימות מילוליות שפתרונן מבוסס על ביצוע חוזר שאורכו אינו ידוע מראש, כמו למשל צבירה, מניה, וחישוב ערכי קיצון של סדרת ערכים שמוזנת ע"י תהליך קלט כלשהו.

יש לדון בתכנון ומימוש "מסנן קלט", דהיינו קטע קוד שמוודא שהמשתמש מזין ערכים תקינים לפי קריטריון נתון מראש.

כשמציגים את מבני while ו- for יש להתעכב על תפקיד תנאי העצירה ולהבהיר שהתנאי ממומש ע"י ביטוי בוליאני – בדיוק אותם ביטויים שדנו בהם בפרק הקודם. ניתן לדון שוב בבעיות אלגוריתמיות שפתרונן מבוסס על צבירה ומניה, ולהשוות את דרך פתרונן בעזרת מבני for ו- while. יש לציין שככלל,

בתכנות ניתן לממש את אותה משימה בדרכים שונות שיניבו את אותה תוצאה. יחד עם זאת, ל- for יש בדרך כלל יתרון על while, כי הוא מביע את מבנה הבקרה בצורה יותר תמציתית וברורה.

יש להסביר שמבני for ו- while שקולים זה לזה בכוח הביטוי שלהם, ולהדגים זאת על ידי הצגת תרגום של כל אחד מהמבנים לרעהו.

יש להדגיש שבביצוע חוזר טקסט האלגוריתם נשאר קבוע, אך הוא משרה תהליכי ביצוע באורכים שונים (כולל אורך 0) התלויים בקלט

תנאי עצירה: בהינתן אלגוריתם או קטע קוד המבוסס על ביצוע חוזר, יש לחקור אם תנאי העצירה אכן מתקיים בשלב מסוים, ולהוכיח ע״י כך שהביצוע אכן מסתיים. יש להדגים קטעי קוד שביצועם אינו מסתיים.

טבלת מעקב: יש להראות את דרך המעקב אחר לולאות, לציין כי שורות הטבלה מייצגות את איטרציות הלולאה אחריה אנו עוקבים. יש להדגיש את הצורך להוסיף עמודה עבור מונה הלולאה (גם אינו מופיע במפורש) וכן להוסיף עמודה בעבור תנאי היציאה/כניסה מן הלולאה.

קינון ושילוב מבנים: לאחר שהתלמיד כתב מספר תכניות המבוססות כל אחת על מבנה בקרה ספציפי (for או while) יש לדון במשימות חישוביות שמצריכות קינון ושילוב לולאות. מומלץ להתחיל בקינון מבנים – למשל for מקונן ב- for, ולסיים בכתיבת תכניות שמשלבות קינון של מבני if, for, ו- while. כרגיל, יש להדגים את המבנים הללו בהקשר של משימות מוכרות ועצמים מוכרים לתלמיד. למשל, הדפסת תבניות גיאומטריות (מלבנים, משולשים) בעזרת שורות של כוכביות או באמצעות הצב, הרחבה של המשימה הזאת להדפסה שמימדיה נשלטים ע״י קלט מהמשתמש, וכדומה.

יעילות: הדיון בביצוע חוזר מספק מוטיבציה להתחלת דיון ביעילות של אלגוריתמים והתכניות שממממשות אותם. יש להגדיר מהי ״איטרציה״ של לולאה, ולבסס את הדיון ביעילות על ספירת מספר האיטרציות של האלגוריתם כפונקציה של גודל הקלט.

תרגול: בפרק זה התלמיד מתחיל לכתוב קטעי קוד מורכבים, ויש להקדיש לכך זמן תרגול ועבודה עצמית משמעותיים. מומלץ שהתרגול יעשה באמצעות תיקי עבודות ותכניות שהתלמיד יריץ בשיעורי הבית.

<u>דרכי הערכה</u>

בחינה במעבדה:

- כתיבת תוכנית הכוללת ביצוע חוזר פשוט.
- כתיבת תוכנית הכוללת ביצוע חוזר מקונן.
- כתיבת פעולה הכוללת ביצוע חוזר וזימונה בתוך לולאה בפעולה הראשית

בחינה עיונית:

- הבנת ומעקב אחר אלגוריתמים וקטעי קוד המבוססים על ביצוע חוזר.
- ניתוח בעיה וזיהוי סוג הביצוע החוזר הנדרש – אורך ידוע מראש או מותנה.
- חישוב מספר הצעדים של לולאה נתונה, כולל לולאות מקוננות.
- ניסוח אלגוריתם לפתרון בעיה הדורשת ביצוע חוזר.

<u>חלוקת שעות</u>

נושא	התנסות	עיוניות	סה"כ שעות
ביצוע חוזר באורך קבוע (for)			10
ביצוע חוזר מותנה (while)			15
קינון ושילוב מבני בקרה			10
תיעוד			3
יעילות			2
סה"כ שעות:			40

פרק 5: מבני נתונים סדרתיים

<u>יעדים:</u> הכרת מערכים כאוסף לינארי של טיפוסים מאותו סוג, עבודה עם מערכים חד-ממדיים ודו-ממדיים. הכרת ומימוש אלגוריתמי חיפוש, מיון ומיזוג.

<u>תכנים</u>

- מערכים של טיפוסי נתונים בסיסיים
- מושגי יסוד בעבודה עם מערכים: מציין (אינדקס), אורך (length), גישה (x[i])
- הגדרת ואתחול מערכים
- אלגוריתמים טיפוסיים:
  - חיפוש סדרתי
  - חיפוש בינארי
  - מיון הכנסה
  - מיזוג
  - דיון והשוואת יעילות האלגוריתמים
- מערך של עצמים:
  - בניית מערך עצמים (השימוש בפעולה new)
- גישה לתכונה של עצם במערך דו-מימדי

<u>מטרות ביצועיות</u>

- התלמיד ידע להסביר את הצורך במבנה נתונים סודר
- התלמיד יכתוב פעולות (methods) שמבצעות חישובים שונים על מערך נתון: סכום איברים, מספר איברים שמקיימים תנאי כלשהו, ערכי קיצון, ממוצע, בדיקת קיום תכונה כלשהי, וכדומה.
- התלמיד יסביר מה מתרחש מאחורי הקלעים כשהפקודה int[] x = new int[n] מתבצעת.
- התלמיד ידע לזהות חריגה מגבולות המערך ולכתוב קטעי קוד הנמנעים מחריגה כזו.
- התלמיד יכתוב פעולה שמקבלת מערך וערך כלשהו כפרמטרים ומאחזרת את מיקום הערך במערך בחיפוש סדרתי.
- התלמיד יסביר את אלגוריתם החיפוש הבינארי במערך ממוין.
- התלמיד ינתח באופן לא פורמלי את יעילות החיפוש הלינארי ואת יעילות החיפוש הבינארי במערך ממוין. (אין הכוונה לסדרי גודל, אלא לחוש את ההבדל בין חיפוש לינארי לחיפוש בינארי במערך בגודל 1000 תאים, למשל)
- התלמיד יכתוב אלגוריתם ויסביר במילים שלו כיצד לבצע מיון מערך חד ממדי בשיטת מיון הכנסה
- התלמיד יכתוב פעולה שממזגת שני מערכים ממוינים.
- התלמיד יקבל קטע קוד שכולל הגדרה, איתחול, ועיבוד פשוט של מערך של עצמים. התלמיד יסביר את דרך הפעולה של הקוד.
- התלמיד יקבל משימה מילולית שדורשת הגדרה, איתחול, ועיבוד פשוט של מערך של עצמים. התלמיד יממש את המשימה.
- התלמיד יגדיר מערך דו-ממדי ריבועי או מלבני

- התלמיד יבצע פעולות גישה לתא במערך (עצם או תכונה פשוטה) דו-ממדי, תוך שימוש מורכב במציינים. פעולות הגישה יכללו:
  - פעולות על האלכסון הראשי והמשני
  - סריקת המערך לפי שורות, לפי עמודות, 'בשבלול'
  - הפיכת מערך
- התלמיד יקבל קטע קוד שכולל הגדרה, איתחול, ועיבוד של מערך דו-מימדי. התלמיד יסביר את דרך הפעולה של הקוד.
- התלמיד יקבל משימה מילולית שדורשת הגדרה, איתחול, ועיבוד פשוט של מערך דו-מימדי. התלמיד יממש את המשימה. למשל: מציאת הערך המקסימלי במערך, חישוב ממוצע האיברים בכל שורה במערך, וכדומה.
- התלמיד ידע לעקוב בעזרת טבלת מעקב, אחר פעולות הכוללות מערכים שאבריהם משתנים במהלך הפעולה.

<u>דרכי הוראה</u>

רצוי לדון במערכים בשלב מוקדם יחסית בתכנית. דיון בסיסי במערכים מאפשר הצגת אלגוריתמים לביצוע חוזר שלא מחייבים הבנה טכנית עמוקה. ניתן בהחלט ללמד מושגי יסוד וכללי עבודה עם מערכים בתחילת הפרק הקודם, שנקרא "ביצוע חוזר", ולדחות את הוראת שאר הנושאים בפרק הנוכחי לשלב מאוחר יותר בתכנית, לפי שיקולי הוראה של המורה.

מבוא: יש לפתוח בהצגת בעיות שטבעי לפותרן בעזרת מבני נתונים ואלגוריתמים סדרתיים. למשל, נתון אוסף ערכים מסוים: ציוני מבחן, מחירי מוצרים, תוצאות ניסוי, וכדומה. אנו נדרשים למנות, לצבור, ולחשב תכונות סטטיסטיות שונות (ערכי קיצון, ממוצע) של האוסף. כעת ניתן להסביר שמבנה הנתונים "מערך" מספק מסגרת נוחה לייצוג וטיפול בבעיות כאלה. יש לשלב דוגמאות של מערכים של עצמים. למשל מערך ציון כאשר ציון הינו עצם.

מושגי יסוד רלבנטיים: יש לדון בטיפוס הנתונים של המערך, אורך המערך, המציין של האיבר הראשון (אפס) והאחרון (אורך המערך פחות אחת). מוסכמות אלו עשויות לבלבל בהכרות ראשונה ולכן חשוב לחדד אותן בכתה. יש להסביר שאם למשל grades הוא שם של מערך מסוג int, אזי grades[3] (או הפנייה לכל איבר אחר במערך) הוא משתנה לכל דבר ועניין שאפשר להשים לתוכו ערך int כלשהו או להשתמש בערכו הנוכחי בכל ביטוי חשבוני או פרמטר שמצפים לקבל ערכי int.

הגדרת ואתחול מערך: יש לדון באופי הדו-שלבי של הגדרת מערך ואיתחולו. זוהי אבחנה עדינה שניתן לדחות את הדיון בה, אך חייבים להבין אותה לפני הדיון במערכים של עצמים.

נתבונן בפקודה טיפוסית להגדרת מערך, למשל, int[] x. פקודה זאת יוצרת משתנה בשם x שמכיל את הערך null. בשלב הבא, אם וכאשר מאתחלים את המערך באמצעות פקודה כמו x = new int[100], נוצר עצם שמייצג את המערך. העצם הזה כולל, בין היתר, בלוק של 100 תאים מסוג int; מכאן ואילך, המשתנה x מצביע אל העצם הזה. בפרט, x[0] מצביע על התא הראשון במערך, x[1] מצביע על התא השני במערך, וכן הלאה, וניתן להשתמש בהם כמו משתנים רגילים לכל דבר ועניין. מומלץ להמחיש את ההסבר הזה בעזרת תרשים זיכרון. הבנה עמוקה של מודל הזיכרון של המערך והגישה לאיבריו סוללת את הדרך

להבנה, בשלב מאוחר יותר, איך ממומש מערך של עצמים. כמו כן, היא מאפשרת להבין שברמת ניהול הזיכרון, מערך הוא למעשה עצם.

כעת ניתן לדון בפקודה int[] x = new int[100], שמשלבת הגדרה ואתחול בפקודה אחת. בשלב זה התלמיד בשל להבין מה מתרחש מאחורי הקלעים כשהפקודה הזאת מתבצעת. כאן זה המקום להעמיק במילה new. ההוראה הזו מקצה את המקום בזיכרון כאשר x מכיל את המקום שהוקצה.

משימות טיפוסיות: יש לדון הן בבעיות שפתרונן מחייב סריקה מלאה של המערך, כמו למשל צבירה, מנייה, ממוצע, ערכי קיצון, מספר ערכים גדול/קטן מערך נתון, וכן הלאה, והן בבעיות שפתרונן אינו מחייב סריקה מלאה, כמו למשל בדיקת קיום או תכונה כלשהי.

חיפוש: הדיון במשימות חיפוש יתחיל באלגוריתם חיפוש סדרתי במערך לא ממוין בן n איברים. יש לדון ביעילות האלגוריתם (להתייחס למערך בגודל נתון כלשהו, למשל 1000 תאים). בשלב הבא יש לדון בחיפוש במערך ממוין, ולסייע לתלמידים לגלות בכוחות עצמם את אלגוריתם החיפוש הבינארי. יש לדון ביעילות האלגוריתם (10 צעדים יספיקו). ניתוח היעילות יעשה ביחידה הרביעית. כאן יש לאפשר לתלמיד לחוש בהבדל הניכר בין חיפוש בין 1000 ערכים לחיפוש ב- 10 ערכים בלבד.

לדיון בחיפוש בינארי יש חשיבות בתמונה הכללית של תכנית הלימודים: זוהי ההזדמנות היחידה של תלמיד 3 יח"ל במדעי המחשב להיחשף לאלגוריתם חשוב ומתוחכם ולערוך דיון משמעותי בנושא יעילות חישובית.

מיון: כיוון שחיפוש בינארי ובהמשך מיזוג מתייחסים למערכים ממויינים, יש להסביר כי ניתן למיין מערכים. אם זאת במסגרת יחידות לימוד אלו נתייחס למיון הכנסה בלבד. זהו מיון פשוט, שקל להגדיר את דרך פעולתו וכן לדון ביעילות החישובית שלו. תלמידים שימשיכו לימודיהם לרמה של 5 יח"ל יחפשו בהמשך גם למיון מיזוג.

מיזוג: כשדנים במשימת מיזוג מערכים, רצוי לדון הן במצבים בהם יש משמעות לכפילויות של ערכים, והן במצבים בהם אין לאפשר כפילות של ערכים. דוגמא טובה היא ביצוע סנכרון בין שתי רשימות אנשי קשר שנמצאות על טלפון נייד ועל מחשב אישי.

מבנה הבקרה foreach מספק תחביר אלגנטי לעבודה עם אוספים בכלל ומערכים בפרט. לכן, בכל מצב שניתן להשתמש בו במקום בלולאת for רגילה יש לעשות זאת. יש כמובן מצבים בהם לא ניתן להשתמש ב-foreach, כמו למשל כשלמציין (אינדקס) המערך יש תפקיד מפורש בגוף הלולאה.

מערך עצמים: יש להדגים משימות שכרוכות בעיבוד אוסף עצמים שהתלמיד מכיר מפרקים קודמים בתכנית. למשל, הזזה "במקביל" על המסך של מספר צבים שמיוצגים במערך עצמים מסוג Turtle, מציאת התלמיד בעל הציון הגבוה ביותר במערך עצמים מסוג Student, וכן הלאה. כללית, יש לחזור על חלק מאלגוריתמי העיבוד והחיפוש בהם דנו קודם, ולהפעילם על מערכים של עצמים.

חשוב להסביר את ההבדל בין שלב הגדרת מערך העצמים לבין שלב אתחול המערך (כפי שהוסבר לעיל), ולהדגימו באמצעות תרשים זיכרון.

מערך דו-מימדי: יש לדון במערכים מלבניים בלבד, דהיינו, מערכים בהם מספר הטורים בכל שורה זהה. יש לתאר משימות חישוב טיפוסיות על מערכים דו-ממדיים (אלכסונים, צבירה, הפיכה, סיבוב). יש

להסביר ולתרגל את שלבי ההגדרה והאתחול של מערך דו-מימדי, ולהסביר ולתרגל את העבודה עם שני מציינים במקביל. יש להסביר את האבחנה שמערך דו-מימדי ממומש למעשה ע״י מערך של מערכים.

<u>דרכי הערכה</u>

בכל הנאמר להלן, המלה ״מערך״ כוונתה מערך חד מימדי, מערך דו-מימדי, ומערך של עצמים.
בחינה במעבדה:

- כתיבת קטע קוד (בתוך תכנית נתונה) שעוסק בעיבוד מערך.
- כתיבת קטע קוד (בתוך תכנית נתונה) שמממש אלגוריתם חיפוש או מיזוג.

בחינה עיונית:

- מעקב אחר קטע קוד שעוסק בעיבוד מערך.
- הסבר מודל הזיכרון של מערך.
- כתיבת קטע קוד שעוסק בעיבוד מערך
- כתיבת פעולה על איברי מערך ותכנית המזמנת פעולה זו עבור כל איברי המערך

<u>חלוקת שעות</u>

נושא	סה״כ שעות
מבוא למערכים	2
פעולות סדרתיות (מקבלות מערך) כולל חיפוש סדרתי	8
פעולות שמחזירות מערך	4
חיפוש בינארי	4
מיון הכנסה	4
מיזוג	4
מערך עצמים	4
מערכים דו-ממדיים	16
סה״כ שעות:	46

פרק 6: תכנות מונחה עצמים

<u>יעדים</u>: מבוא לתכנות מונחה עצמים וכתיבת מחלקות.

<u>תכנים</u>

- חזרה: יצירת והפעלת עצמים תוך שימוש בממשק מחלקה נתון.
- מושגי יסוד: מחלקה כתבנית ליצירת והפעלת עצמים, שימוש לעומת מימוש.
- מושגי יסוד: תכונות, בנאים, פעולות (שאילתות / פקודות).
- מצב העצם / מצב מצב תקין.
- הכמסה: חשיבות העיקרון ודרך מימושו באמצעות הרשאות גישה וכתיבת פעולות מתאימות.
- עצמים שאינם ניתנים לשינוי (immutable objects).
- תנאי קדם / תנאי בתר.
- מחלקות המשתמשות במחלקות אחרות.
- מימוש מחלקה בשפת תכנות לפי ממשק נתון.
- תיעוד ממשקים.
- כתיבת תכניות בדיקה.

<u>מטרות ביצועיות</u>

- התלמיד יקבל ממשק מחלקה, מימוש של המחלקה, ומשימה מילולית שדורשת יצירה והפעלה של מספר עצמים. התלמיד יממש את המשימה ע"י כתיבת קוד שמשתמש בשירותי המחלקה.
- התלמיד יסביר במילים שלו מה מתרחש בצד המחלקה כאשר קטע הקוד שהוא כתב לעיל מתבצע.
- התלמיד יסביר את עקרון ההכמסה ויתאר כיצד הוא בא לידי מימוש בקוד מחלקה נתון.
- התלמיד יסביר האם וכיצד קוד מחלקה נתון מפר את עקרון ההכמסה.
- בהינתן תיאור מילולי וממשקי מחלקה מתאימים, התלמיד יכתוב קוד שיוצר ומפעיל עצמים שטיפוס אחת מתכונותיהם הוא מחלקה.
- התלמיד יתאר מהו מצב "מצב תקין" של עצם וכיצד הוא בא לידי ביטוי בהינתן תיאור מילולי של בעיה.
- התלמיד יקבל ממשק מחלקה כולל תכונותיה ומשימה מילולית. התלמיד יתרגם את המשימה המילולית לפעולה במחלקה שעושה שימוש בתכונותיה.
- התלמיד יקבל מימוש של המחלקה, ותוכנית היוצרת עצמים מהמחלקה הנתונה ומשתמשת בהם. התלמיד ידע לבצע מעקב על תכנית זו ולציין את הפלט המתאים.
- התלמיד יקבל מימוש של המחלקה, ותוכנית היוצרת עצמים מהמחלקה הנתונה ומשתמשת בהם. התלמיד ידע לבצע שינויים בתוכנית לצורך קבלת פלט נדרש.
- התלמיד יתאר מהם תנאי קדם (הדרישות וההנחות על מצב העצם, כדי שניתן יהיה לזמן פעולה על העצם) ותנאי בתר (השפעת הפעולה על העצם, כולל מצב העצם, ערך מוחזר, הדפסות וכו') וכיצד הם באים לידי ביטוי בהינתן תיאור מילולי של בעיה.
- בהינתן ממשק מחלקה, התלמיד יתכנן ויכתוב תכניות בדיקה למחלקה.
- התלמיד יתעד תכניות.

- התלמיד יסביר באופן בסיסי מהי ירושה, ויסביר API הכולל ירושה פשוטה.

<u>דרכי הוראה</u>

בפרקים 1-5 התלמידים קראו וכתבו תכניות מבוססות-עצמים רבות שפעלו מול מגוון מחלקות מוכנות. סגנון העבודה היה תכנות מבוסס עצמים מול ממשקי מחלקה נתונים. בפרק זה אנו פותחים את הקופסא השחורה ועוסקים בתכנות מונחה עצמים. המתודולוגיה מבוססת על קריאה וכתיבה של חלק מהמחלקות שהופיעו בפרקים הקודמים, ומחלקות נוספות.

חשוב לציין שמבחינה מקצועית ופדגוגית קיימת קפיצת דרך משמעותית בין מתכנת שפועל מול ממשק מחלקה נתון לבין מתכנת שכותב מחלקות לטובת מתכנתים אחרים. תכנית הלימודים של יחידות 1-2 כוללת מבוא לתכנות מהסוג הראשון, וחשיפה מבוקרת ומצומצמת לתכנות מהסוג השני. אפיון, עיצוב, וכתיבת מחלקות הם נושאים מורכבים ומתוחכמים שחשיפה לא מבוקרת אליהם יוצרת יותר נזק מתועלת. אי לכך, מטרתנו היא להקנות מושגים כלליים בתכנות מונחה עצמים, אך לא לעסוק בסוגיות של עיצוב מחלקות.

יש לפתוח את הפרק הנוכחי בחזרה על משימות תכנות פשוטות מול ממשקי מחלקות קיימות. בפרט, יש לחזור על המושגים הבאים: קוד שמשתמש בשירותי המחלקה, קוד שמממש את שירותי המחלקה, ממשק מחלקה, יצירת עצמים דרך שימוש בפקודת new, וזימון פעולות על עצמים. כדי להבין שמחלקה היא תבנית ליצירת והפעלת עצמים, חיוני שדוגמאות הקוד תכלולנה יצירה והפעלה של מספר עצמים מאותה מחלקה. לדוגמא, יצירת מספר חשבונות בנק מטיפוס BankAccount וזימון פעולות בנקאיות טיפוסיות על החשבונות השונים. יש לציין שכל העצמים הללו (למשל, חשבונות הבנק השונים) הם מופעים של אותה תבנית. "מופע" (instance) הוא מושג חשוב שיש להסבירו ולהשתמש בו.

בשלב זה אנו מוכנים להתבונן במימוש חלק מהבנאים והפעולות של המחלקות מולן עבדנו. ככלל – אנו מדגישים זאת שוב – לפני שקוראים או כותבים קוד של מחלקה כלשהי יש לדון בממשק המחלקה ולהדגים קוד שמשתמש בה. הכלל הזה תקף גם בכתיבה של מחלקה חדשה: לפני שטורחים להשלים את כתיבת קוד המחלקה, יש לכתוב מחלקה אחרת שנועדה לבדוק את השימוש בה.

חשוב לזכור שהדגש בפרק הנוכחי הוא על הבנת מושג המחלקה ולא על אלגוריתמיקה. לכן, המחלקה האידיאלית לצרכינו היא מחלקה מוכרת לתלמיד שמאופיינת מחד במגוון מעניין של תכונות, בנאים ופעולות, ומאידך לפעולותיה מימוש קצר וקולע. לדוגמא, מחלקות כמו VideoClip, BankAccount, Fraction. יש להימנע מדיון במחלקות מוכרות שהממשק שלהן והשימוש בהן הוא פשוט, אך המימוש שלהן בהחלט אינו פשוט, כמו למשל מחלקת Turtle.

יש להתמקד במחלקות שאינן תלויות במחלקות אחרות. לדוגמא, מחלקת VideoClip טיפוסית תהיה מבוססת מן הסתם על תכונות כגון title, length, category וכדומה, שכולן טיפוסי נתונים בסיסיים, ותכונה ששמה, נניח, media שהיא מטיפוס מחלקת File. יש לדון רק בתכונות שהן טיפוסי נתונים בסיסיים.

תכונות / מצב העצם: הדיון במושג "עצם" מוביל באופן טבעי לדיון במושגים "תכונות העצם" ו- "מצב העצם". בתכנות מונחה עצמים, אוסף הערכים הנוכחיים של כל תכונות העצם נקרא "מצב העצם". יש לדון במשמעות המושג "מצב תקין" של עצם. למשל, סביר לדרוש שאורך סרטון בדקות (length) של עצם

מטיפוס VideoClip יהיה מספר לא שלילי; שיתרת החשבון (balance) של עצם מטיפוס BankAccount לא תחרוג ממגבלת האשראי (credit); שהמכנה (denominator) של עצם מטיפוס Fraction יהיה שונה מאפס, וכן הלאה. יש לדון בכך שבאחריות כותב המחלקה לתעד במפורש את "מצב תקין" של העצמים שנוצרים ממנה.

יש להדגים וללמד הגדרת תכונות במחלקה. יש להקפיד שהרשאת הגישה של כל התכונות המוגדרות תהיה private, אך לדחות את הדיון במשמעות הרשאות הגישה לשלב מאוחר יותר בפרק.

סיווג פעולות: יש לסווג (ברמת הדיון) את פעולות המחלקה לשלושה סוגים: בנאים (פעולות שיוצרות עצמים), שאילתות (פעולות שמאחזרות ערכים ואינן משנות את מצב העצם), ופקודות (פעולות מטיפוס void שמשנות את מצב העצם ואינן מאחזרות ערך). זהו סיווג שהתלמידים אמורים להכיר כי כל ממשקי המחלקות שהם ראו עד כה בפרקים 1-5 אמורים להשתמש במינוח הזה.

בנאי: יש להסביר את הצורך בפעולה שמטרתה בניית עצם חדש ואתחולו למצב חוקי/"מצב תקין", לפי כללי איפיון המחלקה. יש לדון בהגדרת והעברת פרמטרים, במושג "העצם הנוכחי", ובשימוש מושכל במילה השמורה this. יש להראות (במקביל) דוגמאות קוד ליצירת עצמים מצד התכנית שמזמנת את שירותי הבנאי, ודוגמאות קוד של הבנאים שבונים את העצמים הללו. כדי למנוע בלבול, יש להקפיד על שימוש עקבי במינוח הבא: קוד שמשתמש בפעולת new "יוצר עצם"; קוד הבנאי שמגיב לפקודת new "בונה את העצם". במילים אחרות, יצירת עצם היא פעולה מופשטת שהבנאי נותן לה משמעות בנייה ממשית.

"מצב תקין": יש לשוב ולדון במושג "מצב תקין" של עצם ולציין שבאחריות הבנאי (כלומר, באחריות המתכנת שכותב את קוד הבנאי) לאתחל את העצם למצב התחלתי "תקין". המשמעות היא שקוד הבנאי חייב לבדוק את ערכי הפרמטרים שמועברים אליו ולפעול בהתאם.

בנאי ברירת מחדל: יש לדון בכך שאם לא נכתוב בנאי באופן מפורש, מהדר השפה יוסיף לקוד המחלקה כברירת מחדל בנאי נטול פרמטרים. לכן, מחלקה יודעת להגיב לפקודות new גם אם המימוש שלה לא כולל בנאי מפורש. יחד עם זאת, יש לדון בכך שגם אם מעוניינים בבנאי "ריק" שלא מאתחל את העצם, עדיף להגדיר ולממש אותו באופן מפורש, כדי שממשק וקוד המחלקה יהיו בהירים וברורים. יתרה מכך, במידה והוגדר במחלקה בנאי כלשהו, המהדר לא יוסיף בעצמו בנאי ברירת מחדל נטול פרמטרים.

הכמסה: הכמסה (encapsulation) היא מטרה עיצובית מופשטת שהמימוש שלה מתבצע (בין היתר) ע"י תיוג תכונות העצם כ- private וכתיבת פעולות ציבוריות (public methods) שפועלות על התכונות הללו. בפרט, הפעולות שנועדו לאחזר את ערכי התכונות נקראות "שאילתות" (queries), והפעולות שנועדו לשנות את ערכי התכונות נקראות "פקודות" (commands). יש לדון במושגים הללו ולהדגים כיצד הם באים לידי ביטוי בקוד המחלקות בהן אנו משתמשים במהלך הפרק.

הדיון בהכמסה צריך להוביל לדיון נוסף במושג "מצב העצם". אם תכונות העצם מוכמסות, המתכנת שמשתמש בעצם אינו מודע לתכונות הללו. מבחינתו, מצב העצם הוא מושג מופשט שמתבטא בערכים שמאוחזרים ע"י שאילתות שמתועדות בממשק המחלקה. לעומת זאת, מנקודת ראותו של כותב קוד המחלקה, מצב העצם הוא מושג קונקרטי שבא לידי ביטוי בערכי התכונות הנוכחיים.

ממשק המחלקה: יש להסביר שממשק המחלקה מהווה חוזה בין המחלקה לבין לקוחות של המחלקה. בפרט, ממשק שנבנה על עיקרון ההכמסה מאפשר ללקוח להשתמש בשירותי המחלקה בלי להיות מודע לפרטי המימוש שלה. לצורך זה, הממשק כולל את כותרות הפעולות שהמחלקה מייצאת, דהיינו הפעולות הציבוריות שהרשאת הגישה שלהן היא public. "כותרת הפעולה" מורכבת משם הפעולה, הטיפוס המאוחזר ע"י הפעולה, ושמות וטיפוסי הפרמטרים של הפעולה.

בכדי להדגים את עקרון ההכמסה יש לכתוב מחלקה שהממשק שלה ניתן למימוש בדרכים שונות. לדוגמה, המחלקה **מלבן מקביל לצירים**. ממשק המחלקה יכלול, בין היתר, את הפעולות: פינה-ימנית-עליונה, פינה-ימנית-תחתונה, פינה-שמאלית-עליונה, פינה-שמאלית-תחתונה, נקודת-אמצע, שטח, היקף, רוחב, אורך. מימוש הפעולות הללו יתבצע פעם אחת בעזרת חישובים שמתבססים על פינות המלבן, ופעם אחרת בעזרת חישובים שמתבססים על נקודת האמצע, אורך ורוחב המלבן. התלמידים יתבקשו לכתוב תכנית המשתמשת במחלקה ולהריצה פעמיים, בכל פעם תוך שימוש במימוש שונה של המחלקה.

פעולות עזר: יש לציין שמימוש מחלקה יכול בהחלט לכלול גם פעולות פרטיות (private methods), אך אלה אינן כלולות בממשק המחלקה. יש להסביר את חשיבות תכנון ההפרדה בין ההפשטה (ממשק המחלקה) למימוש המחלקה. ההפרדה נועדה לחסוך מלקוחות המחלקה מידע מיותר, ולאפשר שינויים במימוש המחלקה בלי להשפיע על לקוחות שמשתמשים בה. אנלוגיה טובה לכך היא התפתחות תעשיית הרכב בשנים האחרונות. ממשק המכונית (הגה, ידית הילוכים, דוושת האצה, דוושת בלמים) נשאר קבוע לאורך השנים, בעוד כל המערכות מתחת מכסה המנוע משתנות באופן דרמטי. המודולריות הזאת מתאפשרת הודות להפרדה מושכלת בין עיצוב הממשק (אמצעי השליטה במכונית) למימושו (טכנולוגית המנוע, התמסורות, הבלמים, וכדומה). בהקשר הזה, "פעולה פרטית" יכולה להיות, למשל, חישוב כמות הדלק שיש להזריק למנוע במצבי האצה שונים. זוהי פעולה חשובה לפעולה תקינה של המכונית, אך לנהג לא צריכה להיות גישה אליה ולכן היא אינה כלולה בממשק המכונית.

תנאי קדם / תנאי בתר: יש לדון בשאלה איזה מידע צריך לכלול ממשק המחלקה. בנוסף לכותרות הפעולות הציבוריות, יש להתייחס גם לפרטים הבאים: מתי ואיך רשאים לזמן את הפעולה (תנאי קדם)? מה האפקט הכולל של הפעולה (תנאי בתר)? מהי יעילות הפעולה? תנאי הקדם והבתר יוסברו באופן אינטואיטיבי ויתוארו במונחים של שאילתות על המחלקה, ללא תלות במימוש המחלקה.

העמסת פעולות (method overloading): היא טכניקת תכנות נפוצה ולכן יש לדון בה. יחד עם זאת, אנו ממליצים להמעיט את השימוש בה, כדי למנוע בלבול. במקום להעמיס פעולות מומלץ לתת לכל פעולה שם ייחודי שמרמז על דרך פעולתה והפרמטרים שהיא מצפה לקבל. היוצא מן הכלל הוא העמסת בנאים במידה והיא נדרשת.

תכונות ופעולות סטטיות: באופן הטהור ביותר, תכנות מונחה עצמים מבוסס על ההנחה שכל דבר הוא עצם. לכן, פרק זה לא מתייחס לתכונות סטטיות ולפעולות סטטיות, שלפי הגדרה אינן קשורות לעצמים. יחד עם זאת, מכיוון שהתלמידים נחשפו לפעולות סטטיות בפרקים 1-5 (למשל, פעולת main, פעולות עזר, ופעולות ממחלקת Math), גם בפרק זה יכולה בהחלט להתעורר השאלה "ומה עם פעולות סטטיות?" אנו מציעים שלוש תשובות אפשריות לשאלה הזאת: (1) פעולות סטטיות מתאימות למחלקות שמשמשות כ-"ספריות פעולות", כגון מחלקת Math; מכיוון שהמחלקות בהן אנו עוסקים בפרק זה הן מסוג אחר לגמרי – מחלקות שמייצגות עצמים – אין צורך לדון בפעולות סטטיות. (2) לפי כלל עיצובי חשוב, אם במחלקה כלשהי יש פעולה סטטית, אזי כל פעולות המחלקה צריכות להיות סטטיות; שוב, זהו לא סוג המחלקות בהן אנו עוסקים כאן. (3) במקרים רבים, הצורך בהגדרת תכונה סטטית או פעולה סטטית מעיד על כך

שעיצוב המחלקה לוקה בחסר, ואולי, למשל, יש לפצלו לשתי מחלקות. זהו נושא עיצובי שרצוי לדון בו, כי הוא חורג מתכנית הלימודים.

סגנון כתיבה, תיעוד, בדיקה: יש להרגיל את התלמידים לכתוב ולתעד מחלקות בסגנון תכנות אחיד וקריא לפי סטנדרטים מקובלים בתעשייה לכתיבת קוד מונחה-עצמים. יש לשלב הוראה ותרגול של כתיבת תוכניות בדיקה ידניות ואוטומטיות.

ירושה: תלמידים שיעיינו בתיעוד ה API's של מחלקות שונות ייתקלו במונחים כגון "extends Object". יש לחשוף את התלמידים למושג הירושה באופן בסיסי ומושגי בלבד. כלומר יש להתמקד בהבהרת טכניקת הירושה בשורת הכותרת של המחלקה היורשת ובאפשרות להשתמש בפעולות שהוגדרו במחלקת האב. עיקר ההסבר על ירושה נועד לתאר ולהבין את תיעוד ה- API's.

<u>חלוקת שעות</u>

נושא	סה״כ שעות
חזרה: תכנות מול ממשקי מחלקות מוכנות	4
כתיבת בנאים	4
הגדרת תכונות, כתיבת פעולות (שאילתות), הכמסה	8
כתיבת פעולות (פקודות)	10
מצב העצם ״התקין״: הגדרה ומימוש	6
מימוש מחלקות המשתמשות במחלקות אחרות	8
ירושה	4
סה״כ שעות:	44

נושאים במדעי המחשב

(יחידת לימוד 3)

התלמיד מגיע ליחידת הלימוד השלישית לאחר שסיים את פרק היסודות (שתי יחידות הליבה הראשונות) בתכנית הלימודים. בשלב זה, התלמיד ניחן בכישורי תיכנות ואלגוריתמיקה בסיסיים. מטרת יחידת הלימודים השלישית היא להעמיק את הידע הקיים של התלמיד ולהביאו לידי ביטוי בפיתוח פרויקט מאתגר. מטרה נוספת של יחידת הלימוד השלישית היא להציג לתלמיד תחום ידע ספציפי במדעי המחשב.

כללית, שעות ההוראה ביחידה מתפלגות כדלקמן: הוראת תחום הידע הנוסף (60 שעות), פיתוח פרויקט (30 שעות).

בתי ספר שמלמדים מדעי המחשב יכולים לבחור ללמד את אחת החלופות הבאות:

- בסיסי נתונים ומערכות מידע
- גרפיקה ממוחשבת
- ארגון המחשב ושפת סף
- תיכנות פונקציונאלי
- תיכנות בסביבת האינטרנט
- תכנות לוגי (פרולוג)

יחידות לימוד 5-4

מבני נתונים

(יחידת לימוד 4)

יחידת לימוד זאת מחליפה את יחידת לימוד 4 הנוכחית, "עיצוב תוכנה".

היחידה עוסקת בעיקר בבניית ועיבוד מבני נתונים (מחסנית, תור, עץ בינארי), בשימוש מושכל בהם, ובמימושם בעזרת מערכים ורשימות מקושרות. כמו כן נלמד להשתמש במבנים אלו כדי לייצג ולממש טיפוסי נתונים מופשטים ומורכבים.

בנוסף, היחידה כוללת שני פרקי מבוא בעלי חשיבות כללית במדעי המחשב – רקורסיה ויעילות. החומר שנלמד בשני הפרקים הללו בא לידי ביטוי אינטנסיבי בניתוח ובמימוש מבני הנתונים השונים שמוצגים בהמשך היחידה.

נושאי הלימוד מחולקים לשבעה פרקי חובה, כדלקמן:

	פרק מספר	שם הפרק	שעות לימוד
יחידת לימוד 4: מבני נתונים	1	רקורסיה	14
	2	מבוא ליעילות	8
	3	מחסנית	12
	4	תור	8
	5	רשימה מקושרת	15
	6	מימוש מבני נתונים	15
	7	עצים בינאריים	18
		סה"כ	**90**

כשליש מהשעות מוקדש לתרגול במעבדה. חלוקת השעות המדויקת בין שעות המעבדה לשיעורים התיאורטיים נתונה לשיקול דעתו המקומי של המורה.

פרק 1: רקורסיה

<u>יעדים</u>: הכרת מושג הרקורסיה כשיטה לפתרון בעיות; הכרת היתרונות והחסרונות של פתרונות רקורסיביים; ייצוג בעיות ופתרונן בעזרת רקורסיה.

<u>תכנים</u>

1. הגדרה רקורסיבית, תנאי עצירה, קריאה רקורסיבית
2. רקורסיית זנב, רקורסיית הלוך-חזור, רקורסיה כפולה, רקורסיה הדדית.
3. מעקב רקורסיבי ברמה של f(n) ו- f(n-1)
4. כתיבת תכניות רקורסיביות.

<u>מטרות ביצועיות</u>

1. התלמיד יעקוב אחר תהליך רקורסיבי וירשום את סדר המעקב במונחים של f(n) ו- f(n-1)
2. התלמיד יבחין בין סוגי רקורסיות: זנב, הדדית, כפולה, הלוך-חזור.
3. התלמיד יכתוב תכניות רקורסיביות פשוטות שפועלות על מספרים, סדרות, מערכים חד- ודו-ממדיים.
4. התלמיד יסביר את יתרונות הרקורסיה: פשטות הפתרון, הוכחת נכונות.
5. התלמיד יסביר את חסרונות הרקורסיה: צריכת משאבי מקום וזמן.
6. התלמיד יסביר מדוע אין בעיה הניתנת רק לפתרון רקורסיבי.
7. התלמיד יפגין הבנה של מבנים רקורסיביים, ויפתור בעיות על מבנים רקורסיבים.

<u>דרכי הוראה</u>

1. יש להציג פעולות רקורסיביות פשוטות ולהבין את דרך ביצוען. לדוגמה: עצרת.
2. יש להציג פעולות רקורסיביות מורכבות (כולל רקורסיה כפולה ורקורסיה הדדית) ולהבין את דרך ביצוען.
3. יש לדון בפעולות רקורסיביות שמחזירות ערך ובפעולות רקורסיביות שלא מחזירות ערך.
4. יש ללמד ולממש פתרונות רקורסיביים לבעיות שונות וברמות קושי שונות: פעולות על מספרים שלמים, כגון חישוב עצרת, כפל מספרים שלמים, סכום ספרות במספר שלם, היפוך ספרות מספר שלם, סכום ספרות במספר המקיימות תנאי מסוים (לדוגמה, ספרות זוגיות), סדרת פיבונאצ'י, וכדומה.
5. יש ללמד ולממש פעולות רקורסיביות על מערכים חד-ממדיים ודו-ממדיים: חישוב סכום ערכים במערך, חישוב מקסימום / מינימום במערך, חישוב סכום מספרים במערך המקיימים תנאי כלשהו (לדוגמה, מספרים זוגיים), חישוב סכום איברים במערך הנמצאים במקומות המקיימים תנאי כלשהו במערך, חיפוש בינארי במערך חד-ממדי.
6. יש להראות את המעבר בין פתרון איטרטיבי לפתרון רקורסיבי ברקורסיות זנב ולהדגיש את מקומו של משתנה הלולאה המועבר ברקורסיה כפרמטר.
7. יש להדגיש את חשיבות תנאי העצירה.
8. יש להדגיש את תפקיד הפרמטרים השונים בפעולות הרקורסיביות – פרמטרים המשתנים מקריאה לקריאה ופרמטרים שאינם משתנים בין הקריאות.
9. יש להסביר ולממש פעולות רקורסיביות המתבצעת על מחסניות ו/או תורים (לאחר שמחסניות ותורים נלמדו בפרקים הבאים).

10. יש ללמד לעקוב אחר פעולה רקורסיבית המתבצעת על מבנים רקורסיביים.

<u>דרכי הערכה</u>

בחינה עיונית:

1. מעקב אחר רקורסיה
2. ניסוח אלגוריתם רקורסיבי המתואר באופן מילולי.
3. פתרון בעיה רקורסיבית פשוטה.

בחינה במעבדה:

1. התלמיד יקבל תכנית רקורסיבית הכוללת שגיאה (לדוגמא, ללא תנאי עצירה) ויתבקש לתקנה.
2. התלמיד יכתוב תוכנית שממשת אלגוריתם רקורסיבי המתואר באופן מילולי.

<u>חלוקת שעות</u>

נושא	שעות
הסבר ומעקב אחרי תהליך רקורסיבי	2
סוגי רקורסיה	2
דיון בחשיבות תנאי העצירה של הרקורסיה	1
דיון בחשיבות הגדרה נכונה של הפרמטרים ברקורסיה	1
פתרון בעיות וכתיבת תכניות רקורסיביות	8
סה"כ שעות:	14

כשליש מהשעות מוקדש לתרגול במעבדה. חלוקת השעות המדויקת בין שעות המעבדה לשיעורים התיאורטיים נתונה לשיקול דעתו המקומי של המורה.

פרק 2: מבוא ליעילות

<u>יעדים:</u> לפרק זה שני יעדים: (א) חזרה וריענון של מערכים ואלגוריתמי חיפוש ומיון -- חומר שנלמד בפרקי היסודות (יי"ל 2-1); (ב) הכרת מושג היעילות וחשיבותו במדעי המחשב, וזאת דרך הכרת מדדי זמן-ריצה. בפרט, נכיר את מושגי היעילות הבסיסיים "המקרה הגרוע ביותר", "המקרה הטוב ביותר", ו- "המקרה הממוצע". כמו כן נכיר את מושג אורך הקלט ואת מושג היעילות כפונקציה של אורך הקלט, ואת המושג "סדר גודל" (O גדול). המושגים הללו ילמדו באופן אינטואיטיבי, בלי להיכנס להגדרות מתמטיות. כדי להדגים את החומר הנלמד, ננתח את יעילותם של אלגוריתמים שונים שנלמדו כבר בפרקי היסודות, כגון מציאת מקסימום, חיפוש סדרתי, חיפוש בינארי, מיון הכנסה.

<u>מטרות ביצועיות</u>

1. התלמיד יפגין הבנה של תלות זמן הריצה של אלגוריתם נתון כתלות של אורך הקלט.
2. בהינתן אלגוריתם כלשהו וקלטים שונים, התלמיד יזהה את המקרה הגרוע ביותר והמקרה הטוב ביותר במונחי זמן-ריצה.
3. התלמיד ינתח את זמן הריצה של אלגוריתם מילולי נתון כתלות בפעולות בסיסיות ובאורך הקלט.
4. התלמיד ינתח את זמני הריצה של אלגוריתמים שונים שנלמדים בתכנית הלימודים, הן בפרקי היסודות והן ביחידת הלימוד הנוכחית.

<u>דרכי הוראה</u>

1. יש לפתוח בחזרה על מערכים, ולתאר אלגוריתם פשוט על מערך, כמו למשל חיפוש סדרתי.
2. יש להשתמש בדוגמא לעיל כדי להסביר את המושגים הבאים: "אורך קלט", "צעד בסיסי", "פעולות התלויות באורך הקלט", "פעולות שאינן תלויות באורך הקלט".
3. יש לחזור על האלגוריתמים השונים שנלמדו בפרקי היסודות: חיפוש סדרתי, חיפוש בינארי, מיון הכנסה.
4. יש ללמד מיון מיזוג ולנתח את סיבוכיותו
5. יש לנתח את יעילותו של כל אלגוריתם במונחי זמן-ריצה. רצוי להשתמש לצורך זה בפסאודו-קוד, כדי לחדד את ההבדל המהותי בין פעולות התלויות באורך הקלט לבין פעולות בזמן קבוע.
6. יש להסביר את המושגים הבאים: סדר גודל, סדר גודל לינארי, לוגריתמי, ריבועי ומעריכי.
7. יש להסביר את המושגים הבאים: מקרה גרוע, מקרה טוב, מקרה ממוצע, שיפור בסדר גודל, שיפור בקבוע.
8. יש להציג את כל המושגים לעיל באופן אינטואיטיבי, ללא שימוש במתמטיקה. למשל, ניתן לאמוד זמן-ריצה באמצעות ספירת איטרציות. יחד עם זאת, חובה לנסח כל מדד שהוא (כולל מספר איטרציות) כפונקציה של גודל הקלט עליו פועלים.
9. הערה: בפרקים הבאים ביחידת הלימוד הנוכחית נממש אלגוריתמים רבים ונדון ביעילותם.

<u>דרכי הערכה</u>

בחינה עיונית:

- התלמיד יחשב את סיבוכיות הזמן של אלגוריתם נתון או של תכנית נתונה.
- בהינתן אלגוריתם או תכנית כלשהי, התלמיד יציע דרכים לשיפור זמן הריצה, ללא שינוי של סיבוכיות הזמן.

<u>חלוקת שעות</u>

נושא	שעות
חזרה על מערכים ואלגוריתמים מפרקי היסודות (י״ל 2-1)	2
הגדרת והדגמת מושגים בסיסיים בקונטקסט של האלגוריתמים לעיל: אורך קלט, זמן ריצה, המקרה הטוב ביותר, הגרוע ביותר, הממוצע	3
יעילות כפונקציה של אורך הקלט והגדרת המושג ״סדר גודל״	1
ניתוח יעילות של אלגוריתמים	2
סה״כ שעות:	8

כשליש מהשעות מוקדש לתרגול במעבדה. חלוקת השעות המדויקת בין שעות המעבדה לשיעורים התיאורטיים נתונה לשיקול דעתו המקומי של המורה.

פרק 3: מחסנית

<u>יעדים:</u> הכרת המושג "טיפוס נתונים מופשט"; הכרת טיפוס הנתונים המופשט "מחסנית" (stack); הכרת שימושים נפוצים במחסנית; תרגול עבודה מול ממשק מחלקה (API); הכרת מושג הגנריות והשימוש בו.

<u>תכנים</u>

- טיפוס נתונים מופשט
- מחסנית, LIFO
- ממשק מחלקה: בנאים, פעולות שמחזירות ערך, פעולות עדכון
- גנריות

<u>מטרות ביצועיות</u>

1. התלמיד יסביר מהו טיפוס נתונים מופשט.
2. התלמיד יסביר את טיפוס הנתונים המופשט מחסנית (Stack) ושימושיו.
3. התלמיד יבנה ויעבד מחסנית כדי לפתור בעיות המאופיינות ע"י תבנית LIFO
4. התלמיד יסביר את הממשק של מחלקת Stack
5. התלמיד יפגין יכולת לכתוב קוד שמשתמש כהלכה בשירותי ממשק מחלקת Stack.
6. התלמיד יסביר את מושג הגנריות.
7. התלמיד יפגין יכולת לכתוב קוד שמשתמש כהלכה בשירותי ממשק מחלקת Stack<T>
8. התלמיד יסביר כיצד ניתן לפעול על טיפוס נתונים מופשט בלי לדעת כיצד הוא ממומש.

<u>דרכי הוראה</u>

1. יש להציג את המושג "מחסנית" כטיפוס נתונים מופשט (ADT = Abstract Data Type) שניתן לבצע עליו את הפעולות שמתועדות בממשק מחלקת Stack: בניית מחסנית ריקה, בדיקה האם המחסנית ריקה, דחיפת ערך למחסנית, שליפת ערך מהמחסנית, והחזרת הערך העליון במחסנית ללא הוצאתו מהמחסנית.
2. חשוב להתייחס למחסנית כאל הפשטה שאיננו מכירים את אופן מימושה ואיננו מתעניינים בו. בפרט, יש להתייחס אל המחסנית כאל עצם מטיפוס Stack שניתן לבצע עליו את הפעולות שמתועדות בממשק המחלקה, ותו לא.
3. יש לדון באופן כללי בשימושים אופייניים במחסנית. למשל, שירות undo במעבד תמלילים, בדיקת איזון סוגריים בביטוי חשבוני, וכדומה. דוגמאות אלה מחדדות לא רק את חשיבות המחסנית, אלא גם את הצורך בייצוג גנרי של מחסנית, כלומר, את הצורך במחסנית שיכולה לייצג פריטים מטיפוסי נתונים שונים, לפי אופי היישום הנתון.
4. התלמיד יעבוד מול ממשק מחלקת המחסנית (Stack API) המתועד בנספח בסוף הפרק הנוכחי. ממשק זה כולל רק את הפעולות הנחוצות להוראת הפרק, ומהווה תת-קבוצה של ממשק המחסנית הסטנדרטי.
5. יש לקיים דיון במושגי המפתח "ממשק" ו- "הכמסה", ולהדגיש שניתן לשנות את מצב המחסנית (דהיינו, לשנות עצם מטיפוס Stack) רק באמצעות פעולות המבוצעות דרך ממשק המחסנית.

6. יש לדון בהבדל העיצובי בין פעולות שנועדו לאחזר ערכים בלי לשנות את מצב העצם, להן ניתן לקרוא "שאילתות", לבין פעולות שמטרתן לשנות את מצב העצם ללא החזרת ערך, להן ניתן לקרוא "פקודות". כעקרון, כאשר מעצבים טיפוס נתונים מופשט, רצוי להקפיד על ההפרדה הזאת. אך לעיתים, כאשר היתרונות לפעולה משולבת מצדיקים זאת, אפשר לחרוג מהעיקרון ולעצב פעולה שגם משנה את מצב העצם וגם מחזירה ערך. דוגמא: פעולת pop.
7. יש להדגיש שניתן להשתמש במחסנית באופן מלא בלי לדעת דבר על האופן שבו היא ממומשת. יש לדון ביתרונות ההפרדה בין שימוש למימוש, שהיא מושג מרכזי בתכנות מונחה-עצמים.
8. יש להציג את מושג הגנריות (genericity) בהדרגה. מומלץ להתחיל ללמד את כל האמור לעיל בעזרת מחסנית שפועלת על ערכי int בלבד (ראה ממשק מחלקת StackInt שמופיע בנספח לפרק זה). לאחר מכן יש להדגים בעיה כלשהי שמצריכה פתרון המשלב עבודה עם מחסנית שפועלת, נניח, על ערכי String. דיון זה יוביל באופן טבעי לצורך בטכניקת הגנריות. בשלב זה יש להציג את ממשק המחסנית הגנרי <Stack<T וללמד כיצד לעבוד איתו.

<u>דרכי הערכה</u>

בחינה עיונית:

1. התלמיד יפגין יכולת שימוש בממשק מחסנית לצורך פתרון בעיות מתאימות. לדוגמה: תקינות סוגריים, חיבור מספרים בעלי 100 ספרות ויותר, וכדומה.
2. התלמיד יממש פעולות נוספות על מחסנית שאינן מופיעות בממשק Stack.

בחינה במעבדה:

1. התלמיד יפתור בעיות תוך שימוש בממשק מחסנית. בפרט, התלמיד יקבל מחלקת Stack מוכנה עם ממשק נתון ויכתוב קוד מתאים שפועל מול הממשק.
2. התלמיד יקבל מחסנית כלשהי ויכתוב קוד שמבצע עליה עיבודים שונים. לדוגמא: פיצול המחסנית לשני מחסניות קטנות יותר שכל אחת מהן כוללת טווח ערכים מוגדר.

<u>חלוקת שעות</u>

נושא	שעות
הסבר והדגמת המושג "טיפוס נתונים מופשט" (דוגמא: מחסנית)	1
הכרת ממשק מחלקת StackInt והזיקה שלו למושג "טיפוס נתונים מופשט"	1
פתרון בעיות תוך שימוש בממשק StackInt	4
הסבר מושג הגנריות	1
פתרון בעיות תוך שימוש בממשק <Stack<T	5
סה"כ שעות:	12

כשליש מהשעות מוקדש לתרגול במעבדה. חלוקת השעות המדויקת בין שעות המעבדה לשיעורים התיאורטיים נתונה לשיקול דעתו המקומי של המורה.

פרק 4: תור

<u>יעדים:</u> הכרת טיפוס הנתונים המופשט "תור" (queue); הכרת שימושים נפוצים בתור; תרגול עבודה מול ממשק מחלקת Queue נתון.

<u>תכנים</u>

תור, FIFO

<u>מטרות ביצועיות</u>

1. התלמיד יסביר את טיפוס הנתונים המופשט הגנרי "תור" ואת שימושיו.
2. התלמיד ישתמש בתור כדי לפתור בעיות המאופיינות ע"י תבנית FIFO
3. התלמיד יסביר את משמעות הממשק <Queue<T ואת דרכי העבודה מולו.
4. התלמיד יבדיל בין בעיות שפתרונן מצריך מחסנית לבעיות שפתרונן מצריך תור.

<u>דרכי ההוראה</u>

1. יש להציג את המושג "תור" כטיפוס נתונים מופשט (ADT) שניתן לבצע עליו את הפעולות שמתועדות בממשק מחלקת <Queue<T: בניית תור ריק, בדיקה האם התור ריק, הוספת ערך לזנב תור, הוצאת ערך מראש התור, והחזרת הערך שנמצא בראש התור ללא הוצאתו מהתור.
2. התלמיד יעבוד מול ממשק מחלקת תור גנרי (Queue<T>) המתועד בנספח בסוף הפרק הנוכחי. ממשק זה כולל רק את הפעולות הנחוצות להוראת הפרק.
3. יש להדגיש את ההבדלים בין בעיות שבמהותן מתאימות לפתרון ע"י מחסנית (LIFO) לבין בעיות שבמהותן מתאימות לפתרון ע"י תור (FIFO).
4. יש לחזור ולהדגיש את מושגי המפתח "ממשק" ו- "הכמסה", ולהדגיש שניתן לשנות את מצב התור (דהיינו, לשנות עצם מטיפוס Queue) רק באמצעות פעולות המבוצעות דרך ממשק התור. יש לחזור ולהדגיש שניתן להשתמש בתור באופן מלא בלי לדעת דבר על האופן שבו הוא ממומש. יש לדון שוב ביתרונות ההפרדה בין שימוש למימוש, שהיא מושג מרכזי בתכנות מונחה-עצמים.

<u>דרכי הערכה</u>

בחינה עיונית:

1. התלמיד יפגין יכולת שימוש בממשק מחלקת תור לצורך פתרון בעיות מתאימות.
2. התלמיד יפגין יכולת הבחנה בין המבנים מחסנית ותור, על-ידי מימוש פתרון לשאלה בהתאם למבנה המתאים לפתרונה.
3. התלמיד יפגין יכולת שימוש ב-ADT הניתן לו בעת הבחינה.
4. התלמיד יקבל בעיה כלשהי, ממשק מחסנית גנרי, וממשק תור גנרי. התלמיד יחליט באיזה מודל הוא פותר את הבעיה, עם איזה טיפוס נתונים הוא יאתחל את המבנה הגנרי, ויפתור את הבעיה. לדוגמה: פעולות חשבון על מספרים ארוכים הכוללים סוגריים.

בחינה במעבדה:

1. התלמיד יפתור בעיה תוך שימוש בממשק תור. בפרט, התלמיד יקבל מחלקת Queue מוכנה עם ממשק נתון ויכתוב קוד מתאים שפועל מול הממשק.
2. התלמיד יקבל תור כלשהו ויכתוב קוד שמבצע עליו עיבודים שונים. לדוגמא: פיצול התור לשני תורים קטנים יותר שכל אחד מהם כולל טווח ערכים מוגדר.

<u>חלוקת שעות</u>

נושא	שעות
מושג התור, שימושים נפוצים	1
הכרת הממשק <T>Queue ועבודה מולו	1
הכרת בעיות בעלות תבנית FIFO וכתיבת קוד לפתרונן	5
כיצד מחליטים על מודל לפתרון בעיה: תור או מחסנית?	1
סה"כ שעות:	8

כשליש מהשעות מוקדש לתרגול במעבדה. חלוקת השעות המדויקת בין שעות המעבדה לשיעורים התיאורטיים נתונה לשיקול דעתו המקומי של המורה.

פרק 5: רשימה מקושרת

<u>יעדים</u>: הכרת מבנה הנתונים "רשימה מקושרת" (linked list) או "רשימה" בקיצור, ושימושיו הרבים; מימוש רשימות ואלגוריתמים על רשימות.

<u>תכנים</u>

1. המושגים מצביע, הפנייה, חוליה, ורשימה.
2. בניית ועיבוד רשימות
3. מחלקות Node ו- BinNode
4. הקצאת זיכרון סטטית ודינמית

הערה: ביחידת לימודים זאת התלמיד נתקל לראשונה במושגים "מצביע" ו- "הפניה". אנו משתמשים במונח "מצביע" (pointer) כדי לתאר משתנה שערכו הוא הפנייה (reference), כלומר, כתובת בזיכרון.

הקניית מושגים אלו מצריכה תשומת לב מיוחדת ומומלץ להדגימה בעזרת תרשימי עצמים ותרשימי זיכרון. זוהי גם הזדמנות טובה להסביר באופן מדויק כיצד המחשב מייצר עצם חדש באמצעות פעולת new, דהיינו את האנטומיה של פעולות כגון Point p = new Point(10,15).

<u>מטרות ביצועיות</u>

1. התלמיד יסביר וידגים את המושגים מצביע, הפנייה, חוליה, ורשימה.
2. התלמיד יקבל תרשים של רשימה ויזהה בו דוגמאות למצביע, הפנייה, חוליה, ורשימה.
3. התלמיד יסביר באופן מדויק מה מתחולל מאחורי הקלעים במהלך ביצוע פעולת new.
4. התלמיד יממש אלגוריתמים איטרטיבים ורקורסיבים לעיבוד רשימות: מעבר על רשימה, הוספת, ביטול, וחיפוש חוליה לפי ערך או לפי מקום.
5. התלמיד ינתח את יעילות האלגוריתמים הללו.
6. התלמיד יסביר את היתרונות והחסרונות של עבודה עם רשימה לעומת עבודה עם מערך.
7. התלמיד יבין את הצורך בהגדרת מחלקה שאחת (או יותר) מתכונותיה היא מטיפוס המחלקה.
8. התלמיד ישתמש בממשקים של מחלקות Node ו- BinNode כדי לממש ולעבד רשימות.
9. התלמיד יסביר את ההבדלים בין הקצאת זיכרון סטטית (מערכים) להקצאת זיכרון דינמית (רשימות).

<u>דרכי הוראה</u>

1. שימושים: מבנה הנתונים "רשימה" (אנו משתמשים במונח זה כקיצור למונח "רשימה מקושרת") נמצא בשימוש נרחב ביישומים רבים. כדי לחדד את הצורך ברשימות ולהדגיש את אופיין הדינמי, מומלץ לפתוח את הפרק בדוגמא ויזואלית (תרשים) של יצירת ועיבוד רשימה כלשהי, בלי להיכנס לפרטי מימוש כלשהם.
2. הפנייה: בפרק זה, המושגים "מצביע" ו- "הפניה" משחקים תפקיד מכריע. אי לכך, כחומר רקע למה שיוצג להלן, יש לחזור ולדון בדוגמאות שונות ליצירת ואתחול עצמים שונים, לדוגמא Point p = new Point(10,15). יש להסביר שהפקודה יוצרת עצם חדש מטיפוס Point וגורמת ל p להצביע עליו. בפרט, העצם ממומש כבלוק בזיכרון שמכיל את המספרים 10 ו- 15; כתובת הבסיס של הבלוק הזה נשמרת במשתנה p. כל זאת בהנחה שהבנאי הרלבנטי הוגדר כהלכה במחלקת Point.

3. הפנייה לעצם מאותו טיפוס: עצמים מטיפוס כלשהו שחלק מתכונותיהם ממומשות כהפניות לעצמים מאותו טיפוס הם מושג טבעי בתיכנות מונחה-עצמים. למשל, לעצם מטיפוס Person יכולות להיות תכונות father, mother הממומשות ע"י הפניות לעצמים מטיפוס Person.

4. חוליה: לאחר הדיון במושג ההפניה יש לדון במושג "חוליה": עצם שמחזיק ערך (value) מטיפוס נתון כלשהו והפנייה לעצם נוסף מטיפוס חוליה, או לערך null. אנו נשתמש במינוח המקובל ונתייחס לעצמים הללו כאל מופעים של מחלקת Node נתונה. יש להציג את ממשק מחלקת Node (ראה נספח בסוף הפרק הנוכחי) ולדון בו.

5. יש להשתמש בממשק Node כדי לכתוב קטעי קוד שיוצרים כמה חוליות, משרשרים אותן לרשימה, ומבצעים עליה משימת עיבוד פשוטה כגון הדפסת ערכי הרשימה וחיפוש ערך ברשימה. את המשימות הללו יש לממש במחלקה סטטית כלשהי ולעקוב אחר ביצוען בעזרת תרשימי רשימות מתאימים.

6. רשימה: ככלל, מבנה הנתונים "רשימה" ייוצג וימומש בתכנית הלימודים ע"י מחלקת Node , ותו לא. כלומר, לא נגדיר מחלקת List נוספת שכוללת את טיפוס Node כתכונה, אלא נייצג רשימה ע"י משתנה שמצביע על שרשרת בת אפס או יותר מופעים של מחלקת Node.

7. כפועל יוצא של המוסכמה הזאת, כל משתנה מטיפוס Node יכול להתפרש בשני אופנים שונים: כמצביע על חוליה בודדת, או כמצביע על הרשימה שמתחילה בחוליה הזאת ומסתיימת בהפניה null. כדי לחדד את האבחנה הזאת, מומלץ ליצור שמות משתנים באופן מושכל. לדוגמא, אם כותבים פעולה שתפקידה להוסיף חוליה נתונה לסוף רשימה נתונה, אזי סביר לקרוא למשתנה שנועד לייצג את הרשימה "list" ולמשתנה שנועד לייצג את החוליה "p", או שמות דומים. כלומר, יש לקחת בחשבון מראש את ההקשר האלגוריתמי בו המשתנים מופיעים, ולהעניק להם שמות בהתאם. שימו לב ששני משתנים (או יותר) יכולים בהחלט להצביע בעת ובעונה אחת על אותו עצם מטיפוס Node; משתנה אחד יכול לייצג את החוליה עצמה, ומשתנה אחר את הרשימה שמתחילה בחוליה הזאת – הכל לפי צרכי התכנית.

8. פעולות על רשימות: המשימות האופייניות לעיבוד רשימה הן מעבר על כל ערכי החוליות (וביצוע פעולה כלשהי על הערכים, כגון הדפסה), הוספת חוליה, ביטול חוליה, וחיפוש חוליה – לפי ערך או מקום ברשימה. יש לבצע הן מימוש איטרטיבי והן מימוש רקורסיבי של המשימות הללו. יש לממש את כל המשימות הללו במחלקה סטטית כלשהי ולעקוב אחר ביצוען בעזרת תרשימי רשימות מתאימים.

9. יעילות: בכל פעם שממממשים פעולה כלשהי על רשימה, יש לנתח את יעילות זמן הריצה שלה, כפונקציה של אורך הרשימה. יש לדון ביתרונות ובחסרונות שכרוכים בעבודה עם רשימה לעומת עבודה עם מערך. בצד היתרונות, דינמיות הקצאת הזיכרון; בצד החסרונות, העדר גישה ישירה לאיברים והפגיעה ביעילות בעקבות זאת.

10. גנריות: יש להציג את ממשק המחלקה הגנרית Node<T> ולהשתמש בה כדי לבנות ולעבד רשימות שמכילות טיפוסי נתונים שונים. יש לממש את מחלקת Node<T>, ולדון ולהדגים כיצד מאתחלים רשימה מטיפוס Node <T> כאשר T הוא טיפוס נתונים מסוים.

11. חוליה בינארית: חוליה בינארית דומה לחוליה רגילה, אך מכילה שתי הפניות, כל אחת מהן לחוליה בינארית או לערך null. חוליות בינאריות ימומשו ע"י מופעים של מחלקת BinNode. יש לתרגל בניית

רשימה המורכבת מחוליות BinNode ע״י יצירת מספר חוליות ושרשורן. גם כאן לא נשתמש בשום מבנה נתונים נוסף, אלא נייצר רשימות ע״י מופעים של מחלקת BinNode בלבד.

12. מימוש: בשלב זה יש לפתוח את מחלקות Node ו- BinNode איתן עבדנו עד כה, לקרוא ולהבין את המימוש שלהן.
13. הכמסה וחשיפה: יש לדון בכך שכדי לעבד רשימה באופן סדרתי, יש לחשוף את מבנה הרשימה למשתמש ובכך לגרום לאי-קיום עיקרון ההכמסה. עקב כך, אנו נוהגים להתייחס לרשימה כאל ״מבנה נתונים״ כללי ולא כאל ״טיפוס נתונים מופשט״ ממנו אנו מצפים לקיים את עקרון ההכמסה.

<u>חלוקת שעות</u>

נושא	שעות
בניית רשימות / עיבוד פשוט של רשימות	3
כתיבת ומימוש אלגוריתמים בעזרת רשימות תוך שימוש במחלקת Node<T>	8
כתיבת ומימוש אלגוריתמים בעזרת רשימות תוך שימוש במחלקת BinNode<T>	2
ניתוח יעילות אלגוריתמים לעיבוד רשימות	2
סה״כ שעות:	15

כשליש מהשעות מוקדש לתרגול במעבדה. חלוקת השעות המדויקת בין שעות המעבדה לשיעורים התיאורטיים נתונה לשיקול דעתו המקומי של המורה.

פרק 6: מימוש מבני נתונים

<u>יעדים:</u> בפרקים הקודמים עסקנו בטיפוסי נתונים מופשטים (ADT's) וברשימות מקושרות. בפרק זה "נפתח את הקופסה השחורה" ונלמד כיצד ניתן לממש טיפוסי נתונים מופשטים בדרכים שונות, בפרט באמצעות מערכים ורשימות. כמו כן נממש מבני נתונים מורכבים, כגון מערך של רשימות.

<u>תכנים</u>

1. מימוש של טיפוסי נתונים מופשטים
2. הבנה שלהפשטה נתונה יכולים להיות מימושים אלטרנטיבים, ללא שינוי ההפשטה
3. מימוש הטיפוס מחסנית באמצעות מערך
4. מימוש הטיפוסים תור ומחסנית באמצעות רשימה
5. יעילות המימושים השונים
6. הערה: בפרק זה, כמו בפרק הקודם, כשאנו אומרים "מימוש באמצעות רשימה" הכוונה לרשימה המיוצגת ע"י שרשור חוליות שהן מופעים של מחלקת Node. אין צורך להשתמש בשום מחלקה נוספת לייצוג הרשימה.

<u>מטרות ביצועיות</u>

1. התלמיד יסביר את ההבדלים בין שימוש במבנה נתונים מופשט, לבין מימושו.
2. התלמיד יסביר כיצד ניתן לממש טיפוס נתונים מופשט בדרכים שונות.
3. התלמיד יממש מחסנית באמצעות מערכים ורשימות (שני מימושים אלטרנטיבים)
4. התלמיד יסביר את היתרונות והחסרונות של מימושים שונים.
5. התלמיד יממש תור באמצעות רשימה
6. התלמיד יממש מבני נתונים מורכבים כגון מערך של רשימות ומערך של תורים.

<u>דרכי הערכה</u>

בחינה עיונית:

1. התלמיד יסביר וינמק את היתרונות והחסרונות של מימושים שונים של טיפוס נתונים נתון.

בחינה במעבדה:

1. התלמיד יממש מבני נתונים מורכבים כגון מערך של רשימות ומערך של תורים.

<u>חלוקת שעות</u>

נושא	שעות
מימוש מחסנית ע"י מערך	3
מימוש מחסנית ע"י רשימה	3
מימוש תור ע"י רשימה	2
ניתוחי יעילות של המימושים השונים	1
מימוש מבני נתונים מורכבים	5
סה"כ שעות:	14

כשליש מהשעות מוקדש לתרגול במעבדה. חלוקת השעות המדויקת בין שעות המעבדה לשיעורים התיאורטיים נתונה לשיקול דעתו המקומי של המורה.

פרק 7: עצים בינאריים

<u>יעדים</u>: הכרת עצים בינאריים: בנייה, שימושים, ואלגוריתמי סריקה רלבנטים.

<u>תכנים</u>

1. המבנה והמינוח של עצים.
2. מושג החוליה הבינארית כאבן בנייה בסיסית של עצים בינאריים.
3. אלגוריתמי סריקה של עצים בינאריים.
4. עץ חיפוש בינארי: מבנה הנתונים ואלגוריתמים רלבנטיים.
5. ניתוח יעילות האלגוריתמים.

<u>מטרות ביצועיות</u>

1. התלמיד יקבל תרשים של עץ וידע לזהות בו רכיבים שונים כגון שורש, עלה, מסלול, וכדומה.
2. התלמיד יקבל תרשים של עץ ויכתוב קטע קוד שבונה את העץ ע"י יצירת ושרשור חוליות בינאריות, תוך שימוש בממשק מחלקת BinNode נתונה.
3. התלמיד יסביר איך ניתן להשתמש בעץ בינארי כדי לייצג ולחשב ביטוי חשבוני נתון.
4. התלמיד יסביר את המושגים pre-order, in-order, post-order.
5. התלמיד יממש אלגוריתמי סריקה שונים של עצים: סריקת עומק (depth first) וסריקת רוחב (breadth first). האלגוריתמים ימומשו הן באופן איטרטיבי והן באופן רקורסיבי.
6. בהינתן עץ בינארי ואלגוריתם שפועל עליו, התלמיד ינתח את יעילות האלגוריתם.
7. התלמיד יסביר מהו עץ חיפוש בינרי.
8. התלמיד יממש אלגוריתמי חיפוש והוספה בעץ חיפוש בינארי, וינתח את יעילותם.

<u>דרכי הוראה</u>

1. **עץ:** יש להציג את המושג הכללי "עץ" (לאו דווקא בינארי), ואת המינוח הרלבנטי: צומת, ילד, אח, צאצא, הורה, הורה קדמון, עלה, שורש, רמה, גובה, תת-עץ, מסלול. יש לדון בשימושים רלבנטיים כגון עץ משפחה, עץ קבצים, ייצוג ביטויים חשבוניים, וכדומה.
2. **עץ בינארי:** יש לציין שמכאן ואילך נגביל את הדיון לעצים בינאריים בלבד.
3. **שימושים:** לעצים בינאריים שימושים רבים במדעי המחשב. לדוגמא, יש להדגים שימוש בעץ בינארי כדי לייצג ביטוי חשבוני, ולהעיר שזה האופן שבו המהדר מארגן ומנתח את קוד המקור של שפת תיכנות עילית לפני תרגומו לשפת מכונה. בשלב מאוחר יותר בפרק נשוב לדוגמא הזאת ונדון באלגוריתם שמחשב את הערך של ביטוי חשבוני המיוצג ע"י עץ בינארי.
4. **מבנה עצים:** יש להציג עצים בינאריים שונים ולדון במבנה שלהם. בפרט, יש לדון בשני מקרי קצה: עץ במבנה "שרוד", שהוא למעשה שקול לרשימה, ועץ מלא או כמעט מלא, שעומקו $\log_2 N$ כאשר N הוא מספר הצמתים בעץ. אלו אבחנות שנעשה בהן שימוש בעתיד; בשלב זה מטרתנו לפתח תובנה ראשונית שמבנה העץ מכתיב את יעילות האלגוריתמים שיפעלו עליו.

5. **בניית עצים:** יש להציג את ממשק מחלקת BinNode ולהשתמש בו כדי ליצור עץ בן מספר חוליות. יש לתרגל בנייה "ידנית" (ע"י יצירת ושירשור חוליות) הן של עצי "שרוך" והן של עצים שלמים או כמעט שלמים. בניית העצים תיעשה במחלקה סטטית כלשהי ע"י תיכנות מול מחלקת BinNode מוכנה.
6. **סריקת עצים:** יש לדון באלגוריתמי סריקה שונים של עצים: סריקת עומק (depth first) וסריקת רוחב (breadth first). יש לממש את האלגוריתמים באופן איטרטיבי ובאופן רקורסיבי, ולדון ביעילותם. סריקת הרוחב תמומש בעזרת טיפוס הנתונים תור (Queue) שנלמד בפרקים קודמים ביחידת הלימודים. יש לשוב לדוגמת השימוש בעץ בינארי לייצוג ביטוי חשבוני ולממש אלגוריתם סריקה רקורסיבי שמחשב את ערך הביטוי.
7. **עץ חיפוש בינארי:** יש להציג את הצורך והשימוש בעץ חיפוש בינארי. יש לציין שבניגוד לעצים בהם דנו עד כה, בעץ חיפוש בינארי מוגדר סדר על הערכים. יש להגביל את הדיון ואת הדוגמאות לעצי חיפוש בינארים שהם מלאים או כמעט מלאים.
8. יש לדון בבניית עץ חיפוש בינארי ע"י סדרת הכנסות של ערכים ולהבחין שמבנה העץ נקבע ע"י סדר הערכים בקלט. יש לתכנן את הקלט כך שהעצים שייבנו יהיו מלאים או כמעט מלאים. יש להראות שמספר צעדי החיפוש של ערך כלשהו בעץ חיפוש בינארי מלא או כמעט מלא הוא לכל היותר $\log_2 N$. יש לדון באלגוריתמים לחיפוש והכנסה של ערך נתון לעץ חיפוש בינארי, ולממשם באופן איטרטיבי ורקורסיבי.
9. **חוליה בינארית:** יש לדון במושג "חוליה בינארית", שנייצג ע"י עצם מטיפוס BinNode. עצם מטיפוס BinNode כולל ערך מטיפוס <T> ושתי הפניות left, right לעצמים מטיפוס BinNode, או לערך null. עץ בינארי ייוצג וימומש ע"י שרשור של חוליות בינאריות. שימו לב לדמיון בין האופן בו אנו מייצגים את מבני הנתונים "רשימה" ו- "עץ בינארי", ובונים בהתאם את המחלקות הרלבנטיות: רשימה מבוססת על שרשור חוליות שכל אחת מהן היא עצם מטיפוס Node; עץ בינארי מבוסס על שרשור חוליות בינאריות שכל אחת מהן היא עצם מטיפוס BinNode. אנו מדגישים שוב שאין להגדיר ולממש מחלקות נוספות כגון List ו- Tree; יש לייצג רשימות ועצים באמצעות מחלקות Node ו- BinNode בלבד, בהתאמה.
10. בדומה לדיון שערכנו בפרק הרשימות, משתנה מטיפוס BinNode יכול להצביע הן על חוליה ספציפית בעץ והן על העץ שמשתרג מהצומת הזאת – אבחנה שיש לחדדה באמצעות יצירה מושכלת של שמות משתנים רלבנטיים. במקרים מסוימים כדי לדון בעיבוד עץ, יש לחשוף את מבנה העץ למשתמש ובכך לגרום לאי-קיום עיקרון ההכמסה.

<u>דרכי הערכה</u>

בחינה עיונית:

1. התלמיד יפגין יכולת לממש פעולות על עץ בינרי – הן פעולות שתוארו בפרק והן פעולות חדשות.
2. התלמיד יפגין יכולת מעקב אחר פעולות של עצים.
3. התלמיד יסרוק עצים בצורות שונות.

בחינה במעבדה:

1. התלמיד יממש אלגוריתמים תוך שימוש בממשק העצים.
2. התלמיד יידרש להוסיף ולממש פעולות נוספות על עצים.

<u>חלוקת שעות</u>

נושא	שעות
מושגים בסיסיים של עצים	2
ממשק מחלקת BinNode ובניית עצים בעזרתו	2
סריקת עצים במבנה pre-order, in-order, post-order	3
מימוש רקורסיבי של סריקת עצים	3
כתיבת ומימוש אלגוריתמים שונים בעצי חיפוש בינאריים	6
ניתוח יעילות אלגוריתמים בעצים	2
סה"כ שעות:	18

נושאים תיאורטיים במדעי המחשב

(יחידת לימוד 5)

התלמיד מגיע ליחידת הלימוד החמישית לאחר שסיים את פרק היסודות ומבני נתונים ביחידה הרביעית בתכנית הלימודים. מטרת יחידת הלימודים השלישית היא לפתח נושאים תיאורטיים ברמה גבוהה שחלקה יופנה ללימודי המשך במקצוע "תכנון ותכנות מערכות".

בתי ספר יכולים לבחור ללמד את אחת החלופות הבאות:

- מערכות מחשב ואסמבלי
- מבוא לחקר ביצועים
- מודלים חישוביים
- תכנות מונחה עצמים ב- Java או ב- C#

ממשקי המחלקות שמופיעות ביחידת לימוד 4

לצורך חישוב סיבוכיות זמן הריצה נסמן ב- |T| את סיבוכיות זמן הריצה של הפעולה toString במחלקה T

ממשק המחלקה Stack (מחסנית)

מחלקת StackInt המחלקה מייצגת מחסנית של ערכי int. מחסנית מאופיינת ע"י נקודת הכנסה והוצאה יחידה שמשרה סדר הכנסה/הוצאה LIFO		
בנאי:		
• בונה מחסנית ריקה	Stack()	O(1)
שאילתות:		
• האם המחסנית ריקה?	boolean isEmpty()	O(1)
• מחזיר תיאור של המחסנית כ- [x1, x2, …, xn]. הערך שנוסף לאחרונה למחסנית מופיע ראשון.	String toString()	O(n)
פקודות:		
• דחיפה: מוסיף את הערך x לראש המחסנית.	void push(int x)	O(1)
• שליפה: מחזיר את הערך שנמצא בראש המחסנית, ומסיר אותו מהמחסנית. תנאי קדם: המחסנית אינה ריקה.	int pop()	O(1)
• הצצה: מחזיר את הערך שנמצא בראש המחסנית, בלי לשנות את מצב המחסנית. תנאי קדם: המחסנית אינה ריקה.	int top()	O(1)

מחלקת Stack<T> המחלקה מייצגת מחסנית של ערכים מטיפוס גנרי T מחסנית מאופיינת ע"י נקודת הכנסה והוצאה יחידה שמשרה סדר הכנסה/הוצאה LIFO		
בנאי:		
• בונה מחסנית ריקה	Stack()	O(1)
שאילתות:		
• האם המחסנית ריקה?	boolean isEmpty()	O(1)
• מחזיר תיאור של המחסנית כ- [x1, x2, …, xn]. הערך שנוסף לאחרונה למחסנית מופיע ראשון.	String toString()	O(n·|T|)
פקודות:		
• דחיפה: מוסיף את x לראש המחסנית.	void push(T x)	O(1)
• שליפה: מחזיר את הערך שנמצא בראש המחסנית, ומסיר אותו מהמחסנית. תנאי קדם: המחסנית אינה ריקה.	T pop()	O(1)
• הצצה: מחזיר את הערך שנמצא בראש המחסנית, בלי לשנות את מצב המחסנית. תנאי קדם: המחסנית אינה ריקה.	T top()	O(1)

(אנו מציגים את שני הממשקים זה לצד זה, כדי להדגים את ההכללה של ממשק רגיל לממשק גנרי.)

ממשק המחלקה Queue (תור)

מחלקת <Queue<T> המחלקה מייצגת תור של ערכים מטיפוס גנרי T תור מאופיין ע"י נקודת כניסה ונקודת יציאה וסדר הכנסה/הוצאה FIFO.		
בנאי:		
• בונה תור ריק	Queue()	O(1)
שאילתות:		
• האם התור ריק?	boolean isEmpty()	O(1)
• מחזיר את הערך שנמצא בראש התור, בלי לשנות את מצב התור. תנאי קדם: התור אינו ריק.	T head()	O(1)
• מחזיר תיאור של התור כ- [x1, x2, …, xn]. הערך שנוסף לאחרונה לתור מופיע אחרון.	String toString()	O(n·|T|)
פקודות:		
• מכניס את x לסוף התור	void insert (T x)	O(1)
• מוציא את הערך מראש התור ומחזיר אותו	T remove ()	O(1)

ממשק המחלקה Node (חולייה)

המחלקה הגנרית <Node<T> מייצגת חוליה המכילה ערך מטיפוס גנרי T והפנייה לחוליה או לערך null. ניתן להשתמש במחלקה זאת כדי לייצג רשימה המורכבת משרשרת של אפס או יותר חוליות.		
בנאים:		
• בונה חוליה: משים (מלשון השמה) את ערך החוליה ל- x, ואת ההפניה שלה לערך null	Node (T x)	O(1)
• בונה חוליה: משים את ערך החוליה ל- x, ואת ההפניה שלה לחוליה next. ערכו של next יכול להיות null	Node (T x, Node<T> next)	O(1)
שאילתות:		
• מחזיר את ערך החוליה	T getValue()	O(1)
• מחזיר את החוליה הבאה, או null	Node<T> getNext()	O(1)
• האם יש חוליה נוספת?	boolean hasNext()	O(1)
• מחזיר את ערך החוליה כמחרוזת	String toString()	O(|T|)
פקודות:		
• משנה את ערך החוליה ל- x	void setValue (T x)	O(1)
• משנה את ההפניה לחוליה הבאה ל- next ערכו של next יכול להיות null	void setNext (Node<T> next)	O(1)

(אנו מציגים את שני הממשקים זה לצד זה כדי להדגים את ההכללה של ממשק רגיל לממשק גנרי.)

ממשק המחלקה BinNode (חולייה בינארית)

מחלקת <u>BinNode<T></u>

מייצגת חוליה בינארית מטיפוס BinNode<T> שמכילה ערך מטיפוס T והפניות לשתי חוליות בינאריות.
ניתן להשתמש במחלקה זאת כדי לייצג עץ בינארי המורכב מאפס או יותר חוליות בינאריות.

בנאים:		
• בונה חוליה בינארית: משים (מלשון השמה) את ערך החוליה ל- x, ואת שתי ההפניות שלה ל- null	BinNode (T x)	O(1)
• בונה חוליה בינארית: משים את ערך החוליה ל- x, ואת שתי ההפניות שלה לחוליות הבינאריות left ו- right ערכן של כל אחת משתי ההפניות הללו יכול להיות null	BinNode (BinNode<T> left, T x, BinNode<T> right)	O(1)
שאילתות:		
• מחזיר את ערך החוליה	T getValue()	O(1)
• מחזיר את החוליה השמאלית, או null	BinNode<T> getLeft()	O(1)
• מחזיר את החוליה הימנית, או null	BinNode<T> getRight()	O(1)
• האם יש חוליה משמאל?	boolean hasLeft()	O(1)
• האם יש חוליה מימין?	boolean hasRight()	O(1)
• מחזיר את ערך החוליה כמחרוזת	String toString()	O(|T|)
פקודות:		
• משנה את ערך החוליה ל- x	void setValue (T x)	O(1)
• משים את ההפניה לחוליה השמאלית ל- left ערכו של left יכול להיות null	void setLeft (BinNode<T> left)	O(1)
• משים את ההפניה לחוליה הימנית ל- right ערכו של right יכול להיות null	void setRight (BinNode<T> right)	O(1)

נספח 1: הערות דידקטיות כלליות

נספח זה כולל כמה עצות פדגוגיות כלליות שנוגעות לכלל התכנית, ומיועדות הן למורים והן לכותבי חומרי וספרי לימוד.

1. הפשטה ומימוש: לאורך כל התכנית יש להדגיש את השוני והדואליות בין הפשטה (abstraction) למימוש (implementation). לדוגמא, כשמציגים פעולה (method) כלשהי, חשוב שהפעולה תוצג תחילה באופן מושגי ופונקציונלי בלבד, מנקודת ראותו של המשתמש, ורק לאחר מכן – אם יש צורך בכך – יוצג גם מימוש של הפעולה. יש להדגיש שאת ההפשטה ניתן לממש בדרכים שונות.

2. עקרון הלימוד הספיראלי: עקרון זה דוגל בהצגת נושאי לימוד מורכבים באופן מדורג, ובשלבים שונים לאורך תכנית הלימודים, כולל השלבים המוקדמים. לדוגמא, ביצוע מותנה (שימוש בפקודת if) מתואר בפרק 3 של יחידת הלימוד הראשונה. יחד עם זאת, רצוי לחשוף את התלמידים לפקודות if פשוטות כבר בפרק 1, כשדנים במושגי תכנות בסיסיים. ככלל, אין שום מניעה ללמד את נושאי הלימוד המוצגים במסמך זה בסדר שונה מזה שמתואר בתכנית הלימודים, הכל לפי שיקול דעתו של המורה.

4. נושאי נכונות ויעילות: לאורך כל תכנית הלימודים יש להתייחס, ולו באופן לא פורמלי, לנושאי נכונות ויעילות. לדוגמא, כשדנים בביצוע חוזר ובלולאות, יש לנתח את יעילות ונכונות האלגוריתמים והתכניות הנלמדות, ולא להשאיר את הדיון בנושאים אלה לפרקים נפרדים בתכנית הלימודים.

5. עבודה עם מחלקות ספרייה סטנדרטיות: שפות תכנות מודרניות מצוידות בספריות עשירות בנות אלפי מחלקות מוכנות. ככלל, אין סיבה ללמד את המחלקות הללו, אלא אם כן הן משרתות צורך פדגוגי מוגדר. אם צריך לחשוף את התלמיד למחלקה מסוימת, רצוי לעבוד עם המחלקה הסטנדרטית ולא להמציא מחלקה חדשה שמבצעת שירותים דומים עם סינטקס שונה שהוא לכאורה קל יותר להבנה. אם המורה סבור שמחלקת הספרייה הקיימת יכולה לבלבל את התלמיד, יש להתמקד רק בפעולות מסוימות מתוך המחלקה ולא להתייחס לשאר המחלקה.

6. סגנון לשוני: לאורך כל התכנית, יש ללמד, להדגיש, ולטפח את הסגנון ודרך הביטוי של התלמיד. אלגוריתמים טובים, תכניות טובות, ותיעוד טוב ניחנים בפשטות, חסכנות, ובהירות. מבחינה פדגוגית, העיסוק במדעי המחשב מעניק לנו הזדמנות מצוינת לשפר את סגנון הכתיבה והניסוח של התלמיד. יש לנצל זאת, לפי שיקול דעתו של המורה, ולהפוך את העיסוק בסגנון לחלק חשוב בתכנית הלימודים.

7. עבודה עצמית: כמו בכל מקצוע מאתגר, חלק ניכר מתהליך הלמידה במדעי המחשב חייב להתבסס על עבודה עצמית. בפרט, העיסוק בתיכנות מחייב שעות עבודה נוספות מחוץ לכתה. יש לתת עבודות בית ופרויקטים שמבוססים על השקעת זמן לא רק במעבדה אלא גם מחוץ למסגרת בית הספר.

נספח 2: סביבת הלימוד

תכנית הלימודים במדעי המחשב מניחה שלכל תלמיד יש גישה למחשב מחובר לאינטרנט אם בבית הספר או בביתו. יש לעודד תלמידים להביא את מחשביהם הפרטיים לבית הספר ולעשות בהם שימוש נרחב במהלך הלימודים. במקביל, יש לקבוע כללי עבודה ברורים לשימור תרבות הדיון והלימוד בכתה. בפרט, אין להרשות שימוש במחשבים שאינו קשור לשיעור הנלמד בכתה.

המחשב הפרטי של התלמיד יכיל את כל התוכנות הנדרשות ואת כל חומרי הלימוד הכלולים בתוכנית הלימודים. מחשב זה יהווה מעבדה אישית אותה ניתן לנייד לכל סביבה בה התלמיד פועל: כתת לימוד, מעבדה, בית, בית של חבר, וכדומה. המחשב המתאים ביותר לצורך זה הוא מחשב נייד פשוט או מחשב לוח (tablet PC) המצויד במקלדת.

קיימות גרסאות חינמיות מעולות של כל התוכנות המאפשרות את הוראת מדעי המחשב באופן מלא ומספק; לכן, ככלל, כל התוכנות בהן ישתמשו תלמידים בתכנית הלימודים במדעי המחשב תהיינה חינמיות.

כדי לעודד את התלמידים לעשות שימוש במחשביהם הפרטיים (ניידים או נייחים בבית), יש לפרסם מסמך שמנחה בצורה ברורה ומדויקת כיצד להתקין על מחשב התלמיד את כלי התוכנה וסביבות הפיתוח בהם ייעשה שימוש במהלך לימוד התכנית.

כמו כן, על בית הספר להיערך לאחסון המחשבים הניידים של התלמידים בזמני הפסקות, מבחנים, ופעילות מחוץ לכתה. לדוגמא, אפשר להקצות ארון ייעודי נעול שניתן לאחסן בו את המחשבים. זהו פתרון שמספר בתי ספר משתמשים בו כבר עכשיו בהצלחה.

כדי לתמוך בתלמידים שאין להם גישה למחשב פרטי, בכל בית ספר בו מלמדים מדעי המחשב צריכה להיות מעבדת מחשב. המעבדה יכולה להיות חדר פיזי מצויד במחשבים, או, לחליפין, מספר מחשבים ניידים פשוטים וזולים אותם ניתן לספק לתלמידים שאין ברשותם מחשב פרטי, לפי דרישה. את המחשבים הללו ניתן לנייד לכל כיתת לימוד, דבר שמייתר את הצורך באחזקת מעבדה פיזית ותכנון לוחות זמנים לשימוש בה.

חלק חיוני בסביבת הלימוד של התלמיד הוא אתר אינטרנט שמכיל את כל חומרי הלימוד בהם נעשה שימוש במהלך התוכנית. חומרים אלה כוללים את הטקסט של ספרי הלימוד, כל תכניות המחשב שמוזכרות בספרים, פתרונות לתרגילים נבחרים, מבחנים לדוגמא, תרגילי בית, וכדומה. אי לכך, בחוזה ההתקשרות עם כותבי ספרי לימוד מכאן ואילך יש לדרוש הקמה ותחזוקת אתר אינטרנט כזה, לפי מפרט שייכתב ע״י משרד החינוך.

בנוסף לאתר חומרי הלימוד, לתלמידים צריכה להיות גישה לשני כלי עבודה חיוניים: (א) מערכת להגשת שיעורי בית באופן אלקטרוני, ו- (ב) פורום שאלות ותשובות. מערכות אלו נמצאות כבר בשימוש בבתי ספר רבים.

נספח 3: מרכיב התיכנות בתכנית הלימודים

בדומה למקצועות מדעיים אחרים כמו ביולוגיה, כימיה, ופיסיקה, בלימוד מדעי המחשב יש חשיבות עליונה להתנסות מעשית ולתרגול פרקטי של הרעיונות העיוניים המוצגים בכתה. במדעי המחשב, התרגול נעשה בעיקר ע"י תיכנות: תכנון, כתיבה, בדיקה, והרצת תכניות מחשב.

העיסוק בתיכנות ייעשה בארבעה אופנים שונים:

<u>קריאת תוכניות קיימות:</u> המורה קורא יחד עם התלמידים תוכנית קיימת, ועובר איתם על קטעי קוד נבחרים. התכנית יכולה להילקח מספר לימוד, להיכתב ע"י המורה, או להיות תכנית נבחרת של אחד התלמידים. יש לקרוא הן תכניות טובות בסגנון "ראה וקדש", והן דוגמאות שליליות מהן צריך להימנע; במקרים אלה יש לתעד באופן ברור את הקטעים הבעייתיים.

<u>כתיבת קטעי קוד בכתה:</u> המורה כותב (יחד עם התלמידים) קטעי קוד על הלוח. בכל מקרה של הצגת תכנית בכתה שהיא יותר מכמה שורות של קוד, יש להעלות מראש לאתר בית הספר קבצים שמכילים את התכניות שמוצגות בכתה. זה יאפשר לתלמידים להתרכז בהבנת החומר במקום לעסוק בהעתקתו למחברות. כמו כן, שיטה זאת מאפשרת המשך תרגול במעבדה או עבודה עצמית שכוללים הרחבה של תכנית קיימת שהוצגה בכתה.

<u>תרגול במעבדה:</u> המורה נמצא במעבדת מחשבים עם קבוצת תלמידים ומסייע להם להתמודד עם משימות תיכנות נבחרות. לחליפין, המורה פועל באופן דומה בכיתת לימוד רגילה בה התלמידים מצוידים במחשבים ניידים. אם יש גישה למקרן, ניתן להשתמש בו לצורכי הדגמה, דגש, ודיון. כדי לחסוך זמן מעבדה יקר, מומלץ להכין ולהעלות לאתר בית הספר תכניות כתובות מראש ולהקדיש את זמן המעבדה יותר להרחבת קוד קיים ופחות לכתיבת קוד "מאפס".

<u>עבודה עצמית:</u> כמו בכל מקצוע לימודים אחר, לימוד מדעי המחשב לא מסתיים בכתה, וחייב להימשך בהתעמקות ועבודה עצמית של התלמיד. יש לתת לתלמידים תרגילי תיכנות משמעותיים, ולעודד אותם לתכנת על מחשביהם הפרטיים. אם לתלמיד אין מחשב, יש לאפשר גישה למחשבים שנמצאים ברשות בית הספר מחוץ לשעות הלימודים.

התנסות מעשית בתיכנות מהווה חלק אינטגרלי וחיוני של תכנית הלימודים. כישורי התיכנות של התלמידים ייבדקו במספר אופנים, כדלקמן:

<u>הגשה אלקטרונית של שיעורי בית:</u> מומלץ שהתלמידים יחויבו להגיש את תכניות המחשב אותם הם כותבים במסגרת שיעורי הבית באופן אלקטרוני. גם אם המורה דורש הדפסה והגשה על נייר, מומלץ לדרוש במקביל הגשה אלקטרונית.

<u>ציוני מגן</u> תרגילי התיכנות שיינתנו בשיעורי הבית יקבלו ציונים. סך הציונים על תרגילי התיכנות יהווה חלק מציון המגן של התלמיד במקצוע מדעי המחשב. בחישוב ציון המגן, משקלות תרגילי התיכנות יהיו יחסיים לרמת המורכבות שלהם. יש לפרסם את מדיניות ציוני המגן בתחילת שנת הלימודים ולהדגיש את רכיב תרגילי התיכנות בציון המצרפי. יש לפרסם את הציון המצטבר בתרגילי התיכנות באופן שוטף במהלך השנה כדי להדגיש את חשיבות התרגול ולאפשר לתלמידים שפותחים פער לזהות ולשפר את מצבם.

<u>בחינות הבגרות</u> תכלולנה שאלות שדורשות כתיבת קטעי קוד בשפת התיכנות שנלמדה ברמה דומה לזאת שתידרש בספרי הלימוד. תלמידים שלא יעסקו בתיכנות במהלך תכנית הלימודים לא יוכלו לספק תשובות מספקות לשאלות הללו.

נספח 4: רשימת הנושאים בשפת Java או C# בהתאמה, שכלולים בתכנית הלימודים

נספח זה מפרט את כל תכני שפת Java או שפת C# בהתאמה שכלולים בתכנית הלימודים ועשויים להופיע בבחינות הבגרות הרלבנטיות במדעי המחשב.

בדומה למפרט תכנית הלימודים, מדובר ברשימת מינימום שמורים מוזמנים להרחיב, לפי שיקול דעתם. יחד עם זאת, חשוב לזכור ששפת התיכנות בה נשתמש מהווה כלי להוראת מדעי המחשב ולא מטרה פדגוגית בפני עצמה. בפרט, תכנית הלימודים לא בנויה ולא מתיימרת להפוך את התלמידים למתכנתים מתוחכמים, והוראת או תרגול נושאי תיכנות טכניים תבוא על חשבון הקדשת זמן לימודים יקר לנושאי לימוד אחרים. לכן, מורים שיעודדו את תלמידיהם להתעמק בשפת התיכנות או בספרייה כזאת או אחרת עשויים להחטיא חלקים חשובים מתכנית הלימודים ולגרום להכנה לא מאוזנת של תלמידיהם לבחינות הבגרות במדעי המחשב.

כל רשימת נושאים שמבוססת על תת-קבוצה של שפת תיכנות היא מטבעה קונטרוברסלית, ויכולה לאכזב מורים שרגילים ללמד כלים שאינם כלולים בתכנית. כאמור, כל מורה מוזמן ללמד אספקטים שונים ונוספים של השפה שאינם מצוינים כאן, לפי שיקול דעתו. אבל, החומר הנוסף לא נדרש בתכנית הבסיסית ולא יופיע בבחינות הבגרות הרלבנטיות.

בכל האמור להלן, המונח "תכנית" מתייחס לתכנית הלימודים במדעי המחשב.

1. טיפוסי נתונים בסיסיים כלולים בתכנית: int, double, boolean, char. לא כלולים: short, long, byte, float. הסבר: אין צורך להקדיש זמן לימוד יקר לטיפוסי נתונים שאין להם ערך מוסף משמעותי.
2. אופרטורים אריתמטיים כלולים: +, -, *, /, %
3. אופרטורי הקיצור ++ ו- -- כלולים, אך רק במובן האופרטיבי שלהם, ללא התייחסות לערך שהם מחזירים. לכן לא יתקיים דיון בהבדל בין ++i ל- i++, והשימוש יהיה תמיד דרך ++i. הסבר: מניעת בלבול מיותר (מב"מ).
4. אופרטור ההשמה = כלול. האופרטורים +=, -=, *=, /=, %= אינם כלולים משיקולי מב"מ.
5. האופרטורים היחסיים >=, >, <=, <, !=, == כלולים. האופרטורים הלוגיים !, ||, && כלולים. כל האופרטורים האחרים בשפה אינם כלולים משיקולי מב"מ.
6. האופרטור הטרנרי : ? אינו כלול, כי הערך הקוסמטי שלו אינו שווה את מחיר הבלבול.
7. ההמרות (int) ו- (double) כלולות. אלה ההמרות היחידות שחובה ללמד; המרות אחרות עוסקות בטיפוסי נתונים שאינם נכללים בתכנית.
8. מחלקת וטיפוס הנתונים מחרוזת (String) כלולות בתכנית הלימודים, אך מוגבלות לפעולות הבאות בלבד: charAt, contains, indexOf, length
9. שרשור מחרוזות ומספרים באמצעות האופרטור + כלול בתכנית; התלמידים אמורים להבין את ההמרות ל- String שנעשות ברקע.

10. קיימות שיטות שונות לקריאת קלט מהמשתמש, ותכנית הלימודים אינה מתייחסת אליהן באופן ספציפי. בבחינות הבגרות, בכל מקום שיהיה צורך בביצוע פעולת קלט, הצורך ימומש ויתועד כדלקמן:
קריאה לפעולת קלט שקוראת ומחזירה ערך מסוג int // . . . = int x

11. טכניקת הפלט היחידה שכלולה בתכנית היא System.out.print/ln או Console.Write/ln ב- C# טכניקות אחרות כגון NumberFormat ו- System.out.printf או string.format אינן כלולות.

12. מחלקת Math כלולה בתכנית אך מוגבלת לפעולות הבאות בלבד:
abs, max, min, pow, round, sqrt

13. מבני הבקרה if, if/else, while, for, foreach, return כלולים בתכנית. מבני הבקרה הבאים אינם כלולים בתכנית: do/while, switch, break, continue

14. מערכים חד- ודו- ממדיים (מלבניים בלבד) כלולים בתכנית, וכן גם מערכים של עצמים. איתחול מקוצר של מערכים (לדוגמא int[] array = {2, 3, 5, 7};) כלול. מכיוון שהתכנית מוגבלת למערכים מלבניים, אין צורך לדון בכך שמערך רב-מימדי ממומש ב Java כמערך של מערכים או ב- C# על ידי הגדרה פשוטה. ככלל, יש להתייחס לביטוי כמו x[2][3] או בהתאמה x[2,3] ב- C# כאל תופעה סינטקטית בלי להתעמק בייצוג הפנימי / אובייקטלי של המערך. התכונה array.length ומבנה הבקרה foreach כלולים בתכנית ויש לעודד את השימוש בהם בעבודה עם מערכים.

15. העמסת מושכלת של בנאים (constructor overloading) והבנה של מושג חתימת פעולה (method signature) כלולים בתכנית.

16. פקודת new ושימוש מושכל בבנאים כלולים בתכנית.

17. מימוש מחלקות לפי API נתון כלול בתכנית. עיצוב מחלקות (דהיינו, תכנון ה API) כלול בתכנית רק ברמה של הבנת הנקרא. כלומר, תלמידים אמורים להבין את שיקולי המעצב אך אינם נדרשים לבצע משימות עיצוב בעצמם.

18. הרשאות גישה: כל המחלקות הכלולות בתכנית והמחלקות שהתלמידים יידרשו לכתוב הן public. כל התכונות הן private, פרט למקרים מיוחדים (כמו מחלקות "ערך") בהן ניתן לדון באפשרות להשתמש בתכונות public. פעולות, בנאים, וקבועים הם תמיד public או private. הרשאות הגישה protected ו- package-private אינן כלולות בתכנית.

19. הערות מסוג //, /* ... */, /* ... **/, כלולות בתכנית.

20. המאפיין final (const ב- C#) כלול רק בהקשר של קבועים. המאפיין final בכל הקשר אחר (מחלקות, פעולות, פרמטרים, תכונות) אינו כלול בתכנית.

21. this כלול בתכנית, אך יש להגביל את השימוש בו לקוד של: (א) בנאים עם פרמטרים ששמותיהם זהים לשמות תכונות, ו- (ב) פעולות בהן מתעורר צורך להתייחס לפרמטר מסוג עצם של אותה מחלקה.

22. הפעולות equals, hashcode, ו- clone אינן כלולות בתכנית.

23. השימוש בפקודת import ב- Java והבנה כללית של מושג ה package כלולים בתכנית.

24. מחלקות מקוננות ומחלקות פנימיות אינן כלולות בתכנית.

25. threads אינם כלולים בתכנית.

26. (ArrayIndexOutOfBoundsException, ArithmeticException, IllegalArgumentException כלולות בתכנית. המבנים try / catch / finally ו- throws אינם כלולים בתכנית.

27. ירושה, ממשקים (interface), מחלקות אבסטרקטיות, ופולימורפיזם אינם כלולה בתכנית. יש להכיר את המושגים ירושה וממשקים (interface) כדי להבין את משמעותם בקריאת תיעוד API של מחלקות שונות. יחד עם זאת, כאמור, השימוש בהם אינו כלול בתכנית.

<u>נושאים בעייתיים לידיעה בלבד</u>

בכל שפת תיכנות קיימות יכולות בעייתיות שמומלץ לא להשתמש בהן (דוגמא בולטת: goto). חלק מהיכולות הללו צריכות להילמד ברמת הידיעה כי הן רע הכרחי. בפרט, הן מופיעות בתיעוד API של כמה מחלקות חשובות של השפה ולכן התלמיד צריך להיות מודע לקיומן:

1. פעולות סטטיות ודרך הפעלתן: ClassName.methodName(). יש להדגיש ולהסביר את המוסכמה שאם מחלקה כוללת פעולות סטטיות, היא חייבת לכלול רק פעולות סטטיות.

2. משתנים סטטיים: יש להסביר את הבעייתיות של השימוש במשתנים סטטיים במחלקות שאמורות לייצג טיפוסי נתונים.

3. מחלקות עוטפות (wrapper classes) ושיטות המרה רלבנטיות (autoboxing). ניתן לערוך דיון על כך שבשפה מונחית עצמים טהורה כל טיפוס נתונים שהוא, כולל int, double וכדומה, צריך להיות מיוצג כעצם. מחלקות עוטפות ושיטות המרה רלבנטיות נועדו לתקן עיצוב בעייתי של השפה ולכן יש ללמדן באופן מינימלי בלבד.