תרגילים מתוך קורס **JAVA** אוניברסיטת תל אביב,

תרגיל 1:

בחלק זה נתרגל כתיבת מחלקות המממשות מנשק נתון. לאחר מכן, נשתמש בהן בעזרת תכונת הפולימורפיזם של תכנות מונחה עצמים ב.-Java

המנשק נתונים Vehicle, אותו מרחיבים אשר המנשקים ושני: LandVehicle, ו- SeaVessel.

1. עליכם לכתוב שלוש מחלקות המממשות את המנשקים הנ"ל :

המחלקה .א Boat המנשק את מממשת אשר SeaVessel

המחלקה .ב Jeep המנשק את מממשת אשר LandVehicle

.ג המחלקה HoverCraft המנשקים את מממשת אשר: SeaVessel, LandVehicle

- כל המחלקות בחלק זה ימומשו כחלק מחבילה
- על כל מחלקה להכיל שדות פרטיים אליהם ניתן לגשת עם מתודותgetter ו-setter ציבוריות.

כל אחת משלוש המחלקות מממשת את אחד המנשקים,

ת בו. המוגדרו

- המתודה()launch והמתודה()drive ידפיסו הודעה למסך, כמתואר במנשק, ולא יעשו דבר נוסף.
- המתודה()getDetails תחזיר מחרוזת המייצגת את פרטי כלי התחבורה, ותכיל את שם הכלי, ואת כל השדות שבה לצד הערכים שלהם. דוגמא למחרוזות שתוחזרנה עבור ג'יפ, סירה ורחפת בהתאמה:

```
"Jeep: name:Grand-Cherokee, max-passengers:5, max-speed:210, num-of-wheels:4"

"Boat: name:Caravel, max-passengers:10, max-speed:15"

"Hovercraft: name:Pomornik, max-passengers:140, max-speed:110, num-of-wheels:8"
```

2. עתה נעבור לכתיבת התוכנית (המחלקה)**Main** שתעשה שימוש בטיפוסים שיצרתם בסעיף הקודם. התוכנית תקבל מהמשתמש (דרך ה-`System.in`) סדרה של נתוני כלי רכב, ולבסוף תדפיס דו"ח מסכם לקובץ הפלט.

נתחיל בכתיבת המתודה getVehicleFromUser שהינה בעלת החתימה הבאה:

```
public static Vehicle[] getVehicleFromUser()
```

המתודה תדפיס למסך תפריט שיאפשר למשתמש להגדיר כלי תחבורה, תקלוט את הנתונים המתאימים דרך המקלדת, ותחזיר לבסוף מערך מטיפוס Vehicle (שימו לב שמערך מטיפוס של המנשק Vehicle יכול להכיל מופעים של מחלקות שונות המממשות את המנשק.)

עבור כל כלי תחבורה שנבחר ע"י המשתמש, תבקש המתודה את הנתונים הנדרשים לבנייתו. בגמר הכנסת הנתונים עבור כלי תחבורה מסויים, המתודה תיצור אובייקט מהמחלקה המתאימה (סירה, ג'יפ או רחפת) ותוסיף אותו למערך כלי התחבורה. לאחר מכן תדפיס המתודה הודעה שהכלי התווסף (תוך שימוש במתודת ה-getDetails) של האובייקט החדש .

דוגמא לתוכן חלון הConsole- בגמר הרצת המתודה (קלט המשתמש מופיע בירוק:)

```
Please choose vehicle type:
J – Jeep
B – Boat
H – Hovercraft
X - Exit
J
Please enter name: Grand-Cherokee
Please enter max passengers: 5
Please enter max speed: 210
Please enter num of wheels: 4
Vehicle added: [Jeep: name:Grand-Cherokee, max-passengers:5, max-speed:210, num-of-wheels:4]

Please choose vehicle type:
J – Jeep
B – Boat
H – Hovercraft
X - Exit
H
Please enter name: Pomornik
Please enter max passengers: 140
Please enter max speed: 110
Please enter num of wheels: 8
```

```
Vehicle added: [Hovercraft: name:Pomornik, max-passengers:140, max-speed:110,
num-of-wheels:8]

Please choose vehicle type:
J – Jeep
B – Boat
H – Hovercraft
X - Exit
B
Please enter name: Caravel
Please enter max passengers: 10
Please enter max speed: 15
Vehicle added: [Boat: name:Caravel, max-passengers:10, max-speed:15]

Please choose vehicle type:
J – Jeep
B – Boat
H – Hovercraft
X - Exit
K
Unknown command. Please try again.

Please choose vehicle type:
J – Jeep
B – Boat
H – Hovercraft
X - Exit
X
```

<u>עליכם לממש את המתודה על פי דוגמת הפלט המופיע לעיל.</u>

בדוגמת הרצה זו, המתודה תחזיר מערך מטיפוסVehicle המכיל 3 אובייקטים שנוצרו על פי נתוני המשתמש (המערך יכול להיות גדול יותר ושאר ערכיו יהיוNull).

<u>הערות:</u>

- המשת
- מש יכול להכניס 22 כלי תחבורה לכל היותר. ריצת המתודה תסתיים אם המשתמש בחר באפשרות היציאה בתפריט (הקיש על'X'), או אם הוכנסו 22 כלי תחבורה.
- ניתן להניח שהמשתמש מכניס קלט חוקי (אותיות או מספרים שלמים כנדרש,) אך אם הוקש תו שאינו כלול בתפריט, יש להדפיס למסך "Unknown command. Please try again." ולהדפיס מחדש את התפריט.

3. א. ממשו את המתודהwriteVehiclesToFile אשר מקבלת מחרוזת המייצגת שם קובץ-פלט ומערך מטיפוסVehicle המחזיק אובייקטים של כלי תחבורה, וכותבת דו"ח מסכם לקובץ המובאת בהמשך. על פי הדוגמא

חתימת המתודה:

```
public static void writeVehiclesToFile(String outputFilename, Vehicle[]
vehicles)
```

המתודה תכתוב לקובץ את רשימת כלי התחבורה ביחד עם הפרטים, מסודרים על פי שתי קטגוריות: כלי שיט, כלי רכב (על פי שני המנשקים.) כל כלי תחבורה שעונה לשתי ההגדרות, יופיע בשתי

הרשימות.

עליכם לממש את המתודה כך שתשמור לקובץ את הפלט. הפלט הבא הינו דוגמה עבור הנתונים שהוכנסו בדוגמא של הסעיף הקודם:

```
Land vehicles:

Jeep: name:Grand-Cherokee, max-passengers:5, max-speed:210, num-of-wheels:4
Hovercraft: name:Pomornik, max-passengers:140, max-speed:110, num-of-wheels:8

Sea vessel:

Hovercraft: name:Pomornik, max-passengers:140, max-speed:110
Boat: name:Caravel, max-passengers:10, max-speed:15
```

הערות:

- ייתכן והמערך אותו תקבל המתודה יהיה גדול יותר ממספר כלי הרכב אותו הוא מכיל (כלומר החל ממקום מסויים במערך יתכן וערך התאים הואNull(. ניתן להניח שלפחות במקום הראשון המערך יש כלי תחבורה אחד שאינוnull.

ב. ממשו את המתודהwriteVehiclesSummaryToFile אשר מקבלת מחרוזת המייצגת שם קובץ-פלט ומערך מטיפוסVehicle המחזיק אובייקטים של כלי תחבורה, וכותבת דו"ח מסכם לקובץ על פי הדוגמא המובאת בהמשך.

חתימת המתודה:

```
public static void writeVehiclesSummaryToFile(String outputFilename,
Vehicle[] vehicles)
```

המתודה תכתוב לקובץ את הנתונים הבאים:

א. כמות כלי התחבורה היבשתיים\ימיים
ב. מספר הנוסעים הכולל של כלי התחבורה היבשתיים\ימיים
ג. המהירות המקסימאלית מבין המהירויות המקסימאליות של כלי התחבורה היבשתיים\ימיים
ד. המהירות המינימאלית מבין המהירויות המקסימאליות של כלי התחבורה היבשתיים\ימיים

עליכם לממש את המתודה כך שתשמור לקובץ את הפלט. הפלט הבא הינו דוגמה עבור הנתונים שהוכנסו בדוגמא של הסעיף הקודם:

```
Land vehicles:

Total land vehicles:2
Total passengers possible:145
Max speed:210
Min speed:110

Sea vessel:

Total sea vessel:2
Total passengers possible:150
Max speed:140
Min speed:15
```

הערות:

- ייתכן והמערך אותו תקבל המתודה יהיה גדול יותר ממספר כלי הרכב אותו הוא מכיל (כלומר החל ממקום מסויים במערך יתכן וערך התאים הואNull(. ניתן להניח שלפחות במקום הראשון המערך יש כלי תחבורה אחד שאינוnull.

4. להלן מנשק נוסף בשם VehicleInSpace המייצג כלי תחבורה במרחב הקרטזי:

```
public interface VehicleInSpace{

        // Updates the position of the shape
        public void move(int x, int y);
}
```

- המתודה move תעדכן את ערכי המיקום של הכלי תחבורה הנוכחי, לנקודה. (x,y)

המנשק את עדכנו Vehicle המנשק את שירחיב כך VehicleInSpace. המנשק של הכותרת Vehicle לאחר השינוי אמורה להיות :

```
public interface Vehicle extends VehicleInSpace
```

עתה יכלול המנשק Vehicle גם את המתודות שהוגדרו במנשק VehicleInSpace. לכן, כל מחלקה המצהירה שהיא מממשת את המנשק Vehicle, תצטרך לממש גם את המתודות של VehicleInSpace.

עדכנו את שלושת המחלקות כך שיכללו מימוש למתודות של המנשק VehicleInSpace. בנוסף הוסיפו gettersו -settersלמיקום של כלי התחבורה.

5. ממשו את המתודה:

```
public static double getTravelTime(VehicleInSpace[] vehicles, int source_x,
int source_y, int dest_x, int dest_y, int passengers, boolean land)
```

המתודה מקבלת מערך של כלי תחבורה במרחב (כלומר כלי תחבורה עם מיקום,) קורדינאטות הx- וקורדינאטות הy- של נקודת המקור ונקודת היעד, מספר נוסעים, ומשתנה בוליאני המציין האם מדובר במסלול יבשתי או ימי.

על המתודה לחשב כמה זמן ייקח להעביר את כמות הנוסעים הנתונה מנקודת המקור לנקודת היעד, ולהחזיר את הזמן המינימאלי הנדרש לכך.

הערות

- ניתן להניח כי כל המסלול מנקודת המקור לנקודת היעד הינו כולו בים, או כולו ביבשה.
- המסלול בו נוסעים כלי התחבורה הוא המסלול הקצר ביותר בין שתי נקודות (כלומר הישר בניהן)

- יש לקחת בחשבון גם את זמן הבאת כלי התחבורה לנקודת התחלת המסלול.
- אם לא ניתן להעביר את כמות הנוסעים הנדרשת, המתודה תחזיר מינוס 1.
- ניתן להניח כי כלי התחבורה נעים במהירות המקסימאלית שלהם, והמרחק בין נקודות הינו בקילומטרים (כלומר המרחק בין הנקודה(0,0) לנקודה(1,1) הינו SQRT(2) קילומטרים.
- הזמן אותו מחזירה המתודה מייצג את הרגע בו הנוסעים הגיע ליעד
- יש לקחת בחשבון את כל האפשרויות להבאת הנוסעים ליעדם, ולהחזיר את הזמן המינימאלי הנדרש להבאת כל הנוסעים. אין צורך לקחת בחשבון את הזמן שלוקח לנוסעים לעלות על כלי התחבורה, כלומר מרגע הגעתו לנקודה ניתן להניח כי הוא ממשיך ישירות ליעד.

ניתן להוסיף מתודות עזר במידת הצורך

- ניתן להניח כי כל המיקומים הינם ברביע הראשון, כלומר שתי האורדינאטות חיוביות.
- על כל הנוסעים לנסוע באותו כלי תחבורה, כלומר אין לחלק את הנוסעים בין שני כל תחבורה שונים או יותר, ולכן אם אחד מכלי התחבורה אינו יכול להכיל את כל הנוסעים, לא ניתן להשתמש בו לצורך הנסיעה כלל.

<u>דוגמה:</u>

למשל עבור כלי התחבורה בסעיף הקודם, כאשר נניח כי כולם ממוקמים תחילה בראשית הצירים (2,2), נקודת התחלה(1,1) ונקודת סיום(2,2), כאשר נרצה להעביר 4 נוסעים בדרך יבשתית, על המתודה להחזיר:2*SQRT(2)/210 , כיוון שהג'יפ נוסע במהירות 210 קילומטר לשעה,

הדרך הנוספת היא להשתמש ברחפת, אך כיוון שמהירות הנסיעה של איטית יותר, עדיף להשתמש בג'יפ (כמובן שחישוב מוכיח זאת.)

דוגמה נוספת:

נניח כי כלי התחבורה הינם כלי התחבורה המתוארים בדוגמה הקודמת (מבחינת מהירות וכמות נוסעים מקסימאלית.) כמו כן, הג'יפ ממוקם בנקודה(1,4), והרחפת ממוקמת בנקודה(4,1). נניח כי נרצה להעביר 6 נוסעים מהנקודה(3,1) לנקודה(3,3), דרך היבשה (ולכן מיקום הסירה אינו רלוונטי.)

אופציה א: שימוש בג'יפ בלבד. במקרה זה לא ניתן להשתמש בג'יפ כיוון שאינו יכול להכיל את כל הנוסעים. (לא ניתן להשתמש בג'יפ לסיבובים, שכן על כל הנוסעים להגיע יחד.)

אופציה ב: שימוש ברחפת בלבד.

בחלק זה מותר לשנות את פונקציית ה.*main*- בדקו את עצמכם על ידי יצירת מערך של כלי תחבורה במרחב, וקראו למתודה.

2 תרגיל

המנשק IPAddress של כתובת מייצג למטה המופיע Internet Protocol (IP). לכתובות דוגמאות IP הן:

127.0.0.1
192.168.1.10

כתובת IP, כפי שניתן לראות, מורכבת מארבעה חלקים. ערכו של כל אחד מהחלקים הוא מספר שלם בין 2 ל255- בסעיף זה נממש את המנשק IPAddress על ידי שלושה ייצוגים שונים: הראשון עושה שימוש במחרוזות, השני במערך של מספרים מסוג short ואילו השלישי משתמש בint- יחיד (הסבר מפורט בהמשך.)

א. כתבו שלוש מחלקות שונות המממשות את המנשק IPAddress המוגדר למטה:

1. מחלקה בשם IPAddressString, המממשת את המנשק בעזרת ייצוג פנימי של String.
2. מחלקה בשם IPAddressShort, המממשת את המנשק בעזרת ייצוג פנימי של מערך בגודל 4 של short. כל תא במערך יחזיק מספר בתחום 0..255.
3. בשם מחלקה IPAddressInt, בעזרת המנשק את מממשת ה int יחיד.

לכל אחת מהמחלקות יהיה בנאי המתאים לייצוג הפנימי שלה וכמובן כל אחת מהן מממשת את המנשק. ניתן להניח בחוזה שהבנאים מקבלים קלט תקין IP (בהתאם לייצוג הפנימי של כל ליצירת כתובת

מחלקה.)

```
public interface IPAddress {

            /**
              * Returns a string representation of the IP address,
       e.g.
              *
       "192.168.0.1"
              */
            public String toString();

            /**
                Compares this IPAddress to the specified object
                @param other
                the IPAddress to compare the current against
                @return true if both IPAddress objects represent the
         same
                IP address, false otherwise.
              */
            public boolean equals(IPAddress other);

            /**
                Returns one of the four parts of the IP address. The
         parts
                are indexed from left to right. For example, in the IP
              * address 192.168.0.1 part 0 is 192, part 1 is 168,
                part 2 is 0 and part 3 is 1.
                (Each part is called an octet as its representation
                requires 8 bits.)
                @param index
                The index of the IP address part (0, 1, 2 or 3)
                @return The value of the specified part.
              */
            public int getOctet(int index);
```

```
* 192.168.0.0 - 192.168.255.255
* 169.254.0.0 - 169.254.255.255
*
* This query returns true if this object is a private network
address
*/
        public boolean isPrivateNetwork();

}
```

להלן פירוט לגבי אופן מימוש הייצוגים השונים:

1. מחרוזת – יש להשתמש במחרוזת יחידה לצורך ייצוג כתובת ה IP . כל הפעולות יבוצעו בעזרת מחרוזת זו.
2. מערך – כל אחד מחלקי הכתובת (מספר שלם)2-255 יוחזק בתא במערך.
3. int – נשים לב שכל אחד מחלקי כתובת הIP- הוא מספר שלם בתחום 2-255 (כולל,) לפיכך ניתן לייצג אותו בעזרת 8 ביטים. על כן, את ארבעת חלקי הכתובת ניתן לייצג באמצעות 32 ביטים 4(בתים) וזהו בדיוק גודלו של int.

לפיכך, נשתמש בint- לא כמספר, אלא כרצף בינארי של 32 ביטים. לדוגמא, הכתובת 127.2.2.1 הביטים רצף י"ע תיוצג 22222221**00000000**22222222**01111111**.

החלק הראשון ייוצג ע"י הביטים במקומות 2-7 (משמאל לימין,) החלק השני ע"י הביטים 8-15, השלישי ע"י 16-23 והרביעי ע"י ביטים 24-31.

127 -> 21111111)לימין משמאל 2-7 ביטים(
2 -> 22222222)ביטים 8-15(
2 -> 22222222)ביטים 16-23(
1 -> 22222221)ביטים 24-31(

<u>הנחיה:</u> על מנת להשתמש בייצוג זה, עליכם לדעת איך לחלץ את ערכו של כל בית 8(ביט) מהרצף הבינארי באורך 4 הבתים 32(ביטים) שמרכיב את ה.int- נציע 2 דרכים אפשריות:

1. שימוש באופרטורים על ביטים)|,&,~,>>,<<(להזזה או למיסוך של הביטים ברצף הבינארי כך שיאופסו כל הביטים מלבד אלו השייכים לבית הרצוי.
2. במחלקה שימוש ByteBuffer
   - י"ע בתים 4 בגודל אובייקט צרו ByteBuffer.allocate(4)
   - היעזרו במתודות put(i)/get(i) להצבה ולחילוץ של בית במיקום i.
   - שימו לב שהמתודה get(i) תחזיר את הבית באינדקס i באובייקט ה-ByteBuffer שיצרתם כמשתנה מטיפוס byte עם טווח ערכים של (-128..+127).

כדי לבצע המרה של בית b מטיפוס byte	למשתנה מטיפוס int עם	טווח

הבא בטריק להשתמש ניתן ,זקוקים אנו לו 0..255 הערכים:

int i = (int))b & 0xFF(;

הערה חשובה: עליכם לממש את המתודות באופן שונה בכל מחלקה בהתאם לייצוג הפנימי. אין להמיר את הייצוג הפנימי לייצוג אחר לצורך מימוש פעולה (רק לצורך פלט.) המחלקות השונות לא "ייעזרו" זו ב להשתמש אסור ,למשל(בזו- IPAddressString את לממש מנת על IPAddressInt(.

בנוסף, ממשו את המחלקה IPAddressFactory המגדירה את המתודות הבאות:

```
public class IPAddressFactory {

	public static IPAddress createAddress(String ip) {
		...
	}

	public static IPAddress createAddress(short[] ip) {
		...
	}

	public static IPAddress createAddress(int ip) {
		...
	}
}
```

כל אחת מהמתודות הסטטיות יוצרת אובייקט מטיפוס IPAddress, כשהאובייקט הקונקרטי נקבע על סמך טיפוס הקלט.

הערה: מחלקה שתפקידה היחיד הוא יצור אובייקטים של מחלקות אחרות נקראת *factory class*. מחלקות אלו מסתירות את פרטי יצור האובייקטים מלקוחות של אובייקטים אלו. השימוש בטכניקה זו נועד להסתיר את המחלקות הקונקרטיות שמממשות מנשק.

להלן תכנית המדגימה את השימוש במחלקה IPAddressFactory ובמנשק.

```
public class TestIPAddress {

	public static void main(String[] args) {
		int address1 = -1062731775; // 192.168.0.1
		short[] address2 = { 10, 1, 255, 1 }; // 10.1.255.1

		IPAddress ip1 = IPAddressFactory.createAddress(address1);
		IPAddress ip2 = IPAddressFactory.createAddress(address2);
		IPAddress ip3 = IPAddressFactory.createAddress("127.0.0.1");

		for (int i = 0; i < 4; i++) {
			System.out.println(ip1.getOctet(i));
		}

		System.out.println("equals: " + ip1.equals(ip2));
	}
}
```

בהצלחה!