

שאלה מס' 2 (30 נקודות)

בשאלה זו נכתוב התחלה של משחק מלחמה שבו דמויות נלחמות ויורות זו על זו. בשאלה זו תכתבו 3 מחלקות וממשק אחד. להלן הוראות מפורטות מה יש לכתוב בכל אחת מהמחלקות ובממשק. אתם רשאים להוסיף למחלקות שיטות כרצונכם לנוחיותכם ולפי הצורך.

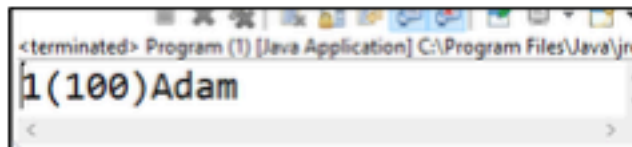
מחלקה בשם `Person` : מייצגת אדם, מכילה 3 תכונות פרטיות: (אתם רשאים לקרוא לתכונות כרצונכם) עבור:
1. ת.ז. (מסוג `int`) - על תכונה זו להיות מוגדרת כך שניתן יהיה לקבוע אותה רק בבנאי ולא יהיה ניתן לשנות אותה אח"כ - גם לא מתוך המחלקה עצמה.
2. שם (מסוג `String`)
3. כמות ה"חיים" (מסוג `int`) - זהו מספר שלם בין 0 ל-100. כאשר אובייקט ה-`Person` נוצר הוא מתחיל עם 100 נקודות חיים. כאשר הדמות נפגעת על ידי דמויות אחרות - נקודות החיים יורדות וכאשר נקודות החיים יורדות ל-0 הדמות נחשבת מתה.

2.1 - הגדירו את המחלקה `Person` לפי הנדרש למעלה. יש להגדיר בנאי מתאים שיאפשר את אתחול התכונות הנ"ל.

2.1.1 - הגדירו מחדש את השיטה `toString` במחלקה כך שהשורה הבאה:

```
System.out.println(new Person(1, "Adam"));
```

תדפיס את הפלט :



2.1.2 - הגדירו במחלקה `Person` שיטה ציבורית בשם `decLifeVal` שתאפשר להוריד לאנשים את ערך החיים. על השיטה לקבל פראמטר יחיד (כמות החיים שיש להוריד לאדם) ותפעל לפי החוקיות הבאה:
אם הפראמטר שהועבר שלילי - אין לשנות את ערך תכונת כמות החיים ויש לזרוק חריגה (עבור שאלה מספר 2) עם ההודעה:

"decLifeVal must get a positive parameter."

אחרת (הפראמטר שהועבר גדול או שווה ל-0) - יש להוריד את תכונת כמות החיים בערך הפראמטר.
אם לאחר ההורדה ערך תכונת כמות החיים קטנה או שווה ל-0 - יש להדפיס למסך הודעה מהצורה:
just died !!!
כאשר ### הוא תוצאת ה-`toString()` של האובייקט. כמו כן, יש לקבוע את תכונת כמות החיים ל-0 (אפילו אם אחרי ההורדה הערך היה קטן מ-0)

2.1.3 - הגדירו מחדש במחלקה `Person` את השיטה `equals` כך ששני אובייקטים מסוג `Person` ייחשבו זהים אם ורק אם הם זהים בתכונת הת.ז. (ובלי קשר כלל ל-2 התכונות האחרות במחלקה).

במידה והפראמטר שהועבר לשיטה `equals` הינו `null` או שאינו מסוג `Person` (או יורש מ `Person`) על השיטה להחזיר את הערך `false`.

שאלה מס' 1 (10 נקודות)

צור חריגה חדשה בשם Exc_A_919. לחריגה החדשה תהיה תכונה מסוג int שתקרא numQ. בתכונה זו תישמר מספר השאלה שבזמן הרצת הקוד שלה התרחשה החריגה.

על החריגה החדשה להיות unchecked. כלומר השיטה הראשית הבאה צריכה להתקמפל בלי בעיות.

```
1 public class Program {
2
3     public static void main(String[] args) {
4         throw new Exc_A_919 (1, "Just an example of how to use Exc_A_919");
5     }
6
7 }
```

בנוסף, עליכם להגדיר מחדש את getMessage() כך שכשהשיטה הראשית הנ"ל תרוץ, על הפלט להראות כך:

```
Exception in thread "main" Exc_A_919 : In question # 1:Just an example of how to use Exc_A_919
at Program.main(Program.java:41)
```

בפתרון השאלות במבחן זה – בכל עת שתבקשו לזרוק חריגה -> יש לזרוק חריגה מסוג Exc_A_919 עם מספר שאלה מתאים.

אין צורך להעתיק את המחלקה Exc_A_919 לפרויקטים האחרים כדי להשתמש בה שם. הפרויקטים של השאלות הנוספות שקיבלתם מאיתנו כבר מכילים להתחשב גם בתיקיית הקוד של הפרויקט Q1.

שאלה מס' 4 (20 נקודות)

נתונה המחלקה הבאה Sell שמייצגת מכירה בחנות בגדים.
כפי שניתן לראות, לכל מכירה שומרים את שם הלקוח, שם המוכר שעזר לו ואת סכום המכירה.
(בשאלה זו נניח, לצורך נוחות, שלכל שני אנשים (לקוחות או מוכרים) יש שם שונה)

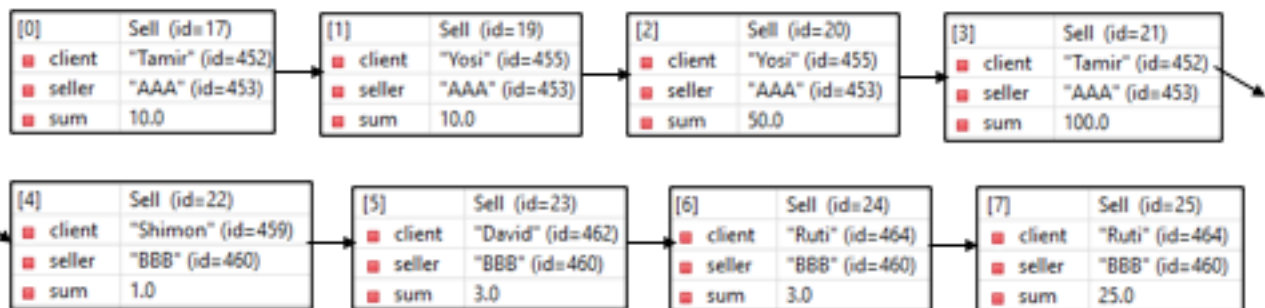
```
public class Sell {  
    private String client;  
    private String seller;  
    private double sum;  
  
    public String getClient(){return client;}  
    public String getSeller(){return seller;}  
    public double getSum(){return sum;}  
}
```

(מחלקה זו כבר חוסמה עבורכם בפרויקט Q4 ואין צורך להוסיף לה שם)

עליכם לכתוב שיטה סטטית בשם bestSeller שמקבלת פראמטר יחיד מסוג java.util.List<Sell> (הממשק שכתוב ב-java.util, להבדיל מהמחלקה בשם List שכתבתם בשאלה 3) ושאינה מחזירה כל ערך. על השיטה לבצע:

- אם הפראמטר שהתקבל הוא null או שהרשימה שהתקבלה כפראמטר היא ריקה על השיטה לזרוק חריגה מסוג Exc_A_919 עבור שאלה 4 עם ההודעה **"illegal list param"**.
- אחרת, (הרשימה שהתקבלה אינה ריקה):
 - יש להדפיס את שם "המוכר המקסימלי". זהו המוכר שמכר להכי הרבה לקוחות שונים, כמו כן יש להדפיס את מספר הלקוחות השונים להם הוא מכר.
 - לאחר מכן להדפיס את רשימת כל הלקוחות שה"מוכר המקסימלי" מכר להם לפי סדר אלפאבתי וליד כל לקוח להדפיס את סכום המכירה הכולל שמכר ה"מוכר המקסימלי" לאותו לקוח (אותו מוכר יכול למכור מספר מכירות שונות לאותו לקוח ולכן יכולים להיות ברשימה הרבה אובייקטים עם אותו שם מוכר ואותו שם לקוח. – יש להדפיס את הסכום המצטבר באובייקטים אלה)

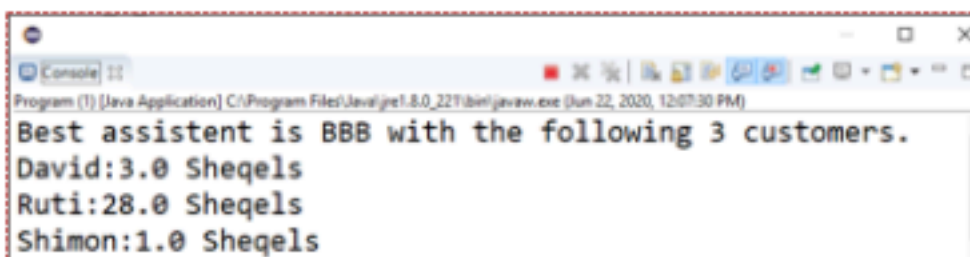
כך לדוגמא, אם
java.util.List<Sell> tempList
היא הרשימה הבאה: (באורך 8 אובייקטים)



אז הקריאה

`bestSeller(tempList);`

צריכה להדפיס את הפלט הבא:



אם כתבתם הכל נכון אז השיטה הראשית הבאה:

```
public static void main(String[] args) throws Exception{

    TimelyList<String> myList = new TimelyList<String>();
    Date dstart = new Date(); // נמדד ונקבע להיות זמן ההתחלה

    // הלולאה הבאה מוסיפה לרשימה 11 איברים (מחרוזות) בהפרש של שניה בין כל הכנסה
    for (int i = 0 ; i <= 10 ; i++){
        System.out.println("Adding item # " + i + " at time " + new Date());
        myList.add("Item "+i);
        Thread.sleep(1000); // נכה גורמים לריצת התוכנית לעצור לשניה אחת
    }
    // השורה הבאה בוחרת את d1 להיות זמן ההתחלה + 3.5 שניות
    // "Item4" להכנסת המחרוזת "Item3" להכנסת המחרוזת
    Date d1 = new Date(dstart.getTime() + 3500);

    // השורה הבאה בוחרת את d2 להיות זמן ההתחלה + 8.5 שניות
    // "Item9" להכנסת המחרוזת "Item8" להכנסת המחרוזת
    Date d2 = new Date(dstart.getTime() + 8500);

    System.out.println("d1 was set to " + d1);
    System.out.println("d2 was set to " + d2);

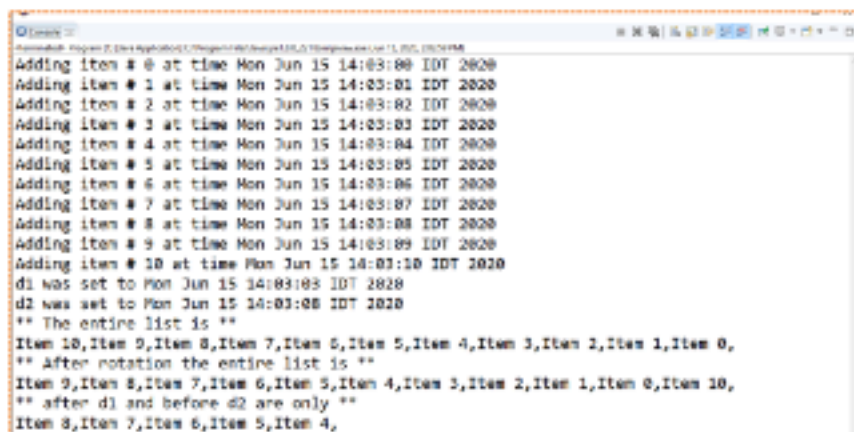
    System.out.println("** The entire list is **");
    for (String item : myList){
        System.out.print(item+",");
    } System.out.println();

    System.out.println("** After rotation the entire list is **");
    myList.remove();
    for (String item : myList){
        System.out.print(item+",");
    } System.out.println();

    System.out.println("** after d1 and before d2 are only **");
    Iterator<String> it = myList.iterator(d1,d2);
    while(it.hasNext()) System.out.print(it.next()+",");

} // end of main
```

אמורה להדפיס את הפלט הבא: (רק שהזמנים שתיראו אצלכם הם בהתאם לזמן שבו תריצו את התוכנית)



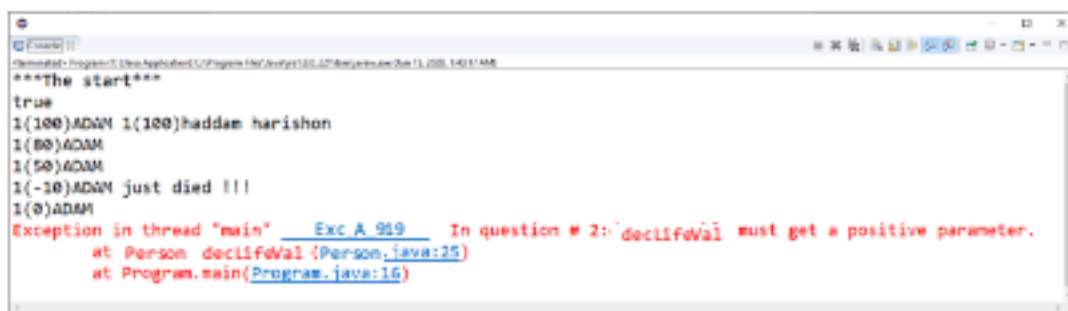
```
Adding item # 0 at time Mon Jun 15 14:03:00 IDT 2020
Adding item # 1 at time Mon Jun 15 14:03:01 IDT 2020
Adding item # 2 at time Mon Jun 15 14:03:02 IDT 2020
Adding item # 3 at time Mon Jun 15 14:03:03 IDT 2020
Adding item # 4 at time Mon Jun 15 14:03:04 IDT 2020
Adding item # 5 at time Mon Jun 15 14:03:05 IDT 2020
Adding item # 6 at time Mon Jun 15 14:03:06 IDT 2020
Adding item # 7 at time Mon Jun 15 14:03:07 IDT 2020
Adding item # 8 at time Mon Jun 15 14:03:08 IDT 2020
Adding item # 9 at time Mon Jun 15 14:03:09 IDT 2020
Adding item # 10 at time Mon Jun 15 14:03:10 IDT 2020
d1 was set to Mon Jun 15 14:03:03 IDT 2020
d2 was set to Mon Jun 15 14:03:08 IDT 2020
** The entire list is **
Item 10,Item 9,Item 8,Item 7,Item 6,Item 5,Item 4,Item 3,Item 2,Item 1,Item 0,
** After rotation the entire list is **
Item 9,Item 8,Item 7,Item 6,Item 5,Item 4,Item 3,Item 2,Item 1,Item 0,Item 10,
** after d1 and before d2 are only **
Item 8,Item 7,Item 6,Item 5,Item 4,
```

אם כתבתם הכל נכון אז השיטה הראשית הבאה :

```
public static void main(String[] args) {
    System.out.println("***The start***");
    Person adm1 = new Person(1, "ADAM");
    Person adm2 = new Person(1, "haddam harishon");
    System.out.println(adm1.equals(adm2));

    System.out.println(adm1+" " + adm2);
    adm1.declifeVal(20); System.out.println(adm1);
    adm1.declifeVal(30); System.out.println(adm1);
    adm1.declifeVal(60); System.out.println(adm1);
    adm1.declifeVal(-30);
    System.out.println(adm1);
    System.out.println("***The end***");
}
```

אמורה לתת את הפלט הבא:



2.2 – הגדירו ממשק (interface) בשם Shooter: על ממשק זה להכיל שיטה בודדת shoot שמקבלת פרמטר יחיד מסוג Person (האדם שעליו יורים) ושאינה מחזירה ערך.

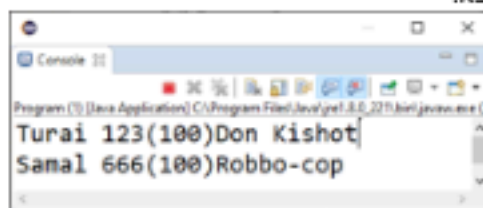
2.3 – הגדירו מחלקה בשם Warrior (שמייצגת חייל) שירשת מ-Person ומממשת את הממשק Shooter. הוסיפו למחלקה Warrior:

1. תכונה פרטית מסוג String שתחזיק את הדרגה של החייל.
2. תכונה ציבורית סטטית בשם ctr מסוג int שתספור כמה אובייקטים מסוג Warrior כבר נוצרו בתוכנית.

2.3.1 – הגדירו מחדש את השיטה toString() במחלקה Warrior והוסיפו למחלקה Warrior בנאים כך שהקוד הבא:

```
Warrior dk = new Warrior(123, "Don Kishot");
System.out.println(dk);
Warrior rc = new Warrior(666, "Robbo-cop", "Samal");
System.out.println(rc);
```

ייתקמפל וכשירץ ידפיס למסך את הפלט הבא:



(שימו לב שהחייל Don Kishot קיבל את דרגת ברירת המחדל "Turai")

2.3.2 – ממשו במחלקה Warrior את השיטה מהממשק Shooter על פי החוקיות הבאה:
אם תכונת כמות החיים של האובייקט הנוכחי קטנה או שווה ל-0 (החייל שאמור לירות בעצם מת) – יש לזרוק
חריגה מסוג Exc_A_919 עם הודעה:

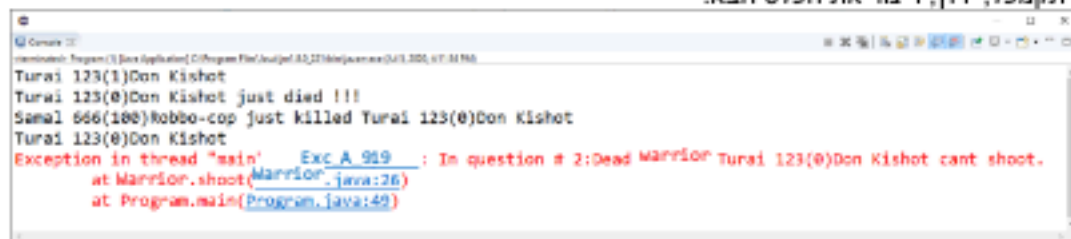
Dead warrior ### cant shoot

כאשר ### הוא תוצאת ה-toString() של האובייקט.
אחרת (החייל שאמור לירות עוד חי) – יש להוריד את כמות החיים של האדם שנורה (ה-Person שהתקבל
כפראמטר לשיטה shoot) ב-1. אם לאחר ההורדה האדם שנורה מת, יש להדפיס הודעה מהצורה:
just killed &&&
כאשר ### הוא תוצאת ה-toString() של החייל ו- &&& הוא תוצאת ה-toString() של האדם שהתקבל כפראמטר.

לאחר שמימשתם את הסעיף, הקוד הבא :

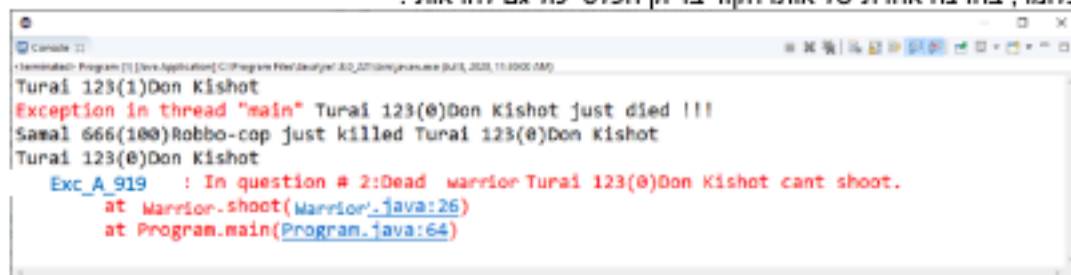
```
Warrior dk = new Warrior(123, "Don Kishot"); // מתחיל עם 100 חיים
dk.decLifeVal(98); // ערך החיים של dk יורד ל-2
Warrior rc = new Warrior(666, "Robbo-cop", "Samal"); // rc הוא חייל
rc.shoot(dk); // rc יורה ב-dk ולכן ערך החיים של dk יורד ל-1
System.out.println(dk); // הדפסתם את מצבו של dk - זוהי השורה הראשונה בפלט
rc.shoot(dk); // rc יורה ב-dk והפעם הורג את dk. אחראי ל-2 שורות הפלט הבאות
System.out.println(dk); // הדפסתם את מצבו של dk - זוהי השורה הרביעית בפלט
dk.shoot(rc); // שורה זו גורמת להריגה כיון ש-dk כבר מת ואינו יכול לירות
```

ייתקמפל, ירוץ, ויצור את הפלט הבא:



```
Turai 123(1)Don Kishot
Turai 123(0)Don Kishot just died !!!
Samal 666(100)Robbo-cop just killed Turai 123(0)Don Kishot
Turai 123(0)Don Kishot
Exception in thread "main" Exc_A_919 : In question # 2:Dead warrior Turai 123(0)Don Kishot cant shoot.
at Warrior.shoot(Warrior.java:26)
at Program.main(Program.java:49)
```

(שימו לב שלפעמים ההודעה האדומה על החריגה "מתערבבת" עם הפלט שלפני למרות שכמובן הודפסה אחרי) –
כלומר, בהרצה אחרת של אותו הקוד בדיוק הפלט יכול גם להראות :



```
Turai 123(1)Don Kishot
Exception in thread "main" Turai 123(0)Don Kishot just died !!!
Samal 666(100)Robbo-cop just killed Turai 123(0)Don Kishot
Turai 123(0)Don Kishot
Exc_A_919 : In question # 2:Dead warrior Turai 123(0)Don Kishot cant shoot.
at Warrior.shoot(Warrior.java:26)
at Program.main(Program.java:64)
```

2.4 – הגדירו מחלקה בשם Officer (שמיצגת קצין – מפקד של מספר חיילים/קצינים) שירשת מ-Warrior.
הוסיפו למחלקה Officer :
תכונה פרטית מסוג מערך של חיילים (אובייקטים מסוג Warrior) שעליהם מפקד הקצין (שימו לב כי קצין גם יכול
לפקד על קצינים אחרים)

2.4.1 - הוסיפו למחלקה Officer בנאי שייקבל (בדומה לחייל) 3 פראמטרים : ת.ז., שם ודרגה. על הבנאי לאתחל
את מערך החיילים לגודל 10.

2.4.2 – הוסיפו למחלקה Officer שיטה בשם add_warriors שתקבל מספר משתנה של פראמטרים מסוג Warrior (חיילים שצריכים להיכנס תחת פיקודו של הקצין) ושתחזיר int. עליכם להוסיף את החיילים למערך החיילים של הקצין לפי החוקיות הבאה:

כל עוד יש במערך מקום ריק (מצביע ל- null) או מקום שמצביע על חייל מת – יש להשתמש במקום זה כדי להכניס חייל מהפראמטרים שהתקבלו.

על כל חייל שלא היה מקום עבורו – יש להדפיס שורה מהצורה:

Could not add new warrior &&& because no room

כאשר &&& הוא תוצאת ה- toString() של החייל עבורו לא היה מקום.

על השיטה להחזיר את מספר החיילים שהצליחה להכניס למערך.

2.4.3 – הגדירו מחדש את השיטה shoot במחלקה Officer לפי הדרישות הבאות:
 "קצין אינו יורה באזרחים" – אם הפראמטר שנשלח ל- shoot הוא לא אובייקט מסוג Warrior – על השיטה להטיל חריגה מסוג Exc_A_919, עבור שאלה מספר 2, עם הודעה:

Officers dont shoot civilians

אחרת (הפראמטר שנשלח לשיטה הוא חייל – יופי, ניתן לירות בו):

○ הקצין יורה (כמו חייל רגיל) במטרה (הפראמטר שנשלח לשיטה)

○ הקצין מפעיל בלולאה את כל החיילים החיים שתחת פיקודו כדי שכל אחד מהם יורה פעם אחת על המטרה.

אם כתבתם הכל נכון, אז השיטה הראשית הבאה:

```
public static void main(String[] args)
{
    Officer cmdr1 = new Officer(20,"Kodkod","Aluf"); // מפקד גדול
    Officer cmdr2 = new Officer(10,"ShuShu","Sagam"); // מפקד קטן
    for (int i = 0 ; i < 3 ; i++){ // לכל אחד מהמפקדים מוסיפים 3 חיילים
        cmdr1.add_warriors(new Warrior(Warrior.ctr,"Hayal"));
        cmdr2.add_warriors(new Warrior(Warrior.ctr,"Hayal"));
    }
    cmdr1.add_warriors(cmdr2); // מוסיפים את המפקד הקטן תחת פיקוד המפקד הגדול

    Warrior x = new Warrior(666,"Enemy"); // חייל אויב
    System.out.println("Before shooting x="+x); // הדפסה לפני (100 חיים)
    cmdr1.shoot(x); // המפקד הגדול יורה באויב – זה גרם לירי של כולם על האויב
    System.out.println("After shooting x="+x); // הדפסה אחרי הירי (92 חיים)

    cmdr2.shoot(new Person(100,"ctzn")); // ירי של קצין על אזרח יגרם לחריגה
}
```

צריכה לייצר את הפלט:

```

C:\code> java Program
Before shooting x=Tural 666(100)Enemy
After shooting x=Tural 666(92)Enemy
Exception in thread "main" Exc_A_919: In question # 2:Officers dont shoot civilians
    at Officer.shoot(Officer.java:23)
    at Program.main(Program.java:54)

```

(שימו לב שלפעמים ההודעה האדומה על החריגה "מתערבבת" עם הפלט שלפני למרות שכמוכן הודפסה אחרי) – כלומר, בהרצה אחרת של אותו הקוד בדיוק הפלט יכול גם להראות:

```

C:\code> java Program
Before shooting x=Tural 666(100)EnemyException in thread "main"
After shooting x=Tural 666(92)Enemy
Exc_A_919 : In question # 2:Officers dont shoot civilians
    at Officer.shoot(Officer.java:23)
    at Program.main(Program.java:54)

```

שאלה מס' 3 (40 נקודות)

להלן המחלקות הגנריות ListNode ו-List הממשות רשימה (בדומה לאלו שנלמדו בכיתה):

```
public class ListNode<T> {  
  
    private T value;  
    private ListNode<T> nextNode;  
  
    public ListNode(T d,ListNode<T> n){  
        value=d; nextNode=n;  
    }  
  
    public T getValue(){ return value;}  
    public void setValue(T val){ value = val; }  
    public ListNode<T> getNext(){ return nextNode; }  
    public void setNext(ListNode<T> next){ nextNode = next;}  
}  
  
public class List <T>{  
  
    protected ListNode<T> head;  
  
    public void add(T insertItem){  
        head=new ListNode<T>(insertItem, head);  
    }  
}
```

(מחלקות אלו כבר חוסמו עבורכם בפרויקט Q3 ואין צורך להזניקם לשם)

3.1 – מעוניינים לשמור בכל ListNode את הזמן המדויק שבו נוצר. לצורך כך עליכם לרשת מהמחלקה הגנרית ListNode מחלקה גנרית חדשה ListTimeNode. יש להוסיף למחלקה ListTimeNode תכונה פרטית מסוג java.util.Date. יש להגדיר את התכונה כך שניתן יהיה לשנות אותה אך ורק בזמן יצירת ה-ListTimeNode. בנוסף יש לכתוב למחלקה החדשה ListTimeNode בנאי שמקבל את אותם פראמטרים כמו ListNode ושמעדכן את התכונה החדשה שהוספתם לזמן הנוכחי.

3.2 – הגדירו מחלקה גנרית חדשה TimelyList שיורשת מ-List. הגדירו מחדש את השיטה add כך שתיצור אובייקט מסוג ListTimeNode במקום ListNode.

3.3 – הוסיפו למחלקה TimelyList שיטה lesovev() שלא מקבלת פראמטרים ולא מחזירה ערך. על השיטה לבצע "גלגול" של האיברים באופן הבא:

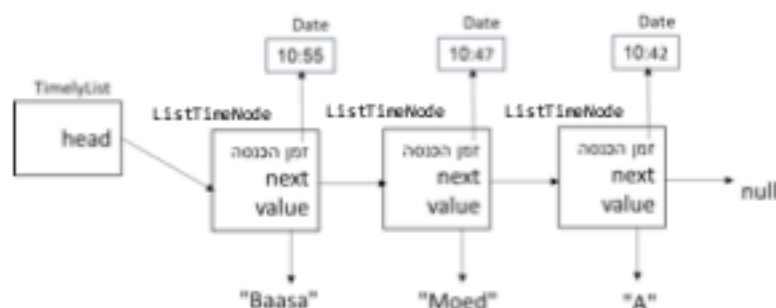
אם יש פחות מ-2 איברים ברשימה – יש להטיל חריגה מסוג Exc_A_919 (עבור שאלה 3) עם ההודעה: "Can not lesovev a list with fewer then 2 items."

אחרת (יש לפחות 2 איברים ברשימה) – יש לקחת את ה-ListTimeNode הראשון (ה-head של הרשימה) ולהעביר אותו לסוף הרשימה. יש כמובן לעדכן את התכונה head כך שתצביע לאיבר הבא שאחרי האיבר שהוזז לסוף.

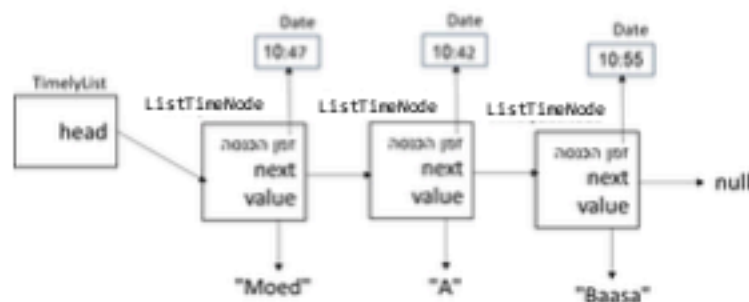
כך לדוגמא, אם הרשימה

`TimelyList<String> myList;`

נראית (לאחר שאותחלה והוכנסו לרשימה 3 איברים בשעות 10:42, 10:47 ו-10:55):



אז לאחר ביצוע השורה `myList.lesovev()` הרשימה תיראה ככה:



בסעיפים הבאים נוסף מנגנון איטרציה (איטרטור) למחלקה `TimelyList`. (אין צורך להוסיף זאת גם ל-`List`). עבור `TimelyList` נכתוב איטרטור מיוחד שמאפשר :

1. לעבור על כל איברי ה-`TimelyList` (כמו איטרטור רגיל). המשתמש יקבל את האיטרטור הזה בכל קריאה לשיטה `iterator()` של `TimelyList` תממש.
2. לעבור רק על איברי הרשימה שזמן ההכנסה שלהם הוא בין 2 תאריכים (`Date`) שנקבעים בזמן היצירה של האיטרטור. המשתמש יקבל את האיטרטור הזה בכל קריאה לשיטה `iterator(Date from, Date to)` של `TimelyList` תממש.

לצורך כך,

3.4 – הגדירו ממשק גנרי `TimelyIterable` שירחיב את הממשק הגנרי `Iterable<T>` כך שיכיל (בנוסף לשיטה `iterator()`) גם הצהרה על שיטה עם חתימה `iterator(Date from, Date to);`

3.5 – הגדירו מחלקה בשם `TimelyIterator` שתממש את הממשק הגנרי `Iterable<T>`. על המחלקה `TimelyIterator` לממש איטרטור לרשימה מסוג `TimelyList`. הוסיפו למחלקה `TimelyIterator` שהגדרתם 2 תכונות פרטיות מסוג `Date`. 2 תכונות אלו יישמרו את טווח הזמן שקובע אילו איברים צריך להחזיר האיטרטור.

(רמז : מצביע מסוג `Date` יכול לקבל את הערך `null` וככה אפשר לדעת אם צריך להתחשב בו כאשר בוחרים אילו איברים האיטרטור צריך להחזיר)

3.6 – שנו את מחלקת הרשימה `TimelyList` כך שתממש את הממשק `TimelyIterable` שהגדרתם בסעיף 3.4. כאמור, על השיטה `iterator()` להחזיר איטרטור שיאפשר מעבר על כל איברי ה-`TimelyList` בלי קשר למתי נוצר ה-`ListTimeNode`. (כמו איטרטור רגיל)

דוגמת הרצה בעמוד הבא