תכנות מונחה עצמים – תרגיל 0

תאריך ההגשה : 15.05.24 בשעה 23:55

תרגיל זה יש להגיש רק ביחידים (לא בזוגות).

0. הקדמה

שימו לב! בהוראות התרגיל משולבים מלבד הוראות הביצוע עצמן גם הרחבות וטיפים.

הסעיפים הממוספרים (למשל 0.1 או 3.4) יכללו את ההוראת שאתם צריכים לבצע.

מלל הנמצא בתבנית כזו מכיל טיפים והרחבה של החומר שהם לא חובה לצורך הגשת התרגיל.

בתרגיל זה נרחיב את השימוש במחלקה ChatterBot. הקובץ עם הקוד של המחלקה נמצא במודל תחת התיקיה Ex0 - Supplied Material.

תזכורת: ChatterBot משתמשת בשיטת replyTo:

```
String replyTo(String statement)
```

אם ההצהרה היא "say <X>", השיטה תחזיר את הפלט <X>. אחרת, היא תחזיר באופן אקראי אחת מהתשובות שסופקו לפונקציית הבנאי עבור מקרה שכזה. כמו כן, מטילה השיטה מטבע כדי לקבוע אם להוסיף לתגובה האקראית את ההצהרה שסופקה.

אתם מוזמנים לעבור על הקוד של ChatterBot!

נשתמש במחלקת ChatterBot ונוסיף לה פונקציונליות נוספת:

1. שם לכל בוט.
2. תמיכה בשיחה מבוססת תור עבור כל מספר של בוטים.
3. נוסיף תמיכה בתשובות מורכבות לבקשות חוקיות. לדוגמה: אותו בוט יוכל לענות לפקודה " say orange" בכל אחת מהדרכים הבאות:
   - orange
   - say orange? Okay: orange
   - say orange yourself!

   ואילן בוט אחר יענה:
   - orange is my favorite thing to say. Here: “orange. orange.”
4. נראה איזה סוגי שיחות תוכלו ליצור באמצעות תבניות תשובה משלכם!

אז לפני שנתחיל:

0.1. עברו על פרק 1.

0.2. התקינו עורך קוד. אפשר להיעזר במדריך ההתקנות שנמצא במודל.

0.3. התקינו Java. היעזרו במדריך ההתקנות שנמצא במודל.

0.4. כמו כן, צריך להיות לכם קובץ בשם ChatterBot.java. העבירו אותו לתיקייה ייעודית. כדאי שהתיקייה הזאת תהיה תיקיית משנה בתוך התיקייה הראשית שתשמש אתכם לשמירת כל התרגילים בקורס.

0.5. פתחו חלון טרמינל ונווטו לתיקייה שבה נמצא הקובץ ChatterBot.java.
ניתן למצוא הסבר לאיך עושים את זה בפרק 3, במדריך ההתקנות.

0.6. וודאו שההידור (קמפול) של הקובץ ChatterBot.java הסתיים בהצלחה. שימו לב שעדיין אי אפשר להריץ את הקוד מכיוון שאין לו שיטת main.

1. השגת אפס פונקציונליות

הדבר הראשון שתמיד נעשה הוא להגיע למשהו שאנחנו יכולים להריץ ולבדוק. הכלל הראשון בתכנות הוא שלא כותבים קוד מבלי לבדוק אותו באדיקות אחרי כל שלב. לכן נתחיל בכתיבת קוד פשוט שרק עובד הידור ורץ (כלומר יש לו אפס פונקציונליות).

1.1. צרו קובץ חדש בשם Chat.java .

1.2. פתחו בעורך הקוד את הקובץ Chat.java החדש ואת הקובץ ChatterBot.java שקיבלתם.

טיפים:

- הוראות ליצירת קובץ ריק חדש תוכלו למצוא בפרק 3, עמוד 7 במדריך ההתקנות.
- כדי לפתוח קבצים בעורך הקוד אפשר לגרור אותם לחלון עורך הקוד.
- ברוב עורכי הקוד אפשר לעבור במהירות בין הקבצים בעזרת קיצור המקשים Ctrl+Tab.

1.3. בקובץ Chat.java, הוסיפו את המחלקה Chat:

```
class Chat {
}
```

1.4. בתוך המחלקה, הוסיפו את השיטה main:

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

תזכורת: Java מחפשת שיטה עם ההצהרה המדויקת הנ"ל. ההצהרה חייבת להיות *public* ו-*static*, עליה להחזיר *void*, שמה חייב להיות *main* (באותיות קטנות) ורשימת הפרמטרים שלה צריכה לכלול רק את מערך המחרוזות שבו לא נשתמש (שם הפרמטר יכול להיות כל דבר ואנחנו משתמשים בשם *args* רק כי כך נהוג). זאת החתימה של השיטה *main*.

1.5. עכשיו אפשר להדר את שני הקבצים האלה ולהריץ את Chat באמצעות הפקודה "`java Chat`". בשלב זה הפקודה לא אמורה להדפיס כלום .

השגנו אפס פונקציונליות. זה עוד לא עושה הרבה אבל אין הודעת שגיאה עכשיו.

2. יצירת צ'אט בסיסי

עכשיו כשהידרנו את הקוד והוא רץ, אפשר להוסיף פונקציונליות. רשימת המטרות שלנו מגדירה את הפיצ'רים שנצטרך להוסיף, אבל מאיפה מתחילים?

כלל אצבע: מתחילים עם הפיצ'ר שיאפשר לבדוק את כל האחרים בצורה הטובה ביותר.

2.1. בתוך השיטה *main*, צרו מערך של שני ChatterBots. עבור בקשות לא חוקיות, ישיב בוט אחד "what" או "say I should say", והבוט השני "whaaat" או "say say". נדרשים כאן שלושה מערכים:

i) מערך אחד המכיל שני מצביעים אל ChatterBot (שכרגע מצביעים לכלום, null).

ii) לכל אחד מהמצביעים האלה יוקצו אובייקטי ChatterBot חדשים. פונקציית הבנאי של ה־ChatterBot מצפה לקבל מערך של מחרוזות. לכן, צריך להתחיל ביצירת ה־ChatterBot הראשון עם המערך {"what", "say I should say"}.

iii) ולהמשיך ליצירת ה־ChatterBot השני עם המערך {"whaaat", "say say"}.

2.2. הידרו את הקוד וודאו שהוא רץ.

תיקון שגיאות הידור

השיטה הטובה ביותר היא להתמקד רק בשגיאת ההידור הראשונה, לתקן את הבעיה ולנסות שוב. צריך לקרוא את הודעות השגיאה של המהדר בסבלנות. הוא באמת משתדל לעזור! בכל שגיאת הידור מוזכר מספר שורה. שימו לב למספר השורה שבהודעת השגיאה וקראו בעיון את תיאור השגיאה. בדרך כלל אפשר למצוא כאן רמז למקור הבעיה.

2.3. צרו את הפרמטר "String statement". מטרתו היא לשמור את המשפט הנוכחי של השיחה. צריך להחליט עבורו על ערך ראשוני כלשהו (אשר אליו יגיב הבוט הראשון).

כדי לגרום לבוטים לנהל שיחה צריך לולאה אינסופית שתעבור בין כל הבוטים שבמערך לפי הסדר (הקוד שלכם אמור לעבוד גם עבור מערכים בכל הגדלים). בעזר הידע שכבר יש לכם ב־Java אתם יכולים לבנות את זה. לשם הרחבת הידע שלכם נציג לכם שלושה מנגנונים של Java שיכולים לעזור:

<u>א. לולאת foreach:</u>

ב־Java, המונח foreach הוא לא מילת מפתח. הוא מתייחס לשימוש מסוים במילת המפתח for שאתם כבר מכירים.
ב-Java יש שתי דרכים להשתמש במילת המפתח for:

1. השימוש המוכר והאהוב:

```
for([initial statement];[condition];[statement to perform after each iteration])
{ ... }
```

לדוגמה:

```
for(int i = 0 ; i > 10 ; i++) {
    System.out.println(i);
}
```

2. דרך נוספת להשתמש במילת המפתח forנקראת לולאת foreach. היא מניחה שכבר יש לכם מבנה נתונים מסוים שאתם רוצים להתייחס לכל אחד מאיבריו לפי הסדר.
לדוגמה, נניח שיש לכם מערך של מספרים שלמים: "int[] arr = {1, 2, 3}". אזי זו דוגמה של לולאת foreach שמדפיסה את האיברים שלו:

```
for(int num : arr) {
    System.out.println(num);
}
```

כמובן, אפשר להריץ תהליך איטרטיבי דומה על מערכים מכל הסוגים ולא רק של מספרים שלמים.

כדי לבצע לולאה איטרטיבית אינסופית על הבוטים, אפשר להשתמש בלולאת foreach באופן דומה לדוגמה הבאה:

```
while(true) {
    for(ChatterBot bot : bots) {
        statement = bot.replyTo(statement);
        System.out.print(statement);
        scanner.nextLine(); // מחכים ל"אנטר" לפני שממשיכים
    }
}
```

שימו לב: מקטע הקוד הזה מניח שהקובץ שלכם מכיל import java.util.Scanner לפני תחילת המחלקה ושהפרמטר "statement" מוגדר ומאותחל .

ב. האופרטור modulo:

כמעט בכל שפות התכנות יש דרך מובנית לחשב את שארית החלוקה של פעולת חילוק. לפונקציה הזאת קורים *modulo* . בשפות תכנות למטרות כלליות (כמו Java), פונקציית modulo מחושבת עם הסימן '%' (אחוז). למשל:

```
5 % 2 == 1 // השארית של חלוקת 5 ב־2 היא 1
12 % 3 == 0 // השארית של חלוקת 12 ב־3 היא 0
27 % 8 == 3
N<M // 17 == 23 % 17 עבור כל, N%M שווה N
```

פונקציית ה-modulo שימושית לאיטרציה על מערכים (במקרה שלנו על הבוטים). ניח שהפרמטר len מבטא את אורך המערך arr. אזי, עבור כל מספר לא שלילי i הביטוי i % len הוא אינדקס חוקי עבור arr (מדוע?). עם קידום הערך של i, עובר הביטוי i % len בין האינדקסים של arr לפי הסדר. כך אפשר להשיג את אותה פונקציונליות של פונקציית ה־foreach הנ"ל גם באופן הבא:

```
Scanner scanner = new Scanner(System.in);
for(int i = 0 ; ; i = i+1) {
statement = bots[ i % bots.length ].replyTo(statement);
System.out.print(statement);
scanner.nextLine();
}
```

נשים לב שהתנאי של ה-for ריק– בלולאת *for* תנאי ריק פירושו לולאה אינסופית (אפשר להחליף בערך *true*).

1. האופרטורים +=, ++

קידום משתנה מספרי הוא פעולה כל כך נפוצה עד שהוא זכה לאופרטור ייעודי משלו:

```
i++; // כמובן זה עובד לכל שם משתנה
```

באופן דומה, הגדלת ערכו של המשתנה *num* בסכום amount מסוים היא פונקציה שימושית מאוד וכמו ב־Python (ואחרות) אפשר לכתוב אותה באופן הבא:

```
num += amount;
```

מכאן שהביטוי

```
i += 1;
```

משרת את אותה מטרה כמו

```
i++;
```

ולכן אפשר לכתוב גם:

```
for(int i = 0 ; ; i++) {
statement = bots[i % bots.length].replyTo(statement);
```

```
System.out.print(statement);
scanner.nextLine();
}
```

כדאי לדעת: ב־Java קיימים אופרטורים נוספים לביצוע פעולות דומות, ובהם --, -=, *=, /=, %=.
אתם מוזמנים לחפש אותם ברשת!

2.4. כתבו לולאה אינסופית שתעבור בין כל הבוטים שבמערך לפי הסדר. כל בוט בתורו יענה על statement ואת תשובתו תשמר ב-statement (הערך החדש ידרוס את הישן).

2.5. הידרו את הקוד וודאו שהוא רץ. כדי לעצור את הלולאה האיסופית או התוכנית אפשר להשתמש ב Ctrl + C.

3. הוספת שם לכל בוט

3.1. הוסיפו השדה "String name" למחלקה ChatterBot.

3.2. ערכו את הבנאי כך שיקבל את השם בתור הפרמטר הראשון ויקצה אותו לשדה.

3.3. הוסיפו את השיטה "String getName()" שתחזיר את השם הזה.

3.4. בקובץ Chat.java, במקום בו מאותחל מערך ה־bots, הקצו להם שמות לבחירתכם. לדוגמה:

```
new ChatterBot("Kermit", … );
```

3.5. בלולאת הצ'אט, הדפיסו את שם הבוט לפני התשובה. הצ'אט יראה ככה:

```
Sammy: what
Ruthy: what what
…
```

טיפ: עדיף להשתמש בשיטה "getName.bot() "על פני "name.bot".

3.6. הדרו את הקוד ובדקו אותו.

4. תשובות מורכבות לבקשות חוקיות

כרגע מגיב הבוט לבקשות בסגנון "say <phrase>" כך: <phrase>. המטרה שלנו היא שהבוט ייתן תשובה מורכבת יותר ולא סתם <phrase>. לדוגמה: בתגובה לבקשה "say door" יענה הבוט משהו

בסגנון של "you want me to say door, do you? alright: door". או משהו כמו " okay, here goes: door".

הבוט ישתמש בתבניות תשובה שלתוכן יכניס במקומות מתאימים את הביטוי *phrase* שעליו התבקש לחזור. הנה דוגמה לתבנית תשובה כזאת:

"You want me to say <phrase>, do you? alright: <phrase>"

לאחר מכן, כשהבוט מקבל את הבקשה "say door" הוא יחליף את כל המופעים של תת המחרוזת <phrase> שבתבנית במחרוזת "door" (אל חשש, אתם תקבלו את הקוד שמבצע את ההחלפה הזאת).

כדי להפוך את הדברים למעניינים עוד יותר, לכל בוט יוגדר מערך של תבניות תשובה והבוט יבחר מתוכו תשובה באופן אקראי. כדי לעשות את זה, תקבל המחלקה פרמטר נוסף בפונקציית הבנאי שלה: מערך של תשובות אפשריות. כך מתקבלת רשימת הפרמטרים הבאה של הבנאי של המחלקה ChatterBot:

```
ChatterBot(String name, String[] repliesToLegalRequest,
                        String[] repliesToIllegalRequest)
```

אם נחזור לדוגמה האחרונה שלנו, אתחול הבוט יתבצע באופן הבא:

```
ChatterBot bot = new ChatterBot(
botty"", // שם הבוט
new String[]{"say <phrase>? okay: <phrase>"}, // תשובה אפשרית לבקשות חוקיות
new String[]{ "what" } // תשובה לבקשות לא חוקיות
);
```

במקרה כזה, השיטה הבאה:

```
bot.replyTo("say door")
```

תחזיר:

```
"say door? okay: door"
```

והשיטה:

```
bot.replyTo("say something")
```

תחזיר:

```
"say something? okay: something"
```

לסיכום: כשמקבל הבוט בקשה חוקית (כלומר בקשה המתחילה בטקסט " say"), יבחר הבוט תשובה באופן אקראי מתוך המערך התשובות לבקשות חוקיות. לאחר מכן, יחליף הבוט את כל המופעים של תת המחרוזת "<phrase>" שבתבנית התשובה שנבחרה בביטוי שבבקשה ויחזיר את התוצאה.

4.1. הוסיפו את השדה "String[] legalRequestsReplies" למחלקה ChatterBot. כמו כן, הוסיפו פרמטר בנאי עם אותו שם ואתחלו את השדה בהתאם לפרמטר.
שימו לב שהאתחול נעשה בעזרת לולאה (לרענון חזרו על הפרק 1.1).

4.2. תקנו את הקריאות לפונקציית הבנאי בקובץ Chat.java. (אם הידור הקוד נכשל, קראו בעיון את הודעת השגיאה ונסו לתקן את השגיאה הראשונה, ואז להריץ שוב.)

4.3. בקוד שמטפל בבקשות חוקיות. ערכו את הקוד כך שבמקום להחזיר את הביטוי אחרי הטקסט " say", שמרו אותו במשתנה מקומי בשם *String phrase*.

4.4. השדה "legalRequestsReplies" מכיל את תבניות התשובה האפשריות. כתבו קוד שיבחר אקראית תבנית אחת.)הקוד הזה כבר נמצא בשיטה*replyToIllegalRequest* שבוחרת באופן אקראי איבר מתוך המערך – כדאי להעיף בה מבט).

4.5. החזירו את התשובה שנבחרה, כשתת המחרוזת "<phrase>" מוחלפת במשתנה המקומי *phrase* שבו שמור הביטוי עצמו. אפשר לעשות זאת עם השיטה: *String.replaceAll*

```
String reply = responsePattern.replaceAll("<phrase>", phrase);
```

4.6. הדרו את הקוד והריצו אותו. וודאו שהבוטים מדברים ביניהם.

4.7. הקוד לטיפול בבקשות חוקיות נמצא כרגע כנראה בשיטה *replyAll*. ייצאו את הקוד הזה לשיטה חדשה:

```
String replyToLegalRequest(String statement)
```

4.8. הדרו את הקוד והריצו אותו.

כעת יש בקוד מחרוזת קסם (Magic String). המחרוזת הזאת נותנת משמעות מיוחדת לתת המחרוזת "<phrase>" גם במחלקה ChatterBot וגם בשיטה main. נניח שרצינו להשתמש במחרוזת שונה למטרה הזאת, לדוגמה: "<the_stuff_the_bot_was_asked_to_say>"? יהיה עלינו לעבור על כל הקוד ולבצע את ההחלפות באופן יזום. כדי לתקן את זה נגדיר את המחרוזת הזו כקבוע במחלקה ChatterBot

4.9. הגדירו את הקבוע הבא במחלקה ChatterBot:

```
static final String PLACEHOLDER_FOR_REQUESTED_PHRASE =
"<phrase>";
```

4.10. שנו את הקוד כך שישתמש בקבוע הזה במקום בתת המחרוזת "<phrase>" המובנית בקוד (בשני הקבצים). שימו לב שכדי לגשת לקבוע הזה מהשיטה *main* צריך לציין קודם כל את שם המחלקה באופן הבא:

```
ChatterBot.PLACEHOLDER_FOR_REQUESTED_PHRASE
```

5. תשובות מורכבות יותר לבקשות לא חוקיות

חשבו על בקשות לא חוקיות. כרגע, בוחר הבוט אחת מבין התשובות המוגדרות ומטיל מטבע כדי להחליט אם להוסיף את הבקשה המקורית אחרי התשובה הזאת. לדוגמה: בתגובה לבקשה הלא חוקית "orange", ובהנחה שהבוט בחר את התשובה "what", יענה הבוט "what" או "what orange".

בחלק זה של התרגיל נשתמש באותו מנגנון שכבר יצרנו לטיפול בבקשות חוקיות, רק שהפעם הוא ישמש לטיפול בבקשות לא חוקיות. כלומר גם כאן התשובות לבקשות הלא חוקיות יכללו תת־מחרוזת המוגדרת בקבוע PLACEHOLDER_FOR_ILLEGAL_REQUEST. לאחר בחירת תשובה מסוימת באופן אקראי, כל המופעים של תת המחרוזת ממלאת המקום יוחלפו בבקשה הלא חוקית עצמה (רק שהפעם בשיעור של 100% מהזמן במקום 50% מהזמן).

לדוגמה: בתגובה לבקשה לא חוקית יענה הבוט משהו כמו:

```
"say what?" + PLACEHOLDER_FOR_ILLEGAL_REQUEST + "? what's" +
PLACEHOLDER_FOR_ILLEGAL_REQUEST + "?"
```

כך שאם נשלחה הבקשה "hello there!", ישיב הבוט:

```
"say what? hello there!? what's hello there!?"
```

כדאי להתחיל בהגדרת הקבוע:

```
static final String PLACEHOLDER_FOR_ILLEGAL_REQUEST =
"<request>";
```

ולהמשיך לבנות על הבסיס הזה.

5.1. הגדירו את הקבוע PLACEHOLDER_FOR_ILLEGAL_REQUEST:

```
static final String PLACEHOLDER_FOR_ILLEGAL_REQUEST =
"<request>";
```

5.2. ערכו את הקוד בשני הקבצים (בדומא לטיפול בבקשות חוקיות) שישתמש בקבוע זה.

שימו לב שעכשיו הטיפול בבקשות חוקיות ולא חוקיות נעשה באופן דומה! לכן אפשר ליצא את הקוד המשותף לשיטה חדשה בשם replacePlaceholderInARandomPattern.
חתימת הפונקציה:

```
replacePlaceholderInARandomPattern(String[] patterns,
String placeholder,
String replacement)
```

5.3. יצאו את הקוד המשותף לשיטה חדשה שתקבל מערך מחרוזות של תשובות אפשרויות שממנו תבחר תשובה, תת־מחרוזת ממלאת מקום שהתשובות האפשריות עשויות להכיל, ואת המחרוזת שתחליף את כל המופעים של אות תת מחרוזת ממלאת המקום.

נקודה למחשבה: מדוע הפונקציונליות הישנה לטיפול בבקשות לא חוקיות (הטלת מטבע כדי להחליט אם להוסיף את הבקשה הלא חוקית לסוף התשובה) היא מקרה מיוחד של הפונקציונליות החדשה? במילים אחרות, איך תשתמשו בפונקציונליות החדשה, מבוססת ההחלפה, כדי לחקות את הפונקציונליות הישנה?

6. סוף דבר

בטח כבר עלו לכם רעיונות לתבניות תשובה משעשעות כתגובה לבקשות חוקיות או לא חוקיות. יש לכם רעיונות לשיחות מעניינות שאפשר ליצור בעזרת המנגנונים שיצרנו? את השיחות שהתקבלו כתבו בקובץ ה-README של התרגיל, ואם השיחה ממש טובה אולי נפרסם אותה!

דוגמא לשיחה:

Kermit: what

Frog: say say what

Kermit: I don't want to say say what

Frog: say say I don't want to say say what

Kermit: I don't want to say say I don't want to say say what

Frog: say say I don't want to say say I don't want to say say what

Kermit: I don't want to say say I don't want to say say I don't want to say say what

Frog: whaaat

Kermit: what whaaat

Frog: whaaat

Kermit: what

Frog: whaaat

Kermit: say what? whaaat? what's whaaat?

Kermit: say what? whaaat? what's whaaat?

Kermit: I don't want to say what? whaaat? what's whaaat?? okay: what? whaaat? what's whaaat?

הרחבה - עיבוד שפה טבעית

אם אי פעם התחשק לכם ליצור בוט עם קצת בינה מלאכותית, יש לא מעט גישות קיימות שיכולות לעזור לכם להתחיל ברגל ימין; חלקן אפילו מאוד פשוטות לשימוש.
אחד מהצ'אטבוטים הוותיקים והמפורסמים ביותר הוא *Eliza* המחקה פסיכולוגית ממוקדת מטופל (פסיכולוגית שתפקידה מצומצם ובדרך כלל תגיב עם שאלות מנחות פשוטות). חפשו " *chat with Eliza bot*": יש כמה וכמה אתרים שיאפשרו לכם להתנסות בשיחה איתה כבר עכשיו, וזה אפילו בחינם. תוכלו גם לקרוא על הרעיון הפשוט שמאחוריה. כיום קיימים כמובן מנועים טובים ומשוכללים יותר: העוזרת האישית שבטלפון שלכם יכולה כנראה לספר לכם משהו על זה.
תת התחום במדעי המחשב שעוסק בניתוח ויצירת טקסט/דיבור בשפה אנושית נקרא "עיבוד שפה טבעית" (ובראשי תיבות באנגלית: *NLP).*

7. טיפים למשתמש

כרגע אנחנו מכירים רק דרך אחת להצהיר על משתנה מערך: "ChatterBot[] bots".

אבל יש עשרות דרכים שונות לאתחל אותו. אנחנו נמנה כמה מהן כאן כדי שתוכלו לבחור בזאת שנראית לכם הכי מתאימה, אבל תוכלו כמובן להשתמש גם בדרכים אחרות שלא מופיעות ברשימה.

<u>אפשרות 1:</u>

```
String[] illegalReplies1 = { "say I should say", "what" };
String[] illegalReplies2 = { "say say", "whaaat" };
ChatterBot[] bots = new ChatterBot[2];
bots[0] = new ChatterBot(illegalReplies1);
bots[1] = new ChatterBot(illegalReplies2);
```

<u>אפשרות 2:</u>

```
ChatterBot[] bots = new ChatterBot[2];
bots[0] = new ChatterBot(
    new String[] {
        "say I should say",
        "what"
    }
);
bots[1] = new ChatterBot(
    new String[] {
        "say say",
        "whaaat"
    }
);
```

אפשרות 3:

```
ChatterBot[] bots = {
        new ChatterBot(
new String[] { "say I should say", "what" }
                ),
        new ChatterBot(
new String[] { "say say", "whaaat" }
                )
        };
```

הערות כלליות בנושא אתחול מערך:

1. התחביר המקוצר:

```
int[] arr = { 1, 2 };
```

חוקי ועובד רק בליווי הצהרת מערך מתאימה.

התחביר הבא לא יעבוד:

```
int [] arr;
arr = { 1, 2 }; // שגיאה!
```

לעומת זאת, התחביר המלא לאתחול מערך תמיד חוקי ויעבוד:

```
int[] arr1;
int[] arr1 = new int { 1, 2 }; // חוקי
int[] arr2 = new int { 1, 2 }; // גם חוקי
```

2. המהדר של Java מתעלם משורות ריקות כך שהבחירה ביצד לרווח את הקוד נתונה לשיקולכם והיא קוסמטית בעיקרה.

7. הוראות הגשה

עליכם להגיש קובץ jar בשם ex0.jar המכיל את הקבצים הבאים:

1. ChatterBot.java – בקובץ זה יופיע המימוש שלכם למחלקה ChatterBot (בהתבסס על התבנית שסופקה לכם). הקפידו על קריאות הקוד, הימנעות מ"קבועי קסם" ותיעוד נכון.
המחלקה ChatterBot אמורה להכיל את השדות הבאות:

- PLACEHOLDER_FOR_REQUESTED_PHRASE
- PLACEHOLDER_FOR_ILLEGAL_REQUEST

ואת השיטות הבאות:

- ChatterBot (Constructor)
- getName
- replacePlaceholderInARandomPattern
- replyTo
- replyToLegalRequest

יתכן שהמחלקה תכלול עוד שדות ושיטות עזר.

2. קובץ Chat.java - בקובץ זה תופיע הפונקציה main.

3. קובץ README –

- בשורה הראשונה בקובץ זה יופיע שם המשתמש שלכם,
- בשורה השנייה יופיע מספר תעודת הזהות שלכם,
- השורה השלישית תהיה ריקה,
- מהשורה הרביעית ואילך עליכם לצרף דוגמה לשיחה של הבוטים שלכם.

שימו לב: הקוד שלכם אמור להיות לפי ה-Coding Style Guideline שנמצא במודל.

8. בדיקת ההגשה:

8.1. לאחר שהגשתם את התרגיל בתיבת ההגשה הקוד שלכם יעבור טסט presubmit ויווצר הקובץ submission.pdf.

8.2. וודאו שהקובץ נוצר בהצלחה.

8.3. הקובץ מכיל פלט של הטסט המוודא שהקוד שלכם מתקמפל, ומפרט על שגיאות בסיסיות. השתמשו בפלט שבקובץ על מנת לתקן שגיאות בתרגיל שימנעו מאיתנו להריץ את הטסטים הסופיים. (זהו טסט קדם הגשה ולא הטסט הסופי של התרגיל.)

8.4. ניתן להגיש את התרגיל שוב ושוב ללא הגבלה עד למועד ההגשה. (ההגשה האחרונה היא ההגשה הסופית.)

8.5. שימו לב: על פי נהלי הקורס חובה לעבור את הטסט ה presubmit ללא שגיאות. קובץ הגשה שלא עובר בהצלחה את הטסט יקבל ציון 0 ולא ייבדק!

8.6. ניתן לחלופין להריץ ישירות את ה presubmit על ידי הרצת הפקודה הבאה (במחשבי בית הספר):
~oop1/ex0_presubmit <path to your jar file>

שימו לב שפקודה זו לא מגישה את התרגיל בפועל אלא רק מריצה את ה presubmit.
חובה לעבור תמיד גם על הפלט של submission.pdf לאחר ההגשה בתיבת ההגשה לוודא שהכל תקין!

בהצלחה!