java 8 features:

Boolean:

פונקציות סטטיות:

- logicalOr פונקציה שמקבלת שני ערכים בוליאנים ומחזירה or שלהם
- logicalAnd פונקציה שמקבלת שני ערכים בוליאנים ומחזירה and שלהם
- logicalXor פונקציה שמקבלת שני ערכים בוליאנים ומחזירה xor שלהם

משתמשים בפונקציות האלה בstream כאשר רוצים
לצמצם reduce ערכים בוליאנים לערך יחיד.

Optional:

אובייקט שעוטף אובייקטים שעלולים להכיל ערכים שהם null. האובייקט מכיל רפרנס לאובייקט אחר.
שימוש ב Optioanl הוצג כפיתרון ל NullPointerException. לפני השימוש ב Optional לא היה ניתן לבטא שיכול להיות לאובייקט ערך null.
פונקציות ליצירת Optional:

- פונקציה סטטית empty: מייצרת אובייקט Optinal ריק.
- פונקציה סטטית of: פונקציה שמקבלת ערך לא null, ויוצרת Optional שעוטף אותו. אם מכניסים ערך null לפונקציה הזאת מקבלים NullPointerException.
- פונקציה סטטית ofNullable: פונקציה שמקבלת ערך שיכול להיות null, ויוצרת Optional שעוטף אותו.

בדיקה המצב של הערך:

- isPresent : בודקת האם הערך קיים או לא.
- isEmpty: (קיים ב Java 11), בודק את ההפך מ isPresent.

ביצוע פעולה עם תנאי:

- ifPresent: פונקציה שמקבלת Function, ומפעילה אותה אם קיים ערך ב Optional.
  דוגמא:
  opt.ifPresent(System.out::println) מדפיס את הערך אם הוא קיים

החזרת הערך:

- get: פונקציה שמחזירה את הערך בתוך ה Optional.
  אם הערך הוא null, נזרקת שגיאה NoSuchElementException.

ערך דיפולטי:

- orElse: משתמשים בו להחזיר את הערך שיש ב Optional, אם הערך לא קיים אז חוזר הערך שהכניסו בפונקציה orElse(T value)
- orElseGet: אותו דבר כמו orElse, רק שהפונקציה מקבלת Consumer.

ההבדל בין הפונקציות הוא:

- כאשר הערך לא קיים: הן פועלות באותה צורה.
- כאשר הערך קיים:
  - orElse: הפונקציה שמגדירה ערך דיפולטי תיקרא, לכן יצרנו סתם אובייקט שאין לנו שימוש בו. (יכול להיות בעייתי גם ב side effects).
  - orElseGet: הפונקציה לא תיקרא.

מניפולציה על המידע: הפונקציות יפעלו רק אם הערך קיים!

- filter: ניתן לפלטר את המידע עם Predicate.
- map: ניתן להפעיל פונקציות לשינוי המידע Function.
- flatMap: כאשר ערך ממחלקה חוזר בעטוף ב Optional, משתמשים ב flatMap.

דוגמא: לוקחים את הגודל של הרשימה, אם היא ריקה הערך הדיפוטי שיחזור הוא 0

```
int size = listOptional
    .map(List::size)
    .orElse(0);
```

חידושים ב Java 9:

- הוספת הפונקציה ()or שמקבל Supplier וומחזירה ערך Optional אלטרנטיבי.
- הוספת הפונקציה ()ifPresentOrElse שמאפשרת הרצת פונקציה אם הערך קיים או פונקציה אחרת אם הערך לא קיים.
- פונקציה ()stream שמאפשרת להפוך Optional ל Stream.

Stream:

ממשק שמאפשר עיבוד של אוסף אובייקטים.
Stream הוא רצף של פעולות שמפעילים על אובייקטים כדי לקבל תוצאה רצויה.
*** אי אפשר למחזר java 8 Streams, כלומר אחרי שקיבלנו תוצאה מStream, אם ננסה לבקש שוב תוצאה מאותו ה Stream נקבל IllegalStateException
פיצרים של Stream:

- Stream לוקח קלט מאוסף, מערך או I/O channels.
- לא משנה את המידע הקיים, מבצע פעולות ומחזיר את התוצאה.
- כל פעולת ביניים היא lazily executed ומחזירה Stream כתוצאה, לכן ניתן לבצע מספר פעולות ביניים. פעולות סיום (Terminal operations) מציינים כל הסוף של ה Stream ומחזיר תוצאה.

פעולות של Stream:

- פעולות ביניים:
  - map: מפעילה פונקציה על האובייקטים ב Stream, ומחזיר Stream של התוצאות.
    כלומר, הפונקציה ממירה Stream מסוג אחד לאחר.
  - flatMap: מפעילה פונקציה על האובייקטים ב Stream, ומחזיר Stream של התוצאות במקרה והערכים עטופים ברשימה זה משטח אותם לאותה רשימה.
  - filter: מפעילה Predicate על האובייקטים, ומחזירה Stream של ערכים שעומדים בתנאי.
  - sorted: ממיינת את הערכים ב Stream.
  - distinct: מחזיר Stream של ערכים שכולם שונים זה מזה.
  - parallel: הופך את ה Stream להיות parallelStream.
- פעולות סיום:
  - collect: ממיר Stream לCollection או Map.
    ישנה מחלקה Collectors שמספקת פונקציונליות להפוך Stream לאוסף. פירוט למטה
  - forEach: פונקציה שמקבלת Consumer ומפעילה אותו על כל אחד מהערכים ב Stream.
  - reduce: פונקציה שמקבלת BiOperator, ומפעילה אותו על כל אחד מהאיברים באוסף, ומצמצמת אותם לערך יחיד. הפונקציה בנוייה מהערכים הללו:

- ערך התחלתי
- ערך צובר
- פונקציה שמקבלת ערך נוכחי, ועושה פעולה בין הערך הצובר לערך הנוכחי, ושומרת את התוצאה בערך הצובר.

  - noneMatch, allMatch, anyMatch: פונקציות שמקבלות Predicate ומחזירות ערך בוליאני אם הערכים ב Stream עומדים בתנאי שיש בפונקציה Predicate.
  - findAny, findFirst: פונקציות שמחזירות ערך עטוף בOptional.
  - count, max, min, sum: פונקציות שאפשר לשתמש בערכים פרמיטיבים כמו IntStream.

יצירת Stream:

- Stream.empty: מחזיר Stream ריק
- collection.stream: יצירת Stream מאוסף.
- Arrays.stream : יצירת Stream ממערך.
- Stream.builder : יצירת Stream לפי ה builder design pattern
- Stream.generate: מקבלת Supplier ומייצרת Stream של ערכים לפי הSupplier. הפונקציה מייצרת ערכים עד "אינסוף" (עד שנגמר הזיכרון), לכן צריך להגביל את הדוגל שלו עם limit.
- Stream.iterate: מקבלת ערך התחלתי, ופונקציה לקדם את הערך, ומייצרת Stream של ערכים. כמו ב Stream.generate צריך להגביל את הגודל.

Stream of Primitives:

- IntStream, LongStream, DoubleStream: ניתן ליצור על ידי הפונקציות range ו-rangeClosed.
- Stream of String: ניתן לפצל String לפי ביטוי רגולרי וליצור מהחלקים שלו Stream.
  Pattern.compile(", ").splitAsStream("a, b, c")
- Stream of File: על ידי הפונקציה Files.lines, כל שורה בקובץ הופכת מחרוזת ב Stream.

פונקציית collect:

- המרת Stream לאוסף: לדוגמא Collectors.toList.
- צמצום לString אחד: שימוש ב Collectors.joining, שמקבלת (delimiter, prefix suffix).
- צמצום לערך פרימיטיבי:
  פונקציות שמקבלות פונקציה כמו ב map שממירה אובייקט לערך, ומבצעת פעולת צמצום על הערכים האלה. הפעולות הן:
  - averagingXXX: מחשבת ממוצע של הערכים ב Stream
  - summingXXX: סוכמת את הערכים ב Stream.
  - summarizingXXX: מחזירה אובייקט מסוג XXXSummarryStatistics שמכיל מידע על האוסף: הממוצע שלו, הערך המקסימלי, הערך המינימלי, כמות האיברים והסכום.
- ביצוע טרנספורמציה נוספת על האוסף:
  על ידי שימוש ב collectingAndThen, נוכל להמיר את הStream לערך ולבצע עליו עוד פעולה לאחר מכן.
  דוגמא:
  Collectors.*collectingAndThen*(Collectors.*toSet*(),Collections::*unmodifiableSet*)
  הפיכת ה Stream ל Set, ואז להפוך אותו ל unmodifiable Set

- קיבוץ ערכים:
  - groupingBy: קיבוץ ערכים לפי הערכים שחוזרים מהרצת ה- Function.
    מחזיר map:
    - המפתח: הערך לפיו חילקנו אתר הרשימה
    - הערך: רשימה של ערכים שמחזירים את הערך כשמריצים עליהם את הפונקציה.
  - partitioningBy: קיבוץ ערכים לפי הערכים שחוזרים מהרצת ה- predicate.
    - מחזיר map:
      - המפתח: true/false
      - הערך: רשימה של ערכים שעמדו בתנאי או לא בהתאמה.

parallelStream:

- ניתן ליצור parallelStream על ידי שימוש בפונקציה parallel.
- parallelStream עובד "מאחורי הקלעים" עם fork-join pool.