# קלט פלט

## Stream

ספריית java.io מספקת לנו כנראה את כל המחלקות שנצטרך אי פעם בשביל לבצע פעולות קלט/פלט. באמצעות המחלקות מספרייה זו נוכל לבנות אובייקטים המייצגים "זרימה" של מידע מסוגים שונים. כיוון שקשה לדמיין אובייקט שמייצג זרימה של מידע, מומלץ לדמיין אותו כאובייקט המייצג צינור שדרכו המידע עובר. באמצעות השיטות המוגדרות על אובייקטים אלו נוכל להזרים מידע ממקום למקום.

Stream (בעברית "זרם") מייצג זרימה של מידע. יש שני סוגים של זרמים:

**InputStream** - זרימה של מידע ממקור מסוים אל התוכנית שלנו (קלט).

**OutputStream** - זרימה של מידע מהתוכנית שלנו אל יעד מסוים (פלט).

בספרייה java.io יש המון מחלקות המייצגות זרימת מידע. ההבדל בין המחלקות הוא בסוג המידע שהן מעבירות (טקסט, טיפוסים פרימיטיביים, אובייקטים וכו'), וסוג הקובץ אליו/ממנו מזרימים מידע. אנו נתעסק בשני סוגי הזרימות המרכזיות שלרוב משתמשים בהם:

**זרימת בתים** - המידע שעובר בזרימה הם בתים (bytes), כל בית בגודל 8-bit. המחלקות המרכזיות בזרימה זו הן: FileInputStream ,FileOutputStream.

**זרימת תווים** - זרמי תווים משמשים לביצוע קלט ופלט עבור תו בגודל 16-bit. המחלקות הנפוצות ביותר בזרימה זו הן FileReader ו-FileWriter. אמנם באופן פנימי FileReader משתמש ב-FileInputStream ו-FileWriter משתמש ב-FileOutputStream, אך כאן ההבדל העיקרי הוא ש-FileReader קורא שני בתים בכל פעם ו-FileWriter כותב שני בתים בכל פעם.

## System.in ו-System.out

## FileInputStream ו-FileOutputStream

### FileInputStream

זוהי מחלקה המייצגת אובייקט זרימה המשמש לקריאה של מידע מקובץ. לא ניתן לכתוב או לשלוח לאובייקט זה בחזרה.

**בנאים** - יש שני בנאים מרכזיים: בנאי המקבל אובייקט מסוג File, ובנאי המקבל מחרוזת של שם קובץ או מסלול אל הקובץ.

**שיטות** - השיטות המרכזיות הן: קריאה אל מערך בתים, דילוג על בתים, חישוב כמה עוד בתים נותרו לקרוא, וסגירת אובייקט הזרימה. חשוב לאחר שמסיימים לקרוא מאובייקט זה לסגור את הקובץ, ובכך לשחרר כל משאבי המערכת שבהם הוא השתמש.

כל השיטות והבנאים ניתן למצוא בקישור:

<https://docs.oracle.com/javase/9/docs/api/java/io/FileInputStream.html>

### FileOutputStream

זוהי מחלקה המייצגת אובייקט זרימה המשמש לכתיבה של מידע אל קובץ. לא ניתן לקרוא מאובייקט זה.

**בנאים** - יש שני בנאים מרכזיים: בנאי המקבל אובייקט מסוג File, ובנאי המקבל מחרוזת של שם קובץ או מסלול אל הקובץ. שני בנאים נוספים הזהים לשני הבנאים שתיארנו, אלא שבהם מוסיפים לבנאי משתנה בוליאני append, שאם הוא true אז כל מה שנכתוב לקובץ יתווסף לאחר מה שכבר כתוב בו ולא ידרוס את ההתחלה כמו בבנאים הראשונים שציינו.

**שיטות** - השיטות המרכזיות הן: כתיבה של בית אחד, כתיבה של מערך בתים, וסגירת אובייקט הזרימה. גם כאן ובכל המחלקות הבאות, נצטרך לסגור את הזרימה בסוף השימוש באובייקט.

כל השיטות והבנאים ניתן למצוא בקישור:

<https://docs.oracle.com/javase/9/docs/api/java/io/FileOutputStream.html>

## קריאת וכתיבת אובייקט

המחלקות המשמשות לקריאת וכתיבת אובייקט לקובץ הן: ObjectInputStream, ObjectOutputStream. אך לפני שנלמד מחלקות נסביר על המנגנון שמאפשר בכלל לקרוא ולכתוב אובייקט.

### Serialization

java מספקת מנגנון, הנקרא Serialization, באמצעותו ניתן לייצג אובייקט **כרצף סדרתי של בתים** הכולל את סוג האובייקט וכל השדות בתוך האובייקט ואת סוגם. סדר השדות ברצף הבתים הוא לפי סדר הכתיבה של השדות במחלקת האובייקט. באמצעות מנגנון זה נוכל לכתוב אובייקט לקובץ באמצעות מחלקה ObjectOutputStream, וכן ניתן לקרוא אותו מקובץ באמצעות המחלקה ObjectInputStream.

כדי שניתן יהיה לכתוב אובייקט ל-ObjectOutputStream המחלקה וכל השדות שלו חייבות לממש את הממשק Serializable. אם יעשה ניסיון לכתוב אובייקט שהוא או אחד משדותיו לא מממש את Serializable, תיזרק השגיאה NonSerializalizableException. אם מחלקה יורשת ממחלקה שמממשת את Serializable היא אינה צריכה לממש אותה שוב. בממשק Serializable אין אף שיטה והוא משמש לצורך סימון בלבד.

אם נרצה ששדה כלשהו של האובייקט לא ייכתב לקובץ נצטרך להוסיף להצהרה של שדה זה ( לפני הצהרת הטיפוס) את המילה השמורה transient.

public transient int unserializable;

לאחר שנכתב אובייקט לקובץ באמצעות serialization, ניתן לקרוא אותו מהקובץ ולשחזר אותו בדיוק כמו האובייקט המקורי. התהליך כולו אינו תלוי ב-JVM, אלא ניתן לכתוב את האובייקט בפלטפורמה אחת ולקרוא אותו מהקובץ בפלטפורמה שונה לחלוטין. נהוג שהסיומת של הקובץ שאליו כתבנו את האובייקט יהיה עם הסיומת '.ser' .

### ObjectInputStream

זוהי המחלקה שבאמצעותה אנו קוראים אובייקט מקובץ.

**בנאים** - יש בנאי אחד המקבל InputStream, בדרך כלל פרמטר זה יהיה מסוג FileInputStream.

**שיטות** - השיטה המרכזית במחלקה זו היא readObject() המחזירה את האובייקט השמור בקובץ באמצעות deserialization. צריך לעשות casting על האובייקט המוחזר משיטה זו. ראה דוגמא בקובץ נפרד.

כל השיטות והבנאים ניתן למצוא בקישור:

<https://docs.oracle.com/javase/9/docs/api/java/io/ObjectInputStream.html>

### ObjectOutputStream

זוהי המחלקה שבאמצעותה אנו כותבים אובייקט מקובץ.

**בנאים** - יש בנאי אחד המקבל InputStream, בדרך כלל פרמטר זה יהיה מסוג FileOutputStream.

**שיטות** - השיטה המרכזית במחלקה זו היא writeObject(object obj) העושה serialization לאובייקט ושומרת אותו בקובץ. ראה דוגמא בקובץ נפרד.

כל השיטות והבנאים ניתן למצוא בקישור:

<https://docs.oracle.com/javase/9/docs/api/java/io/ObjectOutputStream.html>

### קריאת וכתיבת טיפוסים פרימיטיביים

בשתי המחלקות שתיארנו למעלה קיימות שיטות גם לכתיבת טיפוסים בסיסיים כמו int, double וכו'. אמנם בשביל טיפוסים בסיסיים אלו כדאי יותר להשתמש בשיטות המובנות במחלקות DataInputStream ו-DataOutputStream. משתמשים בהם באותה צורה כמו בשתי המחלקות לעיל.

<https://docs.oracle.com/javase/9/docs/api/java/io/DataInputStream.html>

<https://docs.oracle.com/javase/9/docs/api/java/io/DataOutputStream.html>

## FileReader ו-FileWriter

אלו הן שתי מחלקות המייצגות זרימה של **תווים**. דומים ל-FileInputStream ו-FileOutputStream רק שמשתמשים בהן כדי לכתוב ולקרוא תווים ולא בתים. המחלקות שמקבלות מחלקות אלו כאובייקטים ועליהן בפועל נפעיל את השיטות הן: BufferedReader, PrintWriter.

# JSON

## הגדרה

JSON (ראשי תיבות של JavaScript Object Notation) הוא פורמט טקסטואלי קל להבנה, המיועד להעברת מידע בצורה של מפתח-ערך (כמו טיפוסים פשוטים, מערכים, ואובייקטים), באמצעות מחרוזות טקסט פשוטות. קבצים בפורמט זה הם קבצי טקסט פשוטים בעלי סיומת '.json' . אף על פי שפורמט זה פותח בתחילה לשימוש ב-JavaScript ונעשה בו שימוש נרחב בשפה זו, JSON הוא מבנה נתונים לא תלוי שפה, ותמיכה מוכנה בהמרה לפורמט JSON קיימת במגוון שפות תכנות, גם ב-Java.

השימוש העיקרי של הפורמט הוא להעברת אובייקטים בין שרת לצרכן כתחליף לפורמט XML הכבד והמסובך. הרעיון מאחורי פורמט JSON הוא שלוקחים אובייקט מתוכנית אחת הכתובה בשפה כלשהי, ממירים אותו למחרוזת טקסט פשוטה בפורמט מוכר, ואז שולחים אותו לתוכנית אחרת שיכולה להיות כתובה בשפה אחרת, וממירים את המחרוזת חזרה לאובייקט המקורי באופן המותאם לשפה הנוכחית. ממירים דווקא למחרוזת כדי כל שפה תדע להתמודד איתו, שהרי בכל שפה מטפלים במחרוזות.

## תחביר

קובץ JSON מתחיל ונגמר בסוגריים מסולסלות. לאחר כל פרמטר יהיה פסיק.

**טיפוסים פשוטים** - רושמים את שם המשתנה בגרשיים, לאחרים נקודותיים ':', ואז את ערך המשתנה: מספר, ערך בוליאני (true או false), מחרוזת (בגרשיים), או null.

**מערכים** - לאחר הנקודותיים פותחים סוגרים מרובעים, ובהם ערכי המערך שביניהם פסיק.

**אובייקטים** - לאחר הנקודותיים פותחים סוגריים מסולסלות, ומכניסים את שדות האובייקט – שם וערך כמו שתיארנו.

דוגמא לקובץ JSON המכיל אובייקט אדם, עם שדות טיפוסים פשוטים, אובייקט פנימי address, ומערך המכיל אובייקטים של מספרי טלפון:

{

"firstName": "John",

"isAlive": true,

"age": 27.5,

"address": {

"streetAddress": "21 2nd Street",

"city": "New York",

"state": "NY",

},

"phoneNumbers": [

{

"type": "home",

"number": "212 555-1234"

},

{

"type": "mobile",

"number": "123 456-7890"

}

],

}

## Json-simple

json-simple jar היא ספריית Java פשוטה המאפשרת לנתח, ליצור, להפוך ולעבד קבצי JSON. ספרייה זו מורכבת מ-13. המרכזיות שבהן הן:

1. **JSONParser** - נועדת לניתוח מחרוזת של JSON, הבנאי מקבל מחרוזות String או אובייקט זרימת תווים Reader.
2. **JSONObject** - אובייקט המייצג אובייקט שחולץ מ-JSON. הבנאי מקבלת מחרוזת JSON שמייצגת את האובייקט. קיימות בו שיטות המאפשרות לחלץ את כל המחרוזות JSON הקיימות באובייקט זה, לפי הטיפוס שלהם. בנוסף, מחלקה זו יורשת ממבנה הנתונים Map, ולכן על אובייקט זה ניתן להפעיל את כל השיטות שיש ב-Map.

יש לו גם שיטת writeJSONString(Map map, Writer out) המקבלת אובייקט Map ואובייקט זרימת תווים. באמצעות שיטה זו ניתן להמיר אובייקט Map למחרוזת JSON.

1. **JSONArray** - אובייקט המייצג מערך שחולץ מ-JSON. הבנאי מקבל מחרוזת JSON שמייצגת את המערך. קיימות בו שיטות המאפשרות לחלץ את כל המחרוזות JSON הקיימות באובייקט זה, לפי הטיפוס שלהם. בנוסף, מחלקה זו יורשת ממבנה הנתונים List, ולכן על אובייקט זה ניתן להפעיל את כל השיטות שיש ב-List.

יש לו גם שיטת writeJSONString(List list, Writer out) המקבלת אובייקט List ואובייקט זרימת תווים. באמצעות שיטה זו ניתן להמיר אובייקט List למחרוזת JSON.

1. **JSONValue** - נועדה להמרת מחרוזות JSON לאובייקטים של Java, וכן ההיפך.

JSONStringWrite(Object obj, Writer out) המשמשת ליצירת מחרוזת JSON מאובייקט Java.

Parse(Reader in) או parse(String s) מקבלות מחרוזת JSON ומחזירה אובייקט מסוג Object עליו ניתן לעשות casting. מומלץ להשתמש דווקא בשיטת parseWithException, העושה פעולה דומה רק שאינה מתעלמת מחריגות.

ה-API הבסיסי ביותר של המחלקות בקישור הבא:

<http://alex-public-doc.s3.amazonaws.com/json_simple-1.1/index.html>