כתיבת וקריאת עצמים

נניח שאנחנו רוצים לשמור עצם מורכב (עם שדות, עצמים נוספים, מערכים וכו') לקובץ, כדי שנוכל לקרוא אותו מאוחר יותר. או שאנחנו רוצים לשלוח עצם דרך האינטרנט לשרת שנמצא במחשב אחר. איך אפשר לעשות זאת? יש שתי דרכים עיקריות.

דרך א: כתיבה וקריאה כטקסט

הדרך הראשונה לכתיבה וקריאה של עצמים היא להמיר אותם לטקסט. ישנם שני פורמטים עיקריים לייצוג עצמים כטקסט: XML ו-JSON. הראשון ארוך יותר אבל מכיל מידע מדוייק יותר על המחלקות הנשמרות; השני קצר ותמציתי יותר אבל מכיל פחות מידע.

המרת אובייקטים ל-XML נתמכת בג'אבה באופן טבעי ע"י החבילה JAXB. כדי להשתמש בה צריך לעשות כמה שינויים במחלקות שאותן רוצים לשמור:

- צריך לסמן כל מחלקה כזאת בתג @XmlRootElement.
- צריך לוודא שלכל מחלקה יש בנאי בלי פרמטרים.
- צריך לוודא שלכל שדה של המחלקה שרוצים לשמור יש getter ו-setter.
- אם למחלקה יש מחלקות יורשות, צריך להזכיר אותן בתג @XmlSeeAlso במחלקה המורישה.

לפירוט ראו דוגמאות קוד בחבילה lesson10, בפרט במחלקה JaxbDemo.

בחבילה JAXB יש כמה בעיות. אחת מהן היא שאי-אפשר לשמור רשימות (כגון ArrayList) באופן ישיר כי המחלקה ArrayList לא כוללת את התג @XmlRootElement; צריך לעטוף רשימות במחלקות ייעודיות, ראו למשל Employees. בעיה שניה היא, שאם אותו עצם מופיע כמה פעמים במבנה שרוצים לשמור, העובדה הזאת לא נשמרת ב-XML. לכן, כשקוראים את העצמים מחדש מה-XML, העצם שמופיע כמה פעמים ישוכפל. אם יש הרבה עצמים כאלה - הדבר יגרום לבזבוז זיכרון משמעותי.

הפורמט השני נקרא JSON. הוא מקובל מאד באפליקציות רשת, לצורך תקשורת בין השרת ללקוח. ישנן כמה חבילות שמאפשרות לייצג עצמים כטקסט בפורמט זה, למשל GSON ו-Jackson. לדוגמת שימוש ב-GSON ראו GsonDemo. ב-GSON יש אף יותר בעיות מב-JAXB: הספריה לא מתמודדת כראוי עם מחלקות יורשות, הסוג של עצמים ברשימה כלל לא נשמר, וגם הבעיה של שיכפול מיותר קיימת כאן.

אני לא מכיר חבילה ששומרת עצמים כטקסט, בלי בעיית השיכפול המיותר.

דרך ב: כתיבה וקריאה בינרית

בשפת ג'אבה ישנו מנגנון משוכלל יותר לכתיבה וקריאה של עצמים. הוא נקרא serialization. במנגנון זה, העצמים לא נשמרים כטקסט אלא כרצף של בתים (בייצוג בינארי). החיסרון הוא, שהפורמט הזה אינו קריא לבני-אדם; היתרון הוא, שהעצמים נשמרים בצורה מדוייקת יותר ובלי שיכפול מיותר.

כדי להשתמש במנגנון זה, צריך להגדיר כל עצם שרוצים לשמור כמממש של הממשק Serializable. זה ממשק שנראה קצת מוזר - אין לו שום מתודות. הוא ריק לגמרי. הוא משמש רק לסימון.

חידה: איך אפשר לבדוק אם עצם כלשהו שייך למחלקה המממשת את הממשק Serializable?
תשובה: object **instanceof** Serializable.

אם מחלקה מסויימת מממשת את Serializable, אז גם כל צאצאיה מממשים אותו - לא צריך להגדיר זאת בפירוש. אבל, צריך לוודא שכל השדות שלה מממשים את Serializable, אחרת נקבל שגיאה בזמן ריצה.

אחרי שווידאנו שכל העצמים שנרצה לשמור הם Serializable, אנחנו צריכים להשיג **זרם-פלט** שלתוכו נשמור את העצמים. לשם כך צריך לבנות ObjectOutputStream. לשם כך צריך קודם לבנות OutputStream. יש כמה סוגים של OutputStream בשפת ג'אבה, למשל:

```
ByteArrayOutputStream baos = new ByteArrayOutputStream();
OutputStream fos = Files.newOutputStream(Paths.get("test.bin"));
```

אחרי שבנינו אחד כזה, אנחנו יכולים לבנות ObjectOutputStream , למשל:

```
ObjectOutputStream oos = new ObjectOutputStream(baos);
ObjectOutputStream oos = new ObjectOutputStream(fos);
```

ואז לשמור את העצמים שלנו לתוך הזרם בעזרת המתודה writeObject:

```
oos.writeObject(employee)
```

כדי לקרוא את העצמים שכתבנו, אנחנו צריכים **זרם-קלט**. לשם כך צריך לבנות ObjectInputStream, ולשם כך צריך קודם לבנות InputStream. יש כמה סוגים של InputStream בשפת ג'אבה, למשל:

```
ByteArrayInputStream bais = new ByteArrayInputStream(bytes);
InputStream fis = Files.newInputStream(Paths.get("test.bin"));
```

אחרי שבנינו אחד כזה, אנחנו יכולים לבנות ObjectInputStream , למשל:

```
ObjectInputStream ois = new ObjectInputStream(bais);
ObjectInputStream ois = new ObjectInputStream(fis);
```

ואז לקרוא את העצמים שלנו מתוך הזרם בעזרת המתודה readObject:

```
Employee employee = (Employee) ois.readObject();
```

שימו לב - העצם המוחזר הוא מסוג Object, ולכן צריך להשליך (cast) אותו למחלקה שאנחנו יודעים שכתבנו לתוכו.

השימוש ב-writeObject, readObject יותר יעיל ומהיר משימוש בפורמטי טקסט, הוא יודע לשמור מערכים ורשימות בלי שום בעיה, ויודע גם לא לשכפל עצמים - לדוגמה ראו בקובץ ObjectStreamDemo.

הערה: כברירת מחדל, writeObject ו-readObject כותבים וקוראים את כל השדות בעצם. אם רוצים ששדה מסויים לא ייכתב/ייקרא, יש לסמן אותו במילת-המפתח transient.

גירסאות של מחלקות

נניח שהגדרנו מחלקה, שמרנו עצם מהמחלקה בקובץ, שינינו את הגדרת המחלקה, וקראנו עצם מהמחלקה מתוך הקובץ. מה צריך לקרות? יש שתי אפשרויות:

- אפשרות אחת היא, שהשינוי שעשינו בהגדרת המחלקה הוא קטן ולא משפיע על הייצוג של המחלקה בקובץ. למשל, הוספנו מתודה, או הוספנו שדה חדש שיש לו ערך ברירת-מחדל הגיוני.

- אפשרות שניה היא, שהשינוי שעשינו בהגדרת המחלקה הוא משמעותי. למשל, הורדנו שדות והוספנו שדות שאין להם ערך ברירת-מחדל הגיוני.

במקרה הראשון, היינו רוצים לאפשר קריאה של המחלקה החדשה מקובץ שנכתב לפי ההגדרה הישנה; במקרה השני, היינו רוצים לא לאפשר קריאה אלא לזרוק חריגה. רק אנחנו - כותבי המחלקה - יודעים מהי האפשרות הנכונה.

הדרך שלנו להודיע לג'אבה מה האפשרות הנכונה היא ע"י הגדרת **גירסה**. לכל מחלקה המממשת Serializable, יש להגדיר קבוע סטטי בשם SerialVersionUID. ניתן לאתחל שדה זה ל-1, למשל כך:

```
private static final long serialVersionUID = 1L;
```

המתודה writeObject כותבת את הגירסה לקובץ, והמתודה readObject קוראת את הגירסה ומשווה לגירסה הנוכחית של המחלקה. אם הגירסה לא השתנתה, העצם נקרא והשדות החדשים מאותחלים לערכי ברירת-מחדל. אם הגירסה השתנתה (למשל הגירסה בקובץ היא 1 והגירסה הנוכחית בהגדרת המחלקה היא 2) אז readObject זורק חריגה מסוג InvalidClassException.

אם לא מגדירים את השדה ***serialVersionUID***, ג'אבה מגדירה אותו אוטומטית ע"י נוסחה כלשהי התלויה בשדות ובמתודות המוגדרים במחלקה. בדרך-כלל, כל שינוי קטן במחלקה יגרום לשינוי בתוצאת הנוסחה, ולכן לא נוכל לקרוא קבצים ישנים (כלומר התנהגות ברירת-המחדל היא לפי האפשרות השניה).

לכן, כשלא מגדירים את השדה הזה, אקליפס מזהיר אותנו ומודיע שכדאי להגדיר אותו.

מקורות

- Cay Horstmann, “Core Java SE 9 for the Impatient”, chapter 9.

סיכם: אראל סגל-הלוי.