

## חזרה בנושא אוספים ספציפיים ומבנים רקורסיביים

# קבצים ותיקיות Files & Folders

### מטרות

תרגול השימוש באוספים שנלמדו במהלך היחידה.

תרגול החשיבה הרקורסיבית.

### רמת השאלה

גבוהה.

### רקע

במערכת קבצים קיימים קבצים (files) ותיקיות (folders). לכל קובץ יש שם וגודל (מספר שלם שהוא מספר הבתים שהקובץ תופס בזיכרון). עקב גדלי הקבצים נשתמש בטיפוס **long**. לכל תיקיה יש שם ובתוכה מאוחסנים אוסף קבצים ותיקיות (תת-תיקיות).

### המחלקה File

File (String name, long size)	הפעולה בונה קובץ ששמו name וגודלו size
String getName()	הפעולה מחזירה את שם הקובץ
long getSize()	הפעולה מחזירה את גודל הקובץ בבתים.
String toString()	הפעולה מחזירה מחרוזת המתארת את הקובץ באופן הבא: <שם הקובץ> <גודל הקובץ>

### המחלקה Folder

Folder (String name)	הפעולה בונה תיקייה ריקה ששמה name
String getName()	הפעולה מחזירה את שם התיקייה
Folder createSubFolder (String name)	הפעולה יוצרת תת-תיקייה ומחזירה הפנייה אליה. הפעולה לא תאפשר יצירת תת-תיקייה אם כבר קיימת תת-תיקייה באותו שם.
void addFile (File file)	הפעולה מוסיפה את הקובץ file לתיקייה. הפעולה לא תאפשר הוספת קובץ אם כבר קיים קובץ אחר באותו שם.
File[] getFiles()	הפעולה מחזירה את אוסף הקבצים בתיקייה.
Folder[] getSubFolders()	הפעולה מחזירה את אוסף התת-תיקיות שנמצאות בתיקייה.

String toString()	<p>הפעולה מחזירה מחרוזת המתארת את התיקיה בעזרת:</p> <p>רשימת כל הקבצים ואחר כך רשימת התת-תיקיות:</p> <p>&lt;גודל קובץ 1&gt; &lt;שם קובץ 1&gt;</p> <p>&lt;גודל קובץ 2&gt; &lt;שם קובץ 2&gt;</p> <p>:</p> <p>&lt;שם תיקייה 1&gt;</p> <p>&lt;שם תיקייה 2&gt;</p>
-------------------	---

### מה עליכם לעשות?

א. כתבו את המחלקות File ו-Folder.

ב. כתבו תוכנית בדיקה היוצרת מספר תיקיות, תת-תיקיות וקבצים בתוך התיקיות.

ג. הוסיפו לתוכנית הבדיקה שכתבתם, את הפעולה הבאה:

<code>static long getFolderSize(Folder f)</code>	<p>הפעולה תחזיר את גודל התיקיה שהוא סכום כל גדלי הקבצים בתיקיה f ובתת-תיקיות שלה</p>
--	--

זמנו את הפעולה ובדקו שהיא מבצעת את הנדרש ממנה.

ד. הוסיפו למחלקה Folder את הפעולה הבאה:

String getRecursiveView()	<p>הפעולה תחזיר מחרוזת המתארת את התיקיה (בדומה ל-toString()), אלא שהפעם יש להציג את כל התת-תיקיות והקבצים שבתוכן</p>
---------------------------	--

ה. הוסיפו למחלקה Folder את הפעולה הבאה:

<code>boolean fileExists(String fileName)</code>	<p>הפעולה תחזיר true אם קובץ בשם fileName קיים בתיקיה הנוכחית או באחת מהתת-תיקיות שלה. אחרת, תחזיר false</p>
--	--

## הנחיות מיוחדות

- השימוש ב-long לא נפוץ ביחידה אך אין לו שום משמעות מיוחדת, פרט לכך שהוא מאפשר לשמור ערכים גדולים כפי שמתאים לתוכן הבעייה.
- הפעולות `addFile (...)`, `createSubFolder (...)`, חייבות לשמור על תקינות מבנה התיקיות והקבצים. לכן במימוש הפעולות הללו התלמיד נדרש לבצע בדיקות מתאימות כדי למנוע כפילויות. בכוונה נמנענו מלהוסיף הנחות לפעולות כדי לתרגל את התלמיד במימוש בדיקות שכאלה.
- פתרון שאלה זו מצריך לימוד של פרק רקורסיה.
- שאלה זו מיועדת לתלמידים טובים. הקושי בשאלה זו הוא המבנה הרקורסיבי שנוצר מהגדרת המחלקה Folder – תיקייה מכילה אוסף קבצים ואוסף של תת-תיקיות שהן תיקיות בעצמן.
- ייצוג התיקייה יהיה בעזרת אוסף דינמי של קבצים ואוסף דינמי של תת-תיקיות, מאחר ואין מגבלה על כמות הקבצים או התת-תיקיות שניתן לאחסן בתיקייה.
- כמו כן, הקבצים והתת-תיקיות הנמצאים בכל תיקיה הם בעלי שם ייחודי (אין שני קבצים או שתי תיקיות באותו שם). מומלץ להשתמש באוסף Map לאחסון הקבצים ותת-תיקיות.
- ניתן לייצג את המחלקה Folder באמצעות רשימה של תיקיות ורשימה נפרדת של קבצים. במקרה זה יהיה צורך לסרוק את הרשימות כדי למנוע כפילויות. פתרון נוח יותר יהיה באמצעות מפה.
- למי שלמדו את פרק מפה (Map), מומלץ לפתור את השאלה בעזרתה. מאחר וחלק מהסעיפים דורשים "סריקה" בתת-תיקיות, שדומה מאוד לסריקת עץ-בינרי, רצוי אולי לפתור שאלה זו לאחר לימוד הפרק עצים.

## א. המחלקות File ו-Folder:

```
public class File
{
    private String name;
    private long size;

    public File(String fileName, long fileSize)
    {
        this.name = fileName;
        this.size = fileSize;
    }

    public String getName()
    {
        return(this.name);
    }

    public long getSize()
    {
        return(this.size);
    }
}

public class Folder
{
    private String name;
    private Map<File> files;
    private Map<Folder> folders;

    public Folder(String dirName)
    {
        this.name = dirName;
        this.files = new Map<File>();
        this.folders = new Map<Folder>();
    }

    public File[] getAllFiles()
    {
        File[] allFiles = new File[this.files.getAllKeys().length];

        String[] files = this.files.getAllKeys();
        for (int i = 0; i < files.length; i++)
            allFiles[i] = this.files.getValue(files[i]);

        return(allFiles);
    }

    public Folder[] getAllSubFolders()
    {
        Folder[] allSubFolders = new Folder[this.folders.getAllKeys().length];

        String[] folders = this.folders.getAllKeys();
        for (int i = 0; i < folders.length; i++)
            allSubFolders[i] = this.folders.getValue(folders[i]);

        return(allSubFolders);
    }

    public void addFile(File file)
    {
        this.files.insert(file.getName(), file);
    }

    public Folder createEmptySubFolder(String folderName)
    {
        Folder folder = new Folder(folderName);
        this.folders.insert(folderName, folder);
    }
}
```

```
        return(folder);
    }

    public String getName()
    {
        return(this.name);
    }

    public String toString()
    {
        String str = "";
        String[] files = this.files.getAllKeys();
        String[] subFolders = this.folders.getAllKeys();

        str = this.name + " Content:\n";
        for(int i=0; i<subFolders.length; i++)
        {
            Folder folder = this.folders.getValue(subFolders[i]);
            str = str + folder.name + "\t\t" + "Folder" + "\n";
        }

        for(int i=0; i<files.length; i++)
        {
            File file = this.files.getValue(files[i]);
            str = str + file.getName() + "\t\t" + "File" + "\t" + file.getSize() + "\n";
        }
        return(str);
    }
}
```