10. Lekcja

Operacje na plikach

Co to są pliki ?

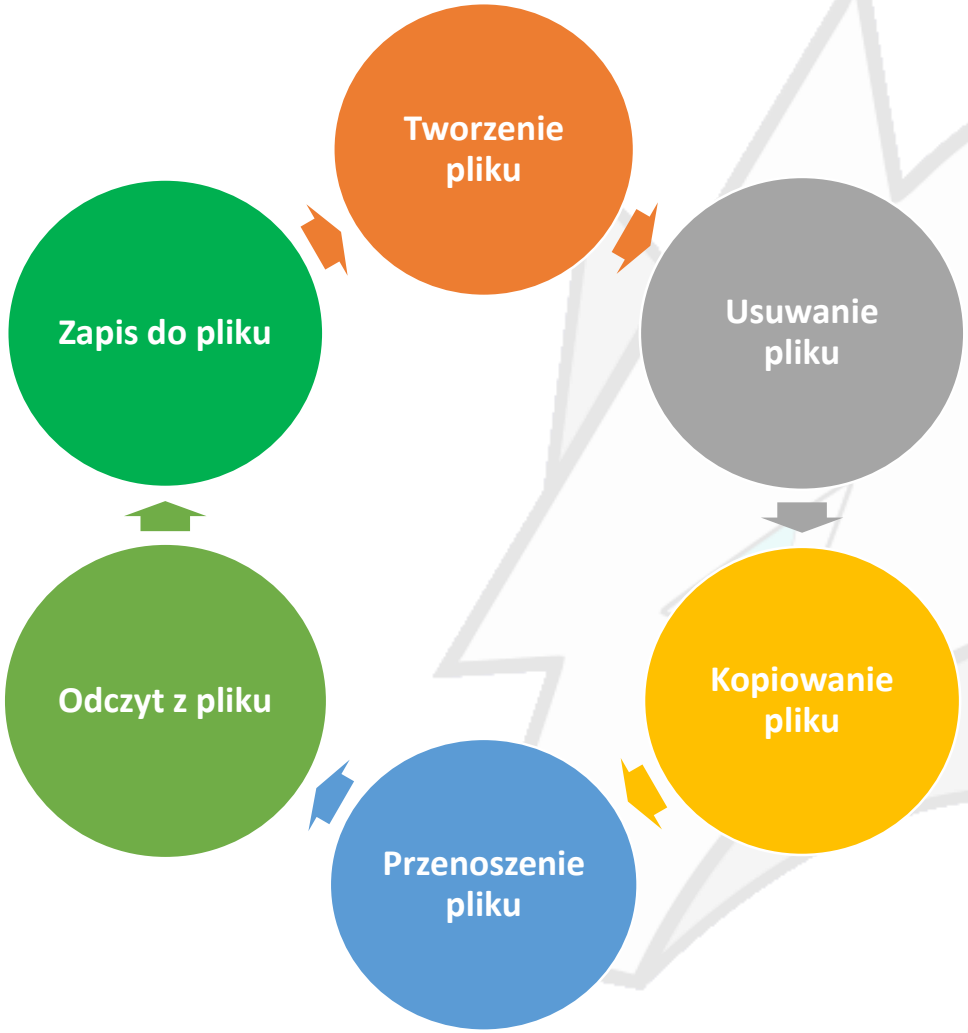
Plik danych, plik komputerowy, zwykle krótko plik – uporządkowany zbiór danych o skończonej długości, posiadający szereg atrybutów i stanowiący dla użytkownika systemu operacyjnego całość. Nazwa pliku nie jest jego częścią, lecz jest przechowywana w systemie plików.

Katalog/folder a plik tekstowy

Katalogi (ang. directory) (stosuje się też nazwy „foldery” lub „teczki”) – pliki zawierające spis odwołań do innych plików (w tym także do katalogów)

Plik tekstowy (ang. text file) – plik zawierający dane w postaci alfanumerycznej.

Operacje na plikach



Tworzenie pliku i katalogu

```
File newFile = new File( pathname: "test1.txt");
Path newFilePath;
Path folder = null;

try {
    // Create new file test1 first methode
    newFile.createNewFile();
    //Create new file test2 second methode
    newFilePath = Files.createFile(Paths.get( first: "test2.txt"));
    // Create new folder test in the same path
    folder = Files.createDirectories(Paths.get( first: "test"));

    System.out.println("File exist ? " + newFile.exists());
    System.out.println("File exist ? " + Files.exists(newFilePath));
    System.out.println("Folder exist ? " + folder.toFile().exists());

} catch (IOException e) {
    System.out.println("Can not create file or directory.");
}
```

- Do reprezentowania plików w Javie służy klasa File. Jako argument podaje się ścieżkę do pliku. Podanie samej nazwy pliku powoduje stworzenie pliku w miejscu wykonywania programu.
- Folder jest reprezentowany przez klase Path. Można go stworzyć przez statyczną metodę klasy Files.
- Metoda exists() służy do sprawdzania czy plik istnieje.
- W przypadku braku odstępu do odpowiedniej lokalizacji zostanie zwrócony wyjątek.

Usuwanie pliku i katalogu

- Usuwanie plików nie powoduje wyrzucania wyjątków.
- Usuwanie folderów powoduje wyrzucania wyjątków. Jeżeli w folderze będą znajdować się pliki folder nie zostanie uśnięty.
- W celu usunięcia wszystkich plików i folderów z danej ścieżki należy wykorzystać algorytm rekurencyjny.

```
newFile.deleteOnExit();
boolean isDeletedFile = newFile.delete();

try {
    Files.delete(folder);
    boolean isDeletedFolder = Files.deleteIfExists(folder);
} catch (IOException e) {
    System.out.println("Can not delete file or directory.");
}
```

```
public static void deleteFolderWithFiles(File file) {
    if (file.isDirectory()) {
        File[] files = file.listFiles();
        for (File f : files) {
            deleteFolderWithFiles(f);
        }
    }
    file.delete();
}
```

Przenoszenie i kopiowanie pliku

- W przypadku kopiowania plik pierwotny pozostaje w swojej lokalizacji
- W przypadku przenoszenia plik pierwotny zostaje usunięty.
- Za opcje kopiowania/przenoszenia odpowiada ostatni argument CopyOption. W naszym przypadku jeżeli plik w docelowym folderze istnie zostanie zastąpiony

```
try {
    Path folder = Files.createDirectories(Paths.get( first: "folder"));

    Files.exists(Path.of( first: "firstFile.txt"));
    Path firstFilePath = Files.createFile(Path.of( first: "firstFile.txt"));
    Path secondFilePath = Files.createFile(Path.of( first: "secondFile.txt"));

    Files.copy(firstFilePath, Path.of( first: folder.toAbsolutePath() + File.separator + "copiedFirstFile.txt"), StandardCopyOption.REPLACE_EXISTING);

    Files.move(secondFilePath, Path.of( first: folder.toAbsolutePath() + File.separator + "copiedSecondFile.txt"), StandardCopyOption.REPLACE_EXISTING);

} catch (IOException e) {
    System.out.println("Can not create, copy or move file");
}
```

Zapis do pliku

```
try {
    Path newFile =  Files.createFile(Path.of( first: "newFile.txt"));

    PrintWriter printWriter = new PrintWriter(newFile.toFile());
    printWriter.println("Text inside file");
    printWriter.close();

    printWriter.println("But after close the file you cant save new information");

    printWriter = new PrintWriter(newFile.toFile());
    printWriter.println("You can open it again...");
    printWriter.println("But only last changes will be save");
    printWriter.close();

} catch (IOException e) {
    System.out.println("Can not create file");
}
```

- Do zapisywania do pliku służy klasa PrintWriter. Do zainicjalizowania obiektu należy podać plik jako parametr aby go otworzyć do zapisu.
- Po zakończeniu korzystania z pliku należy go zamknąć.
- Po zamknięciu pliku nowe zmiany nie będą się pojawiać.
- Zawartość pliku zawsze będzie równa ostatnim zmianą pomiędzy otwarciem a zamknięciem pliku.

Odczyt z pliku

- Do odczytywania danych z pliku służy znana nam klasa Scanner. W tym wypadku jako argument konstruktora podajemy plik.
- Metoda hasNext() sprawdza czy istnieje nowa linia do odczytu. Połączenie z pętlą while() daje efekt wypisywania kolejnych linii aż do końca pliku.

```
try {
    Scanner scanner = new Scanner(new File( pathname: "newFile.txt"));

    while (scanner.hasNext()){
        System.out.println(scanner.nextLine());
    }

} catch (IOException e) {
    System.out.println("Can not create file");
}
```

Inne metody klasy File

- file.canRead() – zwraca informację czy plik da się odczytać
- file.canWrite() – zwraca informację czy do pliku da zapisywać
- file.canExecute() – zwraca informację czy plik wykonywalny
- file.getName() – zwraca tylko nazwę pliku
- file.getAbsolutePath() – zwraca ścieżkę bezwzględną do pliku
- file.getTotalSpace() – zwraca zajmowaną pojemność pliku
- file.isDirectory() – zwraca prawdę jeżeli plik jest katalogiem
- File.isFile() – zwraca prawdę jeżeli plik jest plikiem
- file.listFiles() – zwraca listę wszystkich plików w katalogu

Poćwiczmy!