JAVA 12 New Features

- Switch Expressions (Preview)

Bu değişiklik yerinde olmuş bence. Lambda ifadelerinde kullandığımız “arrow syntax” gelmiş ve basit bir kullanım olmuş. Aşağıdaki her iki koda uzaktan bakıldığında da okunabilirliğin arttığı görülüyor.

Exp: new switch expression (jdk12+)

```
/**
 * @author muludag on 25.03.2020
 */
public class main {

    public static void main(String args[]){
        System.out.println("JDK 12 New Switch Expression result:");
        System.out.println(executeNewSwitchExpression( day: "Monday"));
        System.out.println(executeNewSwitchExpression( day: "Sunday"));
    }

    public static String executeNewSwitchExpression(String day){
        return switch (day) {
            case "Monday", "Wednesday", "Friday" -> "MWF";
            case "Tuesday", "Saturday" -> "TS";
            default ->  "Please insert a valid day.";
        };
    }
}
```

Exp: old switch expr.(jdk12 -)

```
String day="M";
String result = "";
    switch (day) {
    case "M":
    case "W":
    case "F": {
        result = "MWF";
        break;
    }
    case "T":
    case "TH":
    case "S": {
        result = "TTS";
        break;
    }
};
```

Out:

```
JDK 12 New Switch Expression result:
MWF
Please insert a valid day.
```

Bu örnekte break kullanımı olmadığını farketmişsinizdir. Artık isteğe bağlı kullanılıyor. **->** ile return statement durumunda. Direk karşılığı atama ile yapılabiliyor. String data= switch(……)….. şeklinde. Şu şekilde de yazılabilirdi. -> operatöründen sonra **break “message”;** veya örnekte olduğu gibi **-> “message”;** bu iki kullanımda aynı işi yapıyor. Yani break yapısı kullanılsa da kullanılmasa da return statement şeklinde çalışıyor.

- File.mismatch method

File implementasyonunda yenilik var. Mismatch yapısı iki file objesini compare etmemizi sağlayacak. Örnek geçelim bir tane.

Exp:

```
import java.nio.file.Path;

/**
 * @author muludag on 25.03.2020
 */
public class main {
    public static void main(String args[]) throws IOException {
        Path filePath1 = Files.createTempFile( prefix: "file1",  suffix: ".txt");
        Path filePath2 = Files.createTempFile( prefix: "file2",  suffix: ".txt");
        Path filePath3 = Files.createTempFile( prefix: "file3",  suffix: ".txt");
        Files.writeString(filePath1, csq: "MelihHilmiUludag Test String");
        Files.writeString(filePath2, csq: "MelihHilmiUludag Test String");
        Files.writeString(filePath3, csq: "Mel1hHilmiUludag Test String");
        System.out.println("file1,file2 Mismatch position-> "+Files.mismatch(filePath1, filePath2));
        filePath2.toFile().deleteOnExit();
        System.out.println("file1,file3 Mismatch position-> "+Files.mismatch(filePath1, filePath3));
    }
```

out:

```
file1,file2 Mismatch position-> -1
file1,file3 Mismatch position-> 3
```

compare esnasında fark yok ise position bilgisi -1 döner.Fark var ise farkın olduğu position bilgisini döner. (index'leme 0 ile başlar dönüş tipi long)

- Compact Number Formatting

Günümüzde kullanılan yeni kompakt Numara stilini destekleyen kullanımlar getirildi.

Exp:

```
/**
 * @author muludag on 25.03.2020
 */
public class main {
    public static void main(String args[]) throws IOException {
        System.out.println("Compact Formatting is:");
        NumberFormat upvotes = NumberFormat
                .getCompactNumberInstance(new Locale( language: "en",  country: "US"), NumberFormat.Style.SHORT);
        upvotes.setMaximumFractionDigits(1);

        System.out.println(upvotes.format( number: 2592) + " upvotes");


        NumberFormat upvotes2 = NumberFormat
                .getCompactNumberInstance(new Locale( language: "en",  country: "US"), NumberFormat.Style.LONG);
        upvotes2.setMaximumFractionDigits(2);
        System.out.println(upvotes2.format( number: 2011) + " upvotes");
```

Out:

```
Compact Formatting is:
2.6K upvotes
2.01 thousand upvotes
```

uptoves değişkeninin setMaximumFractionDigits(2) şeklinde setleseydik 2.59K değeri üretecek ve vir üst değere yuvarlamayacaktı.

- Teeing Collectors

Stream yapısı içerisinde collect esnasında tüm input değerlerini yeni bir collect yapısı ile **bi-function**' a iletme yöntemi ile kullanılır.

Exp:

```
/**
 * @author muludag on 25.03.2020
 */
public class main {
    public static void main(String args[]) throws IOException {
        double meanData = Stream.of(1, 2, 3, 4, 5)
                .collect(Collectors.teeing(
                        Collectors.summingDouble(i -> i),
                        counting(),
                        (sumData, n) -> sumData / n));

        System.out.println("the output is: "+ meanData);

        System.out.println("Compact Formatting is:");
```

Out:

```
3.0
```

- New String Methods

indent(int n): String dizisine n değeri kadar whitespace yani Unicode boşluk/çıkıntı ekler.

transform(Function f): Sağlanan işlevi string türe uyarlar sonuç T yani generic olur.

```
String data="melih";
System.out.println("no indent: "+data);
System.out.println("indent data n 3 then:"+data.indent(3));
int lengthData =data.transform(s -> s.length());
```

out:

```
no indent: melih
indent data n 3 then:   melih
```

Melih Hilmi ULUDAĞ

Kaynaklar

https://openjdk.java.net/projects/jdk/12/
https://www.journaldev.com/28666/java-12-features
https://www.azul.com/39-new-features-and-apis-in-jdk-12/