Notatki - Formatting by Locale

Spis treści

Formatowanie numerów

Java dostarcza nam możliwość formatowania numerów na podstawie określonego `Locale`. Czyli w sumie nie musimy się zastanawiać jak formatować numery w określonym miejscu na świecie, gdzie ma być kropka, a gdzie przecinek - Java zrobi to za nas.

W tym celu możemy używać metod określonych w poniższej tabeli:

Metoda	Co robi
NumberFormat.getInstance()	Ogólny formatter
NumberFormat.getNumberInstance()	To samo co powyżej
NumberFormat.getPercentInstance()	Do formatowania liczb z procentami
NumberFormat.getCurrencyInstance()	Służy do formatowania pieniążków

Przykłady wykorzystania:

```
import java.text.NumberFormat;
import java.util.Locale;

public class FormattingNumbers {

    public static void main(String[] args) {
        int number = 1_234_567;
        Locale localePL = new Locale("pl", "PL");
        Locale localeUS = Locale.US;
        Locale localeGERMANY = Locale.GERMANY;

        System.out.println("NumberFormat.getInstance()");
        System.out.println("US: " + NumberFormat.getInstance(localeUS).format(number));
        System.out.println("PL: " + NumberFormat.getInstance(localePL).format(number));
        System.out.println("GERMANY: " + NumberFormat.getInstance(localeGERMANY).format(number));
        System.out.println();

        System.out.println("NumberFormat.getNumberInstance()");
        System.out.println("US: " + NumberFormat.getNumberInstance(localeUS).format(number));
        System.out.println("PL: " + NumberFormat.getNumberInstance(localePL).format(number));
        System.out.println("GERMANY: " + NumberFormat.getNumberInstance(localeGERMANY).format(number));
        System.out.println();

        System.out.println("NumberFormat.getCurrencyInstance()");
        System.out.println("US: " + NumberFormat.getCurrencyInstance(localeUS).format(number));
```

```
        System.out.println("PL: " + NumberFormat.getCurrencyInstance(localePL).format(number));
        System.out.println("GERMANY: " + NumberFormat.getCurrencyInstance(localeGERMANY).format(number));
        System.out.println();

        System.out.println("NumberFormat.getPercentInstance()");
        System.out.println("US: " + NumberFormat.getPercentInstance(localeUS).format(number));
        System.out.println("PL: " + NumberFormat.getPercentInstance(localePL).format(number));
        System.out.println("GERMANY: " + NumberFormat.getPercentInstance(localeGERMANY).format(number));
    }
}
```

Formatowanie dat

Skoro już mowa o formatowaniu, to Java pozwala też na różnorakie sposoby formatowania dat i czasów. Najczęściej spotykanym formatem będzie ustandaryzowany format `ISO`, który można narzucić poprzez `DateTimeFormatter`:

```
LocalDate date = LocalDate.of(2001, Month.DECEMBER, 15);
LocalTime time = LocalTime.of(16, 38, 52);
LocalDateTime dateTime = LocalDateTime.of(date, time);

System.out.println(date.format(DateTimeFormatter.ISO_LOCAL_DATE)); // 2001-12-15
System.out.println(time.format(DateTimeFormatter.ISO_LOCAL_TIME)); // 16:38:52
System.out.println(dateTime.format(DateTimeFormatter.ISO_LOCAL_DATE_TIME)); // 2001-12-15T16:38:52
```

Poniżej zostały umieszczone fragmenty kodu pokazujące kolejny możliwy sposób działania klasy `DateTimeFormatter`. Załóżmy, że w każdym z poniższych przykładów będziemy się odnosić do zmiennych określonych poniżej. Kolejne przykłady również będą się odnosiły do zmiennych `date`, `time`, `dateTime`, `offsetDateTime` oraz `zonedDateTime` zdefiniowanych poniżej.

```
LocalDate date = LocalDate.of(2001, Month.DECEMBER, 15);
LocalTime time = LocalTime.of(16, 38, 52);
LocalDateTime dateTime = LocalDateTime.of(date, time);
OffsetDateTime offsetDateTime = OffsetDateTime.of(dateTime, ZoneOffset.ofHours(3));
ZonedDateTime zonedDateTime = ZonedDateTime.of(dateTime, ZoneId.of("Poland"));
```

Przykłady wykorzystania `ISO_FORMAT`:

```
System.out.println(date.format(DateTimeFormatter.ISO_LOCAL_DATE)); ①
System.out.println(time.format(DateTimeFormatter.ISO_LOCAL_TIME)); ②
System.out.println(dateTime.format(DateTimeFormatter.ISO_LOCAL_DATE_TIME)); ③
System.out.println(offsetDateTime.format(DateTimeFormatter.ISO_LOCAL_DATE_TIME)); ④
System.out.println(zonedDateTime.format(DateTimeFormatter.ISO_LOCAL_DATE_TIME)); ⑤
```

① Zostanie wydrukowane: *2001-12-15*

② Zostanie wydrukowane: *16:38:52*

③ Zostanie wydrukowane: *2001-12-15T16:38:52*

④ Zostanie wydrukowane: *2001-12-15T16:38:52*

⑤ Zostanie wydrukowane: *2001-12-15T16:38:52*

Jeżeli chcemy podejść do tematu inaczej, mamy też możliwość wykorzystać poniższe kombinacje metod i parametrów:

DateTimeFormatter	FormatStyle
DateTimeFormatter.ofLocalizedDate()	FormatStyle.FULL FormatStyle.LONG FormatStyle.MEDIUM FormatStyle.SHORT
DateTimeFormatter.ofLocalizedTime()	FormatStyle.FULL FormatStyle.LONG FormatStyle.MEDIUM FormatStyle.SHORT
DateTimeFormatter.ofLocalizedDateTime()	FormatStyle.FULL FormatStyle.LONG FormatStyle.MEDIUM FormatStyle.SHORT

Poniżej przykłady wykorzystania `ofLocalizedDate()`, `ofLocalizedTime()` oraz `ofLocalizedDateTime()`. Na potrzeby poniższych fragmentów kodu, wprowadzamy również zmienne takie jak:

Definicje formatterów:

```
DateTimeFormatter dateFormatterFULL = DateTimeFormatter.ofLocalizedDate(FormatStyle.FULL);
DateTimeFormatter dateFormatterLONG = DateTimeFormatter.ofLocalizedDate(FormatStyle.LONG);
DateTimeFormatter dateFormatterMEDIUM = DateTimeFormatter.ofLocalizedDate(FormatStyle.MEDIUM);
DateTimeFormatter dateFormatterSHORT = DateTimeFormatter.ofLocalizedDate(FormatStyle.SHORT);

DateTimeFormatter timeFormatterFULL = DateTimeFormatter.ofLocalizedTime(FormatStyle.FULL);
DateTimeFormatter timeFormatterLONG = DateTimeFormatter.ofLocalizedTime(FormatStyle.LONG);
DateTimeFormatter timeFormatterMEDIUM = DateTimeFormatter.ofLocalizedTime(FormatStyle.MEDIUM);
DateTimeFormatter timeFormatterSHORT = DateTimeFormatter.ofLocalizedTime(FormatStyle.SHORT);

DateTimeFormatter dateTimeFormatterFULL = DateTimeFormatter.ofLocalizedDateTime(FormatStyle.FULL);
DateTimeFormatter dateTimeFormatterLONG = DateTimeFormatter.ofLocalizedDateTime(FormatStyle.LONG);
DateTimeFormatter dateTimeFormatterMEDIUM = DateTimeFormatter.ofLocalizedDateTime(FormatStyle.MEDIUM);
DateTimeFormatter dateTimeFormatterSHORT = DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT);
```

Przykłady wykorzystania metody `ofLocalizedDate()`:

```
System.out.println(date.format(dateFormatterFULL)); ①
System.out.println(date.format(dateFormatterLONG)); ②
System.out.println(date.format(dateFormatterMEDIUM)); ③
System.out.println(date.format(dateFormatterSHORT)); ④
```

① Zostanie wydrukowane: *sobota, 15 grudnia 2001*

② Zostanie wydrukowane: *15 grudnia 2001*

③ Zostanie wydrukowane: *15 gru 2001*

④ Zostanie wydrukowane: *15.12.2001*

Przykłady wykorzystania metody `ofLocalizedTime()`:

```
System.out.println(time.format(timeFormatterFULL)); ①
System.out.println(time.format(timeFormatterLONG)); ②
System.out.println(time.format(timeFormatterMEDIUM)); ③
System.out.println(time.format(timeFormatterSHORT)); ④
```

① Exception message: *Unable to extract ZoneId from temporal 16:38:52 with chronology ISO*

② Exception message: *Unable to extract ZoneId from temporal 16:38:52 with chronology ISO*

③ Zostanie wydrukowane: *16:38:52*

④ Zostanie wydrukowane: *16:38*

Przykłady wykorzystania metody `ofLocalizedDateTime()` z wykorzystaniem `LocalDateTime`:

```
System.out.println(dateTime.format(dateTimeFormatterFULL)); ①
System.out.println(dateTime.format(dateTimeFormatterLONG)); ②
System.out.println(dateTime.format(dateTimeFormatterMEDIUM)); ③
System.out.println(dateTime.format(dateTimeFormatterSHORT)); ④
```

① Exception message: *Unable to extract ZoneId from temporal 2001-12-15T16:38:52*

② Exception message: *Unable to extract ZoneId from temporal 2001-12-15T16:38:52*

③ Zostanie wydrukowane: *15 gru 2001, 16:38:52*

④ Zostanie wydrukowane: *15.12.2001, 16:38*

Przykłady wykorzystania metody `ofLocalizedDateTime()` z wykorzystaniem `OffsetDateTime`:

```
System.out.println(offsetDateTime.format(dateTimeFormatterFULL)); ①
System.out.println(offsetDateTime.format(dateTimeFormatterLONG)); ②
System.out.println(offsetDateTime.format(dateTimeFormatterMEDIUM)); ③
System.out.println(offsetDateTime.format(dateTimeFormatterSHORT)); ④
```

① Exception message: *Unable to extract ZoneId from temporal 2001-12-15T16:38:52+03:00*

② Exception message: *Unable to extract ZoneId from temporal 2001-12-15T16:38:52+03:00*

③ Zostanie wydrukowane: *15 gru 2001, 16:38:52*

④ Zostanie wydrukowane: *15.12.2001, 16:38*

Przykłady wykorzystania metody `ofLocalizedDateTime()` z wykorzystaniem `ZonedDateTime`:

```
System.out.println(zonedDateTime.format(dateTimeFormatterFULL)); ①
System.out.println(zonedDateTime.format(dateTimeFormatterLONG)); ②
System.out.println(zonedDateTime.format(dateTimeFormatterMEDIUM)); ③
System.out.println(zonedDateTime.format(dateTimeFormatterSHORT)); ④
```

① Zostanie wydrukowane: *sobota, 15 grudnia 2001 16:38:52 czas środkowoeuropejski standardowy*

② Zostanie wydrukowane: *15 grudnia 2001 16:38:52 CET*

③ Zostanie wydrukowane: *15 gru 2001, 16:38:52*

④ Zostanie wydrukowane: *15.12.2001, 16:38*

Należy tylko pamiętać, że `FormatStyle.FULL` oraz `FormatStyle.LONG` stara się pokazać również strefę czasową, zatem stosowanie ich do dat lub czasów bez podanych stref czasowych powoduje błąd podczas działania programu. Chyba, że mamy samą datę, wtedy działa to bez błędu.

No dobrze, ale nigdzie jeszcze nie zostało wspomniane o połączeniu `DateTimeFormatter` z `Locale`. Już podajemy przykład.

```
DateTimeFormatter.ofLocalizedDateTime(FormatStyle.MEDIUM).withLocale(Locale.US);
DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT).withLocale(Locale.GERMAN);
```

Jak widzisz, po określeniu `DateTimeFormatter` należy jeszcze dodać `.withLocale()`, inaczej będziemy używać pod spodem `Locale.getDefault()`.

Poniżej znajdziesz więcej przykładów:

Przykłady wykorzystania `Locale` dla `LocalDateTime`:

```
System.out.println("GRM: " + localDateTime.format(dateTimeFormatterFULL.withLocale(Locale.GERMAN))); ①
System.out.println("US: " + localDateTime.format(dateTimeFormatterFULL.withLocale(Locale.US))); ②

System.out.println("GRM: " + localDateTime.format(dateTimeFormatterLONG.withLocale(Locale.GERMAN))); ③
System.out.println("US: " + localDateTime.format(dateTimeFormatterLONG.withLocale(Locale.US))); ④

System.out.println("GRM: " + localDateTime.format(dateTimeFormatterMEDIUM.withLocale(Locale.GERMAN))); ⑤
System.out.println("US: " + localDateTime.format(dateTimeFormatterMEDIUM.withLocale(Locale.US))); ⑥

System.out.println("GRM: " + localDateTime.format(dateTimeFormatterSHORT.withLocale(Locale.GERMAN))); ⑦
System.out.println("US: " + localDateTime.format(dateTimeFormatterSHORT.withLocale(Locale.US))); ⑧
```

① Exception message: *Unable to extract ZoneId from temporal 2001-12-15T16:38:52*

② Exception message: *Unable to extract ZoneId from temporal 2001-12-15T16:38:52*

③ Exception message: *Unable to extract ZoneId from temporal 2001-12-15T16:38:52*

④ Exception message: *Unable to extract ZoneId from temporal 2001-12-15T16:38:52*

⑤ Zostanie wydrukowane: *GRM: 15.12.2001, 16:38:52*

⑥ Zostanie wydrukowane: *US: Dec 15, 2001, 4:38:52 PM*

⑦ Zostanie wydrukowane: *GRM: 15.12.01, 16:38*

⑧ Zostanie wydrukowane: *US: 12/15/01, 4:38 PM*

Przykłady wykorzystania `Locale` dla `ZonedDateTime`:

```
System.out.println("GRM: " + zonedDateTime.format(dateTimeFormatterFULL.withLocale(Locale.GERMAN))); ①
System.out.println("US: " + zonedDateTime.format(dateTimeFormatterFULL.withLocale(Locale.US))); ②

System.out.println("GRM: " + zonedDateTime.format(dateTimeFormatterLONG.withLocale(Locale.GERMAN))); ③
```

```
System.out.println("US: " + zonedDateTime.format(dateTimeFormatterLONG.withLocale(Locale.US))); ④

System.out.println("GRM: " + zonedDateTime.format(dateTimeFormatterMEDIUM.withLocale(Locale.GERMAN))); ⑤
System.out.println("US: " + zonedDateTime.format(dateTimeFormatterMEDIUM.withLocale(Locale.US))); ⑥

System.out.println("GRM: " + zonedDateTime.format(dateTimeFormatterSHORT.withLocale(Locale.GERMAN))); ⑦
System.out.println("US: " + zonedDateTime.format(dateTimeFormatterSHORT.withLocale(Locale.US))); ⑧
```

① Zostanie wydrukowane: *GRM: Samstag, 15. Dezember 2001 um 16:38:52 Mitteleuropäische Normalzeit*

② Zostanie wydrukowane: *US: Saturday, December 15, 2001 at 4:38:52 PM Central European Standard Time*

③ Zostanie wydrukowane: *GRM: 15. Dezember 2001 um 16:38:52 MEZ*

④ Zostanie wydrukowane: *US: December 15, 2001 at 4:38:52 PM CET*

⑤ Zostanie wydrukowane: *GRM: 15.12.2001, 16:38:52*

⑥ Zostanie wydrukowane: *US: Dec 15, 2001, 4:38:52 PM*

⑦ Zostanie wydrukowane: *GRM: 15.12.01, 16:38*

⑧ Zostanie wydrukowane: *US: 12/15/01, 4:38 PM*

Problemy

Jeżeli chodzi o metody typu `ofLocalizedDateTime()`, w praktyce może wydarzyć się coś takiego, że na 2 różnych komputerach będzie ustawione to samo `Locale`, a mimo to, 2 komputery drukują różne daty. W takim przypadku może to wynikać z różnic w systemach operacyjnych/wersjach Javy/vendorach Javy. Jeżeli taki problem będzie występował, można pobawić się metodą `.withLocale()`. Warto natomiast wiedzieć, że w praktyce takie rzeczy się zdarzają, pomimo, że dokumentacja określa jak powinno wyglądać poprawne zachowanie.