Dates and Times

Exercises and solutions

1. What class would you use to store a date without time and time zone parts in it?

**Answer:**

LocalDate

1. What class would you use to store date and time, which is aware of the Daylight Saving Time?

**Answer:**

ZonedDateTime

1. What is the difference between a ZoneId and ZoneOffset?

**Answer:**

A ZoneOffset represents a fixed zone offset from UTC time zone whereas a ZoneId represents a variable zone offset.

ZoneId = ZoneOffset + ZoneRules

1. What is the difference between a ZonedDateTime and an OffsetDateTime?

**Answer:**

An OffsetDateTime represents a datetime with a zone offset from UTC. A ZonedDateTime represents a datetime for a time zone with zone rules and is aware of Daylight Saving time.

1. Write the code to convert the Instant representing the current time in the system default time zone to a LocalDate.

**Solution:**

Instant now = Instant.now();

ZoneId zone = ZoneId.systemDefault();

LocalDate today = LocalDate.ofInstant(now, zone);

1. Write a program that prints all years from 2001 to 2099 in which the last day of the year (December 31) falls on Monday.

**Solution:**

// Test.java

package com.jdojo.test.exercises;

import java.time.DayOfWeek;

import java.time.LocalDate;

import java.time.Month;

public class Test {

public static void main(String[] args) {

for (int i = 2001; i <= 2099; i++) {

LocalDate dt = LocalDate.of(i, Month.DECEMBER, 31);

if (dt.getDayOfWeek() == DayOfWeek.MONDAY) {

System.out.println(i);

}

}

}

}

1. Write the code that converts a java.util.Date to a LocalDate in the system default time zone.

**Solution:**

Date dt = new Date();

LocalDate ld = LocalDate.ofInstant(dt.toInstant(), ZoneId.systemDefault());

1. Complete the following snippet of code, so it prints "Friday January 12, 1968". It is supposed to format the date 1968-01-12 and print it.  
     
   LocalDate bday = LocalDate.of(1968, Month.JANUARY, 12);  
   String pattern = /\* Your code goes here \*/;  
   DateTimeFormatter fmt = DateTimeFormatter.ofPattern(pattern);

String formattedBDay = fmt.format(bday);

System.out.println(formattedBDay);

**Solution:**

LocalDate bday = LocalDate.of(1968, Month.JANUARY, 12);

String pattern = **"EEEE MMMM DD, YYYY"**;

DateTimeFormatter fmt = DateTimeFormatter.ofPattern(pattern);

String formattedBDay = fmt.format(bday);

System.out.println(formattedBDay);

1. Complete the following snippet of code that prints the number of days between 1968-01-12 and 1969-09-19. It should print 616.  
     
   LocalDate ld1 = LocalDate.of(1968, Month.JANUARY, 12);  
   LocalDate ld2 = LocalDate.of(1969, Month.SEPTEMBER, 19);  
   long daysBetween = /\* Your code goes here \*/;  
   System.out.println(daysBetween);

**Solution:**

import java.time.LocalDate;

import java.time.Month;

import static java.time.temporal.ChronoUnit.DAYS;

...

LocalDate ld1 = LocalDate.of(1968, Month.JANUARY, 12);

LocalDate ld2 = LocalDate.of(1969, Month.SEPTEMBER, 19);

long daysBetween = **DAYS.between(ld1, ld2)**;

System.out.println(daysBetween);

1. Complete the code in the printFirstDayOfMonth() method. The method takes a LocalDate as an argument and prints the first day of the month in which the date occurs. Suppose the LocalDate passed in to this method is 2017-08-05, it will print "First day of AUGUST, 2017 is on SATURDAY".  
     
   public static void printFirstDayOfMonth(LocalDate ld) {  
    LocalDate newDate = ld.with(/\* Your Code goes here \*/);  
    System.out.printf("First day of %s, %d is on %s%n",   
    ld.getMonth(), ld.getYear(), newDate.getDayOfWeek​());  
   }

**Solution:**

LocalDate newDate = ld.with(TemporalAdjusters.firstDayOfMonth());

System.out.printf("First day of %s, %d is on %s%n",

ld.getMonth(), ld.getYear(), ld.getDayOfWeek​());