Practices for Lesson 12: Using the Date/Time API

Chapter 12

Practices for Lesson 12

Practices Overview

In these practices, you will work with the new date and time API.

Practice 12-1: Summary Level: Working with local dates and times

Overview

In this practice you work with `LocalDate`, `LocalTime`, and `LocalDateTime` objects to provide answers to the questions asked in the practice. Local objects have no concept of a time zone, so you can assume that all of the questions in the practice relate to the local time zone. Also, all of the dates utilize the ISO calendar.

Tasks

1. Open the `LocalDatesAndTimes12-01Prac` project in the `/home/oracle/labs/12-DateTime/practices/practice1` directory.
2. Open the Java class file, `LocalDatesAndTimes`, in the `com.example` package.
3. Read through the comments—these indicate what code you need to write to answer the questions provided.
   - Consult the lecture slides and the documentation if you get stuck.
   - When you have completed, the output from your class should look similar to the following output. (You do not need to format the print statements exactly the same way.)

```
Abe was 46 when he died.
Abe lived for 16863 days.

Bennedict was born in a leap year: true
Days in the year he was born: 366
Bennedict is 3 decades old.
It was a SATURDAY on his 21st birthday.

Planned Travel time:  340 minutes
Delayed arrival time: 20:44

The flight arrives in Miami: 2014-03-25T01:30
The delayed arrival time is: 2014-03-25T05:57

School starts: 2014-09-09
School ends:   2015-06-25
Number of school days: 183

The meeting time is: 2014-08-05T13:30
```

Practice 12-2: Detailed Level: Working with local dates and times

Overview

In this practice you work with `LocalDate`, `LocalTime` and `LocalDateTime` objects to provide answers to the questions asked in the practice. Local objects have no concept of a time zone, so you can assume that all of the questions in the practice relate to the local time zone. Also, all of the dates utilize the ISO calendar.

Tasks

1. Open the `LocalDatesAndTimes12-01Prac` project.
   a. Select File > Open Project.
   b. Browse to `/home/oracle/labs/12-DateTime/practices/practice1`.
   c. Select `LocalDatesAndTimes12-01Prac` and click Open Project.
2. Open the Java class file, `LocalDatesAndTimes`, in the `com.example` package.
3. Given the scenario:

   Abe Lincoln's Birthday: February 12, 1809, died April 15, 1855

   How old was he when he died?

   How many days did he live?

   To implement this scenario, add the following code to `LocalDatesAndTimes.java`

```
LocalDate abeBorn = LocalDate.of(1809, FEBRUARY, 12);
LocalDate abeDies = LocalDate.of(1855, APRIL, 15);
System.out.println("Abe was " + abeBorn.until(abeDies, YEARS) +
" when he died.");
System.out.println("Abe lived for " + abeBorn.until(abeDies,
DAYS) + " days.");
    System.out.println("");
```

4. Given the scenario:

   Bennedict Cumberbatch, July 19, 1976

   Born in a leap year?

   How many days in the year he was born?

   How many decades old is he?

   What was the day of the week on his 21st birthday?

To implement this scenario, add the following code to `LocalDatesAndTimes.java`

```
LocalDate bennedict = LocalDate.of(1976, JULY, 19);
    System.out.println("Bennedict was born in a leap year: " +
bennedict.isLeapYear());
    System.out.println("Days in the year he was born: " +
bennedict.lengthOfYear());
    LocalDate now = LocalDate.now();
    System.out.println("Bennedict is " + bennedict.until(now,
DECADES) + " decades old.");
```

```
    System.out.println("It was a " +
bennedict.plusYears(21).getDayOfWeek() + " on his 21st
birthday.");
    System.out.println("");
```

5. Given the scenario:
   Train departs Boston at 1:45PM and arrives New York 7:25PM
   How many minutes long is the train ride?
   If the train was delayed 1 hour 19 minutes, what is the actual arrival time?

To implement this scenario, add the following code to `LocalDatesAndTimes.java`

```
    LocalTime depart = LocalTime.of(13, 45);
    LocalTime arrive = LocalTime.of(19, 25);
    System.out.println("Planned Travel time:  " +
depart.until(arrive, MINUTES) + " minutes");
    System.out.println("Delayed arrival time: " +
arrive.plusHours(1).plusMinutes(19));
    System.out.println("");
```

6. Given the scenario:
   Flight: Boston to Miami, leaves March 24th 9:15PM. Flight time is 4 hours 15 minutes
   When does it arrive in Miami?
   When does it arrive if the flight is delays 4 hours 27 minutes?

To implement this scenario, add the following code to `LocalDatesAndTimes.java`

```
    LocalDateTime leaveBoston = LocalDateTime.of(2014, MARCH,
24, 21, 15);
    LocalDateTime arriveMiami =
leaveBoston.plusHours(4).plusMinutes(15);
    System.out.println("The flight arrives in Miami: " +
arriveMiami);
    System.out.println("The delayed arrival time is: " +
arriveMiami.plusHours(4).plusMinutes(27));
    System.out.println("");
```

7. Given the scenario:

School semester starts the second Tuesday of September of this year.

Hint: Look at the TemporalAdjusters class

What is the date?

School summer vacation starts June 25th

Assuming:

/* Two weeks off in December

 * Two other vacation weeks

* School is taught Monday - Friday

How many days of school are there?

Hint: keep track of the short weeks also

To implement this scenario, add the following code to `LocalDatesAndTimes.java`

```
int excludeWeeks = 5;
    LocalDate schoolStarts = LocalDate.of(2014, SEPTEMBER,
1).with(TemporalAdjusters.firstInMonth(TUESDAY)).with(TemporalAd
justers.next(TUESDAY));
    LocalDate endOfFirstWeek =
schoolStarts.with(TemporalAdjusters.next(FRIDAY));
    long firstWeekDays = schoolStarts.until(endOfFirstWeek,
DAYS) + 1;
    System.out.println("School starts: " + schoolStarts);
    LocalDate schoolEnds = LocalDate.of(2015, JUNE, 25);
    System.out.println("School ends:   " + schoolEnds);
    long lastWeeksDays = 0;
    if (schoolEnds.getDayOfWeek() != MONDAY) {
      LocalDate lastWeekStart =
schoolEnds.with(TemporalAdjusters.previous(MONDAY));
      lastWeeksDays = lastWeekStart.until(schoolEnds, DAYS) + 1;
      excludeWeeks++;
    }
    long days = ((schoolStarts.until(schoolEnds, WEEKS) -
excludeWeeks) * 5); // 7 days per week, weekdays are 5/7 of a
week.
    days = days + firstWeekDays + lastWeeksDays;
    System.out.println("Number of school days: " + days);
    System.out.println("");
```

8. Given the scenario:

   A meeting is scheduled for 1:30 PM next Tuesday. If today is Tuesday, assume it is today. What is the time of the week's meetings?

   To implement this scenario, add the following code to `LocalDatesAndTimes.java`

```
    LocalTime meetingTime = LocalTime.of(13, 30);
    LocalDate meetingDate =
LocalDate.now().with(TemporalAdjusters.nextOrSame(TUESDAY));
    LocalDateTime meeting = LocalDateTime.of(meetingDate,
meetingTime);
    System.out.println("The meeting time is: " + meeting);
    System.out.println("");
```

9. Run the project, the output should look similar to the following output.

```
Abe was 46 when he died.
Abe lived for 16863 days.

Bennedict was born in a leap year: true
Days in the year he was born: 366
Bennedict is 3 decades old.
It was a SATURDAY on his 21st birthday.

Planned Travel time:  340 minutes
Delayed arrival time: 20:44

The flight arrives in Miami: 2014-03-25T01:30
The delayed arrival time is: 2014-03-25T05:57

School starts: 2014-09-09
School ends:   2015-06-25
Number of school days: 183

The meeting time is: 2014-08-05T13:30
```

Practice 12-2: Summary Level: Working with dates and times across time zones

Overview

In this practice, you work with time zone classes to calculate dates and times across time zones.

Tasks

1. Open the NetBeans project `DepartArrive12-02Prac` in the `/home/oracle/labs/12-DateTime/practices/practice2` directory.
2. Open the class file, `DepartArrive.java` in the `com.example` package.
3. Read through the comments—these indicate what code you need to write to answer the questions provided.

- Consult the lecture slides and the documentation if you get stuck.
- When you are complete, the output from your class should look similar to this (note that you need not format the print statements exactly the same way).

```
Flight 123 departs SFO at:   2014-06-13T22:30-07:00[America/Los_Angeles]
Local time BOS at departure: 2014-06-14T01:30-04:00[America/New_York]
Flight time: 5 hours 30 minutes
Flight 123 arrives BOS:      2014-06-14T07:00-04:00[America/New_York]
Local time SFO at arrival:   2014-06-14T04:00-07:00[America/Los_Angeles]

Flight 456 leaves SFO at:    2014-06-28T22:30-07:00[America/Los_Angeles]
Local time BLR at departure: 2014-06-29T11:00+05:30[Asia/Calcutta]
Flight time: 22 hours
Flight 456 arrives BLR:      2014-06-30T09:00+05:30[Asia/Calcutta]
Local time SFO at arrival:   2014-06-29T20:30-07:00[America/Los_Angeles]

Flight 123 departs SFO at:   2014-11-01T22:30-07:00[America/Los_Angeles]
Local time BOS at departure: 2014-11-02T01:30-04:00[America/New_York]
Flight time: 5 hours 30 minutes
Flight 123 arrives BOS:      2014-11-02T06:00-05:00[America/New_York]
Local time SFO at arrival:   2014-11-02T03:00-08:00[America/Los_Angeles]
```

Practice 12-2: Detailed Level: Working with dates and times across time zones

Overview

In this practice, you work with time zone classes to calculate dates and times across time zones.

Tasks

1. Open the project `DepartArrive12-02Prac`.
   a. Select File > Open Project.
   b. Browse to `/home/oracle/labs/12-DateTime/practices/practice2`
   c. Select `DepartArrive12-02Prac` and click Open Project.
2. Open the `DepartArrive.java` file in the `com.example` package and make the following changes:
   a. Set the time zone for the three cities.

```
  ZoneId SFO = ZoneId.of("America/Los_Angeles");
  ZoneId BOS = ZoneId.of("America/New_York");
  ZoneId BLR = ZoneId.of("Asia/Calcutta");
```

   b. Given the scenario:

Flight 123, San Francisco to Boston, leaves SFO at 10:30 PM June 13, 2014
The flight is 5 hours 30 minutes
What is the local time in Boston when the flight takes off?
What is the local time at Boston Logan airport when the flight arrives?
What is the local time in San Francisco when the flight arrives?

Complete the following steps:

To compute the local time in Boston when the flight takes off, add the following code:

```
LocalDateTime departure = LocalDateTime.of(2014, JUNE, 13, 22,
30);
ZonedDateTime departSFO = ZonedDateTime.of(departure, SFO);
System.out.println("Flight 123 departs SFO at:   " + departSFO);
ZonedDateTime departTimeAtBOS =
departSFO.toOffsetDateTime().atZoneSameInstant(BOS);
System.out.println("Local time BOS at departure: " +
departTimeAtBOS);
System.out.println("Flight time: 5 hours 30 minutes");
```

   c. To compute local time at Boston Logan airport when the flight arrives, add the following code:

```
ZonedDateTime arriveBOS =
departSFO.plusHours(5).plusMinutes(30).toOffsetDateTime().atZone
SameInstant(BOS);
    System.out.println("Flight 123 arrives BOS:      " +
arriveBOS);
```

d. To compute the local time in San Francisco when the flight arrives, add the following code:

```
ZonedDateTime arriveTimeAtSFO =
arriveBOS.toOffsetDateTime().atZoneSameInstant(SFO);
    System.out.println("Local time SFO at arrival:   " +
arriveTimeAtSFO);
    System.out.println("");
```

3. Given the scenario:

- Flight 456, San Francisco to Bangalore, India, leaves SFO at Saturday, 10:30 PM June 28, 2014
- The flight time is 22 hours
- Will the traveler make a meeting in Bangalore Monday at 9 AM local time?
- Can the traveler call her husband at a reasonable time?

Modify `DepartArrive.java`.

a. Compute the local departure time at SFO.

```
departure = LocalDateTime.of(2014, JUNE, 28, 22, 30);
    departSFO = ZonedDateTime.of(departure, SFO);
    System.out.println("Flight 456 leaves SFO at:    " +
departSFO);
```

b. Compute the local departure time at Bangalore.

```
ZonedDateTime departTimeAtBLR =
departSFO.toOffsetDateTime().atZoneSameInstant(BLR);
    System.out.println("Local time BLR at departure: " +
departTimeAtBLR);
    System.out.println("Flight time: 22 hours");
```

c. Compute the local arrival time at Bangalore.

```
ZonedDateTime arriveBLR =
departSFO.plusHours(22).toOffsetDateTime().atZoneSameInstant(BLR
);
System.out.println("Flight 456 arrives BLR:      " + arriveBLR);
```

d. Compute the local arrival time at SFO.

```
arriveTimeAtSFO =
arriveBLR.toOffsetDateTime().atZoneSameInstant(SFO);
    System.out.println("Local time SFO at arrival:   " +
arriveTimeAtSFO);
    System.out.println("");
```

4. Given the scenario:

Flight 123, San Francisco to Boston, leaves SFO at 10:30 PM Saturday, November 1st, 2014
Flight time is 5 hours 30 minutes.
What day and time does the flight arrive in Boston?
What happened?

Modify `DepartArrive.java`.

a. Compute the local departure time at SFO.

```
departure = LocalDateTime.of(2014, NOVEMBER, 1, 22, 30);
    departSFO = ZonedDateTime.of(departure, SFO);
    System.out.println("Flight 123 departs SFO at:   " +
departSFO);
```

b. Compute the local departure time at Boston.

```
departTimeAtBOS =
departSFO.toOffsetDateTime().atZoneSameInstant(BOS);
    System.out.println("Local time BOS at departure: " +
departTimeAtBOS);
    System.out.println("Flight time: 5 hours 30 minutes");
```

c. Compute the local arrival time at SFO with the delay of 5 hours.

```
  arriveBOS =
departSFO.plusHours(5).plusMinutes(30).toOffsetDateTime().atZone
SameInstant(BOS);
    System.out.println("Flight 123 arrives BOS:      " +
arriveBOS);
    arriveTimeAtSFO =
arriveBOS.toOffsetDateTime().atZoneSameInstant(SFO);
    System.out.println("Local time SFO at arrival:   " +
arriveTimeAtSFO);
    System.out.println("");
```

5. Run the project, the output could be similar to this:

```
Flight 123 departs SFO at:   2014-06-13T22:30-07:00[America/Los_Angeles]
Local time BOS at departure: 2014-06-14T01:30-04:00[America/New_York]
Flight time: 5 hours 30 minutes
Flight 123 arrives BOS:      2014-06-14T07:00-04:00[America/New_York]
Local time SFO at arrival:   2014-06-14T04:00-07:00[America/Los_Angeles]

Flight 456 leaves SFO at:    2014-06-28T22:30-07:00[America/Los_Angeles]
Local time BLR at departure: 2014-06-29T11:00+05:30[Asia/Calcutta]
Flight time: 22 hours
Flight 456 arrives BLR:      2014-06-30T09:00+05:30[Asia/Calcutta]
Local time SFO at arrival:   2014-06-29T20:30-07:00[America/Los_Angeles]

Flight 123 departs SFO at:   2014-11-01T22:30-07:00[America/Los_Angeles]
Local time BOS at departure: 2014-11-02T01:30-04:00[America/New_York]
Flight time: 5 hours 30 minutes
Flight 123 arrives BOS:      2014-11-02T06:00-05:00[America/New_York]
Local time SFO at arrival:   2014-11-02T03:00-08:00[America/Los_Angeles]
```

Practice 12-3: Summary Level: Formatting Dates

Overview

In this practice, you work with the `DateTimeFormatter`.

Tasks

1. Open the project `TimeBetween12-03Prac` in NetBeans from the `/home/oracle/labs/12-DateTime/practices/practice3` directory.
2. Open the class, `TimeBetween.java`.
3. Modify the class to read a string from the console and correctly identify the delta between today and the entered date in years, months, and days.
   - Use the appropriate method to ensure that the values for the year, month and days are always positive.
4. The output should look similar to this:

```
Enter a date: (MMMM d, yyyy): July 9, 2014
Date entered: July 9, 2014
There are 0 years, 4 months, 16 days between now and the date
entered.
```

Practice 12-3: Detailed Level : Formatting Dates

Overview

In this practice, you work with the `DateTimeFormatter`.

Tasks

1. Open the project `TimeBetween12-03Prac` in NetBeans.
    a. Select File > Open Project.
    b. Browse to `/home/oracle/labs/12-DateTime/practices/practice3`
    c. Select `TimeBetween12-03Prac` and click Open Project.
2. Open the class, `TimeBetween.java`.
3. Modify the class to read a string from the console and correctly identify the delta between today and the entered date in years, months, and days.
    - Use the appropriate method to ensure that the values for the year, month and days are always positive.

a. Declare two variables.

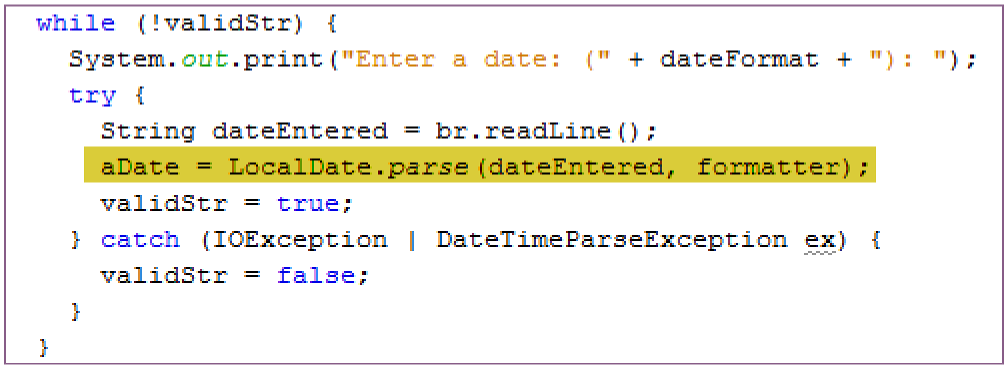
```
String dateFormat = "MMMM d, yyyy";
    LocalDate aDate = null;
```

b. Create a formatter to accept date entries using the USA common standard(month day, year).

```
  DateTimeFormatter formatter =
DateTimeFormatter.ofPattern(dateFormat);
```

c. Use the parse method with the formatter to create a date. Add the following statement within the `try` block.

```
            aDate = LocalDate.parse(dateEntered, formatter);
```



```
while (!validStr) {
  System.out.print("Enter a date: (" + dateFormat + "): ");
  try {
    String dateEntered = br.readLine();
    aDate = LocalDate.parse(dateEntered, formatter);
    validStr = true;
  } catch (IOException | DateTimeParseException ex) {
    validStr = false;
  }
}
```

d. To calculate the years, months, and days between now and the date entered, enter the following code:

```
     Period between;
     if (aDate.isBefore(now)) {
       between = Period.between(aDate, now);
     } else {
       between = Period.between(now, aDate);
     }
```

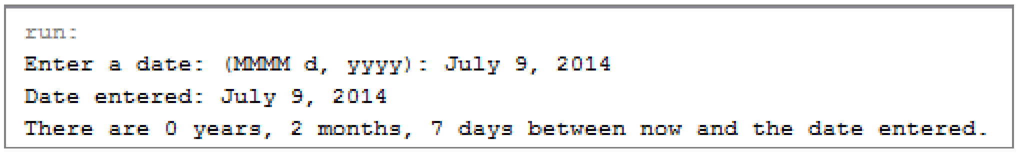
e. Obtain the value of day, month and year and assign it to the variables: `days`, `months`, and `years`.

```
    int years = between.getYears();
     int months = between.getMonths();
     int days = between.getDays();
```

f. Print the values of the years, months, and days.

```
System.out.println("There are " + years + " years, "
            + months + " months, "
            + days + " days between now and the date entered.");
```

4. Run the project, the output could be similar to this:



```
run:
Enter a date: (MMMM d, yyyy): July 9, 2014
Date entered: July 9, 2014
There are 0 years, 2 months, 7 days between now and the date entered.
```