Java Date and Time API

The Java Date and Time API provides a rich set of classes for handling dates, times, durations, and time zones. Prior to Java 8, developers relied on `java.util.Date` and `java.util.Calendar`, which had design flaws and were not thread-safe. To overcome these limitations, Java 8 introduced a new package `java.time`, which provides a modern, immutable, and thread-safe date-time API inspired by the Joda-Time library.

Packages for Date & Time API:

1. java.util (Old)
   - java.util.Date
   - java.util.Calendar
   - java.util.TimeZone

2. java.sql
   - java.sql.Date
   - java.sql.Time
   - java.sql.Timestamp

3. java.text
   - java.text.SimpleDateFormat
   - java.text.DateFormat

4. java.time (New - Introduced in Java SE 8)
   - java.time.LocalDate
   - java.time.LocalTime
   - java.time.LocalDateTime
   - java.time.ZoneId
   - java.time.Clock
   - java.time.Year
   - java.time.YearMonth
   - java.time.Period
   - java.time.Duration
   - java.time.Instant
   - java.time.MonthDay

Example 1: Working with LocalDate, LocalTime, and LocalDateTime

```
import java.time.LocalDate;
import java.time.LocalTime;
import java.time.LocalDateTime;

public class DateTimeExample {
    public static void main(String[] args) {
        LocalDate date = LocalDate.now();
        LocalTime time = LocalTime.now();
        LocalDateTime dateTime = LocalDateTime.now();

        System.out.println("Current Date: " + date);
        System.out.println("Current Time: " + time);
        System.out.println("Current Date and Time: " + dateTime);
    }
}
```

Example 2: Working with ZoneId

```
import java.time.ZonedDateTime;
import java.time.ZoneId;

public class ZoneExample {
    public static void main(String[] args) {
        ZonedDateTime zdt = ZonedDateTime.now(ZoneId.of("Asia/Kolkata"));
        System.out.println("Current Date and Time in India: " + zdt);
    }
}
```

Example 3: Working with Period and Duration

```
import java.time.LocalDate;
import java.time.LocalTime;
import java.time.Period;
import java.time.Duration;

public class PeriodDurationExample {
    public static void main(String[] args) {
        LocalDate startDate = LocalDate.of(2020, 1, 1);
        LocalDate endDate = LocalDate.now();
        Period period = Period.between(startDate, endDate);
        System.out.println("Years: " + period.getYears() + ", Months: " + period.getMonths() + ", Days: " + period.getDay

        LocalTime startTime = LocalTime.of(9, 0);
        LocalTime endTime = LocalTime.now();
        Duration duration = Duration.between(startTime, endTime);
        System.out.println("Duration in hours: " + duration.toHours());
    }
}
```

Summary: - The old date-time classes (java.util.Date, java.util.Calendar) are mutable and not thread-safe. - The new java.time API is immutable, thread-safe, and provides better control over date and time. - It supports ISO-8601 standards and has strong integration with time zones and durations.