Topic 4: Guidelines for Class Design

Part 1: Design Choices Ch3.1-3.3

Class Design Alternatives

Some Background Knowledge

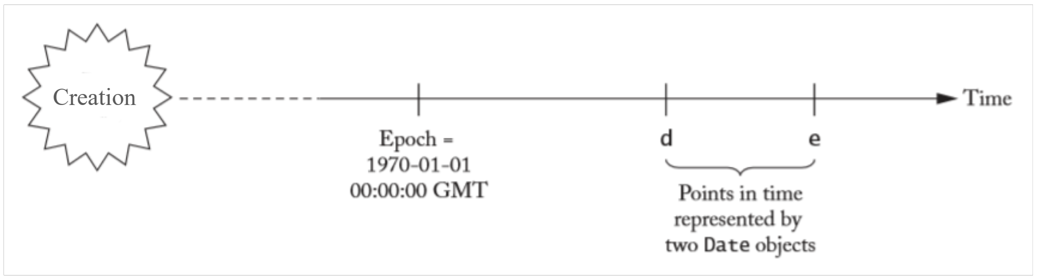
Date Class

- Java provides a Date class (in the java.util package) that uses a long integer to represent the number of milliseconds that has passed since the epoch: Jan 1, 1970, 00:00:00 GMT
- When constructing a new Date object, it will take the current time of execution (Dating.java)
- Some notable methods:
  - `boolean after(Date when)`
  - `boolean before(Date when)`
  - `int compareTo(Date anotherDate)`
  - `long getTime()`
  - `long setTime(long time)`

Some Background Knowledge

Date Class

- What's confusing about the Date class is that:
  - It's named Date, but also represents time.
  - Its implementation only considers one type of usage, considering the total *ordering nature* of Dates
    - It does not implement extraction of months, days, and years.


Creation
Time
Epoch = 1970-01-01 00:00:00 GMT
d
e
Points in time represented by two Date objects

Side note: Total Ordering

- A **total ordering** is a relation between two objects in a set (or universe) that fulfils the following properties:
  - Transitivity
  - Reflexivity
  - Antisymmetric
  - Totality
- A total order indicates that any two objects can be compared. In Java, a total order for a class is typically implemented through the `Comparator` interface
- A **natural order** on the other hand is a default comparison between objects in a class and is typically defined using the `Comparable` interface

Day Class: Background

- From 45 BC to 1582 AD, the Julian calendar was the standard calendar used in most of the world
- On October 15, 1582, Pope Gregory XIII ordered a new standard calendar called the Gregorian calendar.
  - The essential difference is that in the Julian calendar, every four years was counted as a leap year whereas the Gregorian calendar excluded years divisible by 100 unless divisible by 400
  - i.e. 1900 is not a leap year but 2000 was
- A Julian day number is the number of days from January 1, 4713 BC.
  - The date May 23 1968 has the Julian day number 2,440,000
- Java provides a `GregorianCalendar` class that provides methods for points in time.

Day Class

- Class Responsibilities
  - Able to work with a calendar day. For example: Days, Months, Years (Not time, no time-zones...).
- Public Interface

```
public class Day {
   public Day(int year, int month, int day);
   public int getYear();
   public int getMonth();
   public int getDate();
   public Day addDays(int n);
   public int daysFrom(Day other);
}
```

(DayTester.java)

Side note: Deprecated

- **Deprecated** - Parts of a public interface that are not longer supported or recommended
  - Usually means the deprecated part was not a good idea and has been redesigned.
- Date has many deprecated functions
  Ex: `getMonth()` should be avoided.
- Use Calendar class instead.
- Use built in Java classes when possible (here use built in Calendar class instead of our Day class).

Day Class: a comparison

Implementation 1 – DayOne.java

- This class' state is represented by the year, month, and day attributes
- It uses helper methods `nextDay()` and `previousDay()` and calls them repeatedly to find the differences in days.
- Helper methods are typically left private because:
  - They can complicate class interface
  - They may have special preconditions that may be checked within your class
  - They aid in a very specific implementation aspect of the class

Day Class: a comparison

Implementation 2 – DayTwo.java

- This class' state is represented by the Julian day number attribute
- The methods `addDays()` and `daysFrom()` are now trivial, but `getMonth(), getDay(), and getYear()` requires a conversion from the Julian day number.
  - Requires conversions between Julian day number and YMD
  - Consider

```
System.out.printf("%d-%d-%d",
d.getYear(), d.getMonth(), d.getDate());
```

Day Class: a comparison

Implementation 3 – DayThree.java

- This class keeps both the YMD and the Julian day number representation and convert between them only when needed
  - If created via the day number, calculate year only when needed.
  - If created via YMD, calculate the day# when needed.

- Functions check data validity:
  - If valid, then use it.
  - If invalid, calculate it & save answer.

Day Class Design Summary

- Implementations:
- DayOne: Work on year, month, day.
- DayTwo: Work on a day's number (Julian day).
- DayThree: Lazy conversion between both.
- Which is best?
- Working with:
  - Year/Month/Day:
  - Julian days (addDays(),...):
  - Efficiency:
  - Simplest code: