3 Understanding Data: Classes, Containment, Unions

In this lab we will focus on understanding data definitions, the distinction between information and data, how information can be represented as data, how to interpret the information that some instance of data represents, and learn to encode the data definitions, as well as construct instances of data in a class based language (like Java).

Do as many of these exercises as you need to feel comfortable with the material.

3.1 Data Definitions in HtDP

1. Consider the following problem. We are given information about a radio show: name, total running time in minutes, and a list of ads to run during the show, where for each ad we were given its name, the running time in minutes and the profit it generates in dollars.

   Here are the data definitions as we have learned them in HtDP:

```
;; Data definitions

;; A Radio Show (Show) is
;;      (make-show String Number [Listof Ad])
(define-struct show (name minutes ads))

;; An Ad is (make-ad String Number Number)
(define-struct ad (name minutes profit))

;; Examples of data:

(define ipod-ad (make-ad "ipod" 2 100))
(define ms-ad (make-ad "ms" 1 500))
(define xbox-ad (make-ad "xbox" 2 300))

(define news-ads
  (list ipod-ad ms-ad ipod-ad xbox-ad))
(define game-ads
  (list ipod-ad ms-ad ipod-ad ms-ad xbox-ad ipod-ad))
(define bad-ads
  (list ipod-ad ms-ad ms-ad ipod-ad xbox-ad ipod-ad))

(define news (make-show "news" 60 news-ads))
(define game (make-show "game" 120 game-ads))
```

2. Identify the examples that represent the following information:

   - an iPod ad that lasts 2 minutes and brings in $100 profit.
   - an MS ad that lasts 1 minute and brings in $500 profit.
   - an XBox ad that lasts 2 minutes and brings in $300 profit.
   - a news show that runs the ads for iPod, MS, iPod again, and XBox and lasts 60 minutes.
   - a game show that lasts 120 minutes and runs the ads for iPod, MS, iPod and MS again, then XBox.

Additional Example

3. Each of the following pieces of information represents a file in a computer directory for your *iPhoto* collection. Design the Scheme data definitions necessary to represent the information related to *picture files* and convert the information into Scheme data.

   - Picture of a river (jpeg) that is 3456 pixels wide and 2304 pixels high, using up 3,614,571 bytes.
   - Picture of a mountain (jpeg) that is 2448 pixels wide and 3264 pixels high, using up 1,276,114 bytes.
   - Picture of a group of people (gif) that is 545 pixels wide and 641 pixels high, using up 13,760 bytes.
   - Picture of a plt icon (bmp) that is 16 pixels wide and 16 pixels high, using up 1334 bytes.

3.2 Data Definitions in ProfessorJ

Open a new Tab and set the language to *Beginner ProfessorJ*.

1. Design the Java class to represent an ad in a radio show. Name the class `Ad`. Notice that you are just translating all the information given in the data definition for *Beginner Student* HtDP language into the *Beginner ProfessorJ* language.

2. Create a class `Examples` at the end of your `Definitions` as follows:

```
class Examples {
  Examples() {}
```

```
    // examples of ad data
    Ad ipodAd = ...
    Ad msAd = ...
    Ad xboxAd = ...
  }
```

You will be defining all your sample data in the `Examples` class. Later, it will also be a place where you put all your tests.

Include in the `Examples` class definitions of all data that you have defined in the *Beginner Student* HtDP language.

3. Once you are done, run the program. You should see an instance of the `Examples` class in the *Interactions* window.

Additional Example

3. Design the Java class to represent an iPhoto picture. Name the class `Photo`. Create a class `Examples` at the end of your *Definitions* as follows:

```
  class Examples {
    Examples() {}

    // examples of photo data
    Photo p1 = ...
    Photo p2 = ...
  }
```

4. Again, once you are done, run the program. You should see an instance of the `Examples` class in the *Interactions* window.

Notice: You can only define one `Examples` class in any *ProfessorJ* file. Use one `Examples` class to define all data examples for all classes defined above. It is also possible to use names such as `ExamplesPhoto` or `ExamplesAd` for defining the examples of data in the corresponding classes. We prefer to define all related data examples in one `Examples` class.

3.3 Class Diagrams

```
;; An Ad is (make-ad String Number Number)
(define-struct ad (name minutes profit))
```

All information that describes the class `Ad` can be concisely represented by a *class diagram* as follows:

```
+-------------+
| Ad          |
+-------------+
| String name |
| int minutes |
| int profit  |
+-------------+
```

Draw a class diagram for the class `Photo`.

3.4 Data With Containment

1. Start by looking at the classes in the textbook that represent the the clock time. The `ClockTime` class is defined on page 21.

   Open the file *lab3-2.java* and in the `Examples` class make examples of the `ClockTime` data as follows:

   - the time is 8:15 in the morning
   - the time is 4:35 in the afternoon
   - Make two examples of `ClockTime` data and ask your partner to tell what time do these examples represent.

2. Now we would like the class `Show` to represents a radio show in a manner similar to the earlier Scheme definition. However, at this point we will ignore the ads, but will keep track of both the start time for the show and the end time.

   Work out the data definition and draw the class diagram for the class `Show`.

3. Make examples of two radio shows: a news show that starts at 7:00 pm and runs for 30 minutes, and a game show that starts at 10:00 am and runs for an hour.

   Note: When defining the minutes, you cannot us a number `00`. Try it and see what error message you will get.

4. Combine the class diagrams for the classes `Show` and `ClockTime` using the *containment* arrows where appropriate.

Additional Example

6. The photo file information also includes the time of day when the photo file was created. (One typically records also the date when the photo was taken, but adding the date would just increase the complexity of our program.) Design a new `Photo` class that includes the information about the time of day when the photo was taken.

7. Draw the class diagram for the `Photo` class and the `ClockTime` class with the appropriate *containment* arrows.

8. Make sure to include examples of the instances of the new `Photo` class in your `Examples` class.

9. (Optional) Write down the data definitions for these classes of data in the *Beginner HtDP* language as well.

3.5 Unions of Data

The FCC requires that the radio runs not only commercially profitable ads, but also public service ads. Public service ads do not generate any profit, but the station records the code for the ad (a `String`) as well as the duration of the ad.

1. Adjust the *Beginner HtDP* data definition for `Ad` so that we can now represent two variants of ads: *Commercial* and *Service*. Make examples of data as well.

2. Now convert the data definitions to Java - as two classes that *implement* a common `interface Ad`.

3. Draw a class diagram for these data definitions.

4. *Remember* — defining examples of data is a part of the data definition.

Additional Example

You now have a new camera that allows you to take not only still pictures, but also short videos. You want to keep the two kinds of files together. Here is a data definition for the data you may want - written in the *Beginner HtDP* language:

```
;; Data to represent a camera shot:
;; either a still photo or a video clip

;; A Shot is one of
;; -- Photo
;; -- Video

;; we omit the definition of the Photo
;; as you already have done that

;; A Video is (make-video String String Num Num Boolean)
(define-struct video (name kind size duration sound?))
;; Interpretation:
;;    kind of video may be either QuickTime or RealPlayer
;;    size is measured in bytes
;;    duration is measured in seconds
;;    sound? indicated whether or not this video has sound
```

2. Define the Java classes that correspond to the given data definition by sketching the class diagram on a paper. Also, make examples of data - by hand.
3. Add the field that represents the time when the video was created to the class Video. Convert the data definitions to class diagrams and Java code.
4. **Do we have to remind you to make examples of data???**

3.6 Lists of Data

The profit for a radio show is determined by computing sum of the profits for all ads aired during the show. So, we need to modify the definition of the show by including a list of all ads aired during the show.

Let us first think of how we can represent a list of ads. Recall the data definition for a list of ads from the *Beginner HtDP* language, then work out the following problems:

1. Represent these data definitions as a class diagram on a paper.
2. Observe the data definition for the class `ListofAds`. It defines a union. What happens to the class of data that represents an empty

list of ads? How many examples can you make of data that belongs to this class of data? What are the names of the two fields for the `ConsLoAds` class? What are their types?

3. You can now convert this data definition into Java data definitions. Do so.

4. **Do we have to remind you to make examples of data???**

Additional Example

We continue with the the theme of the photo images. Our collections of images surely contains more than one picture. We will first define a list of images.

Once we have the data that represents our pictures, we would like to be able to ask questions about the data. To do so, we design methods in the classes that represent our data.

5. The HtDP data definition for a list of photos is:

   ```
   ;; A ListOfPhotos is one of
   ;; --- empty
   ;; --- (cons Photo ListOfPhotos)
   ```

   Design the Java classes to represent a list of `Photo`s. Remember to make examples of data.

   Note: This is the last time we will remind you to make examples of data.