Java By Abstraction - Companion Notes
Topic 2 -Programming By Delegation

CSE 1720, Winter 2010, Version 1.0, Prepared by: M. Baljko

LEGEND

JD = Java Details
PT = Programming Tip
IMD = In More Detail
Ex = Example
wrt = with respect to

Section 2.1; JD 2.1; PT 2.1; IMD 2.1-2.3; Ex 2.1-2.4

- This section is really key –do you recognize the following?
  - that the procedural paradigm was used in the `makeSunset` function discussed in CSE1710 (e.g., delegation of computation to `makeRed`, `makeBlue`)
  - that the object-oriented paradigm was used for `file` in the `makeHomePage` function. In that example, we don't know the type of the variable `file`, but we know it is an object since we are invoking methods on it.
    - side note: Java requires the declaration of all variables in advance of their use, whereas Python/Jython does not. This is why we don't know the type of `file`.
- If you would like to see the UML diagrams for the classes in `type.lib`, have a look at p. 464
- keep in mind the following: you are shown the modular paradigm in order to illustrate a concept about delegation, but the modular paradigm has a big flaw, which means it is not actually used that much in practice (or at least should not be used!). What is the big flaw? The explanation is in IMD 2.6 (but it depends on the concept of contracts, which is not introduced until sec 2.3.3
- We can discuss the method invocation `System.out.println()` in more detail once you have read this section;
  - `System.out` is a way to access the attribute of the utility class `System` that is named `out`. The class `System` represents the computer upon which the virtual machine is running (generally speaking). The attributes `in` and `out` represent the input and output streams, respectively. I/O streams are discussed in more detail later, in sec 8.1.5 and 9.2.
  - For now, you just need to know that the value of `System.out` is an instance of a `PrintStream` object. Since `PrintStream` is not among the 8 primitive data types (`char`, `short`, `byte`, `int`, `long`, `float`, `double`, `boolean`), `PrintStream` must be a nonprimitive data type. Thus, the value of this attribute is a *nonprimitive value*. The attribute `Rectangle2.width` is an `int`, which is a primitive type. (IMD 2.2 is relevant to this point)
  - How does one know what is *type* of an attribute? The API specifies this.
  - What operations can get performed on a `PrintStream` object? Or, in other words, what services does the `PrintStream` class provide? One of these is `println()`, which causes a carriage return (blank line) to be printed to the console.

- In sum, the class `PrintStream` is a nonutility class, whereas the class `System` is a utility class. The invocation `System.out.println()` is a combination of both types of classes.

Section 2.2-2.3; JD 2.2; PT 2.2; IMD 2.5-2.6; Ex 2.5

- Sec 2.2.1 and 2.2.2 are important; encapsulation is a bread-and-butter concept.
- JD 2.3 is somewhat abstract – keep in mind that the term "interface" in the statement "a public feature is place in the interface" is referring to the interface between the client and implemented (see fig. 2.8). This "interface" is an abstract concept. What the interface actually looks likes can be seen the API, which is typically viewed using a web browser (e.g, figs 3.2-3.8 in the chapter 3 provide graphical approximations and Ex 2.5 provides a text-based approximation)
- Sec 2.2.3 is important. The section presumes that you already know that division by zero causes a problem to the virtual machine (so now you know).

Section 2.4; PT 2.3-2.4

- You can bypass sec 2.4 and PT 2.3 for now and come back to it later
- PT 2.4 makes an important point

Section 2.2.5; JD 2.4

- These are the building blocks for interactive applications
- JD 2.4 is a historical note; you can ignore since we will exclusively be using Java 5.0 (also sometimes called Java 1.5, not that this is terribly important)

Section 2.3; JD 2.5; Ex 2.6-2.8

- read the lead-in to sec 2.3 and then 2.3.1
- set aside sec 2.3.2 and JD 2.5for now
- sec 2.3.3 is TERRIBLY TERRIBLY important!!!! The illustrations in Ex 2.6-2.8 are very clear and on-point. Consider this the hugest hint possible for your term test preparation!

Case Study 2.3.4 "Meet the Managers"

- Don't read this (yet)
  - this case study is tailored to the declaration and assignment of a variable with a primitive value. To make sense of this, one must have some understanding about the digital representation of values in the various representation schemes used by the primitive data types.
  - Since we have skipped over the relevant prior sections in Chapter 1, some of the background information need to understand this case study is missing.
- Stay tuned – an alternative version of "Meet the Managers" will be posted on the course wiki shortly. We will cover the alternative version instead.