**CIS110**

**Class Work – Abstraction & Class Design Objects**

Summary:

In class, we implemented Pet.java from the first table. Our process was to first consider what information about the entity would be relevant to the stakeholder. Then, we formalized how these properties could be written in Java (a name should usually be a String, for example). These properties, along with their types, represent the fields of the classes that we’ll write to *abstract* these entities. Once you have the properties defined, try to fill in a few rows of the table with instances of the entity. Ask yourself: are these properties all relevant to the stakeholder? Is some information missing?

Once you have a few rows of the table filled out, you can translate it into a Java class very quickly. The name of the entity can be the name of the file & class. The properties and their types become the fields of the class, and it’s easy to write a basic constructor for a class once you know its fields. Once you have these fields defined, you can start to write getters, setters, and other methods that will become how your object behaves. This is the process of abstraction: transforming the relevant properties of some physical or real thing into the class that will represent that thing in Java code.

1. We completed this in class using some suggestions about real pets from students.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Entity: Pet / Stakeholder: Person looking to adopt | | | | |
| Name (String) | Species (String) | Color (String) | isAlive (boolean) | age (int) |
| Izzy | Cat | Tuxedo | true | 11 |
| Chamomile | Cat | Brown | false | 1 |
| Zeus | Dog | Gray | true | 8 |
|  |  |  |  |  |

1. For this one, I’ll give you a few example instances. You should try to fill out the properties of a restaurant, with types, from there. I got you started on the right-hand side.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Entity: Restaurant / Stakeholder: Customer | | | | |
|  |  |  |  | isVegetarian (boolean) |
| “Koreana” | “Korean” | 4.8 | “$” | false |
| “Halal Guys” | “Middle Eastern” | 3.3 | “$$” | false |
| “Hip City Veg” | “Vegan” | 4.1 | “$$” | true |
| “Fogo de Chao” | “Steakhouse” | 2.8 | “$$$$” | false |

1. Here you’re on your own! What properties of a flight are relevant to a person looking to travel? Think about what information has been relevant to you when travelling in the past.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Entity: Flight / Stakeholder: Traveler | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Flight in this case means a scheduled airline service.

1. This example, paired with the next one, illustrates how the representation of an entity should depend on the stakeholder for whom it’s designed. Contrast how a Doctor would use and view a Person object compared to how Facebook would use a Person object.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Entity: Person / Stakeholder: Doctor | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Entity: Person / Stakeholder: Facebook | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |