### Notes about OOP

*(Kaminski)*

#### Instance Variables
- All variable names (including instance variables) should always begin with a small letter, like: `title, dateAdded`
- They should be declared as `private`, which means that only other members (methods, constructor, properties) of their same class CAN access them.
- However, good OOP practice is to NOT have any methods or constructor access instance variables directly. Instead, methods and constructors should access the variables’ PROPERTIES.
- Only a property should access its corresponding instance variable.

#### Properties
- Property names should always use the SAME NAME as their corresponding instance variable, except that the name begins with a capital letter, like: `Title, DateAdded`
- They should be declared as `public`, so any member of their own class OR another class can access them.
- Properties are:
  - **USED** *(i.e., referred to in some method or constructor or another property)* as if they were just regular VARIABLES.
  - **But they are DESCRIBED** *(i.e., the code inside the `get` and `set`)* as if they were METHODS – i.e., they DO ACTIONS *(e.g., calculate, read/write, loop, make decisions)*. Where as instance variables just STORE DATA.
- A **set accessor PROTECTS** its instance variable by ensuring that only valid data is actually put into its instance variable. It may also TRANSFORM what the caller provides into a form suited to the instance variable’s storage type.
- A **get accessor TRANSLATES** the stored data *(in the instance variable)* into a format better suited to the caller’s needs.
- Examples of transforming stored data into the caller’s required format:
  - Data stored as metric, caller needs feet/inches/… or gallons/quarts/…
  - Time stored as number of seconds past midnight, caller needs 12-hour clock with hours and minutes with am/pm.

#### Data Types of Instance Variables and their Properties
- An instance variable and its corresponding property will usually have the same data type.
- On occasion they may not, so the property’s `get` and `set` code transforms the stored data value in the instance variable to and from the caller’s preferred data type – e.g., data stored as numeric, caller needs a string.

#### Using instance variables vs. properties
- Methods and constructors *(and special properties which have no corresponding instance variable)* use properties as if they are regular variables. They should NEVER access the instance variables directly (even though they legally could). That is, they use `Title` and `Artist`, not `title` and `artist`.
- The properties in turn access the instance variables, e.g., `title` and `artist`.

#### Constructors
- Constructor names must be the same as its class name. Hence the name should always begin with a capital letter.
- They are declared as `public` since they are called when an object is declared *(e.g., in Main)*.
- Because constructors typically initialize the instance variables via their properties based on input parameter values, the parameter names typically correspond to the instance variable names with the addition of a `par` or `p` prefix – e.g., `parTitle` and `parArtist`.
- A constructor is just like a method, except that it:
  1) must have the same name as the class’s name, e.g., `Song`
  2) it doesn’t have a data type *(like `void` or `int` or…*)
  3) it therefore doesn’t have a `return`