FUN FACTS about USING a user-defined class

last modified: 2022-10-02

- In each function using a class you have defined, don't forget to `#include` the `.h` file for the class you are using!

  And include the `.cpp` file for the class in the `g++` command compiling/linking/loading a program using that class.

- Once you declare a class, you can make a (static) array of elements of that class using the syntax you might have expected:

  ```
  int quantities[10];           // an array able to hold 10 int values

  double measures[10];          // an array able to hold 10 double values

  PlayerChar participants[10]; // an array able to hold 10 PlayerChar objects
  ```

- And you can set an array element -- or a plain local variable, for that matter -- to contain an object instance by assigning to it an appropriate call to its constructor. But these look different than the calls when you are declaring an object!

  - That is -- consider these working declarations, from `PlayerChar-test.cpp`:

    ```
    PlayerChar sven;

    PlayerChar angie("Angie", 10, 2.7, "tank", 15);
    ```

  - Now consider these working declarations and assignments (that I tested before posting this handout):

    ```
    PlayerChar team[3];

    team[0] = PlayerChar();

    team[1] = PlayerChar("Angie", 10, 2.7, "tank", 15);

    team[2] = PlayerChar("Sven", 5, 1.35, "creampuff", 2);
    ```

- **NOTE:** `cout`'s << operator does NOT know how to output an object of your card class!

  But, it does know how to output a `string`! (or an `int` or `double` or `bool`)

  - So, using our `PlayerChar` class as an example, while the following **WILL NOT WORK:**

    ```
    PlayerChar angie("Angie", 10, 2.7, "tank", 15);

    cout << angie << endl;     // WARNING, DOES NOT WORK!!!!!
    ```

  - The following **WILL work:**

    ```
    PlayerChar angie("Angie", 10, 2.7, "tank", 15);

    cout << angie.player_to_string() << endl;

    angie.display_player();

    cout << angie.get_name() << " " << angie.get_strength() << endl;
    ```