TECHNICAL DUCUMENTATION
Object-Oriented Programming (LAB/LEC)
GORDON COLLEGE(CCS)
Bachelor of Science in Information Technology (BSIT)

Course & Block: BSIT_2D **Date: Dec 9, 2025**

Project Title: CYBER SERPENT

I. Encapsulation

Encapsulation keeps data safe by preventing direct access and allowing controlled interaction through methods.

How it is implemented:

- **Private Property (Data Hiding):**
  - The **$file** property inside the **Storage** class is declared **as private string $file;.**
  - **Use:** This prevents any code outside the **Storage** class from directly changing the path of the **scores.json** file, thereby keeping the data location safe and controlled.
- **Controlled Internal Access (Read/Write):**
  - The core file operations (reading and writing the JSON data) are contained in the **protected** methods **read()** and **write()**.
  - **Use**: These methods are only accessible to the Storage class itself and its children (like **Leaderboard**). This ensures that the sensitive file input/output logic is never executed by external code, protecting the file's integrity.
- **Public Interface (Score Management):**
  - All interactions are channeled through the public methods **getTopScores(), addScore()**, and **resetScores().**
  - Use: This provides a clean, controlled interface where the class enforces its own rules. For example, the **addScore()** method automatically:
    - Sanitizes the player name using **htmlspecialchars().**
    - Sorts the entire list of scores.
    - Limits the list to the top 10 scores (**array_slice**) before writing to the file. This process ensures the data is always stored in a consistent and secure format.

II. Abstraction

Abstraction hides complicated processes and only exposes simple methods to use.

How it is implemented:

- **Simple Public Methods:**
  - Code outside the class only uses simple commands:
    - **$leaderboard->addScore($name, $score)**
    - **$leaderboard->getTopScores()**
    - **$leaderboard->resetScores()** **Internal Logic Hidden:**
  - The complex operations happen inside the class: ▪ Reading the **scores.json** file
    - Handling JSON decoding and fallbacks
    - Sorting the scores using **usort()**
    - Capping the leaderboard to 10 entries using **array_slice()** ▪ Writing the updated data back to the file

III. Inheritance

Inheritance allows one class to reuse the functions of another.

How it is implemented:

- **Storage (Base Class):**
  - Contains shared functions for file-handling: **__construct()**, **read()**, **write()**, and the general **getTopScores()**.
- **Leaderboard Class Extends Storage:**
  - The class **Leaderboard extends Storage**.
  - Use: It inherits all the essential file-handling functions (**read()**, **write()**) and adds score-specific logic, such as sorting, sanitizing, and capping the list in its custom **addScore()** method.

IV. Polymorphism

Polymorphism allows a method name to behave differently depending on the class that uses it.

How it is implemented:

- **Overridden addScore() Method:**

- The base **Storage** class defines an empty **addScore()** method (**public function addScore(string $name, int $score){}**).
- The derived **Leaderboard** class provides its own, specific implementation for **addScore()** that contains all the ranking, sorting, and sanitization logic.
- Use: Although both classes have a method named **addScore(),** the version called is determined by the object used **($leaderboard = new Leaderboard(...)**), executing the specialized ranking logic from the **Leaderboard** class.