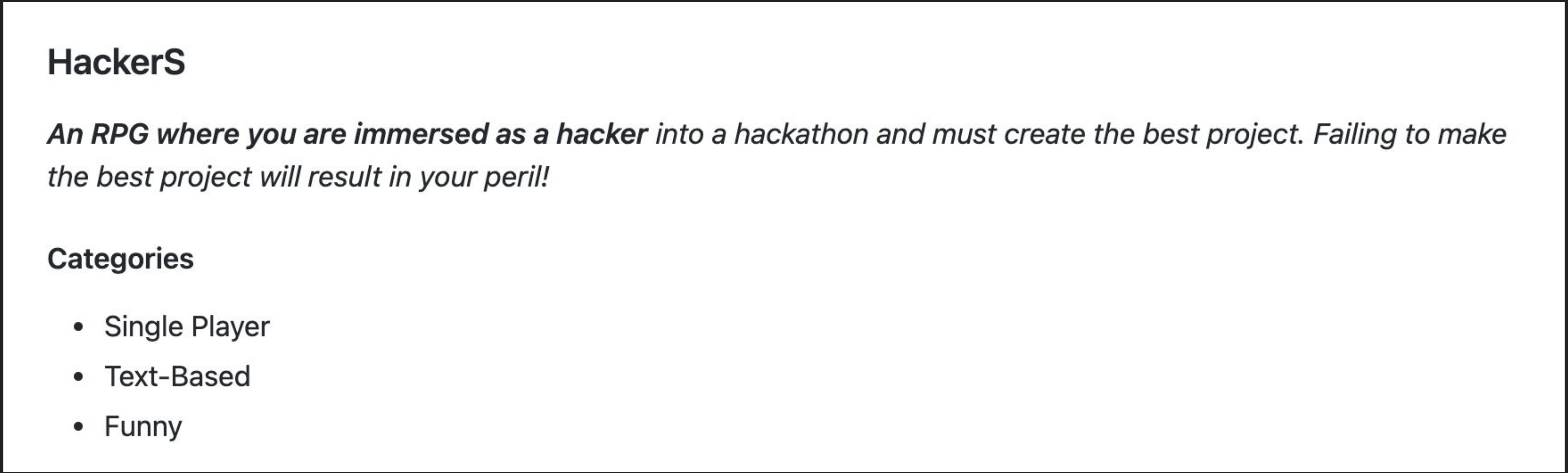
ISU Hackathon 2021

stdafx

A Hackathon IN THE HACKATHON??



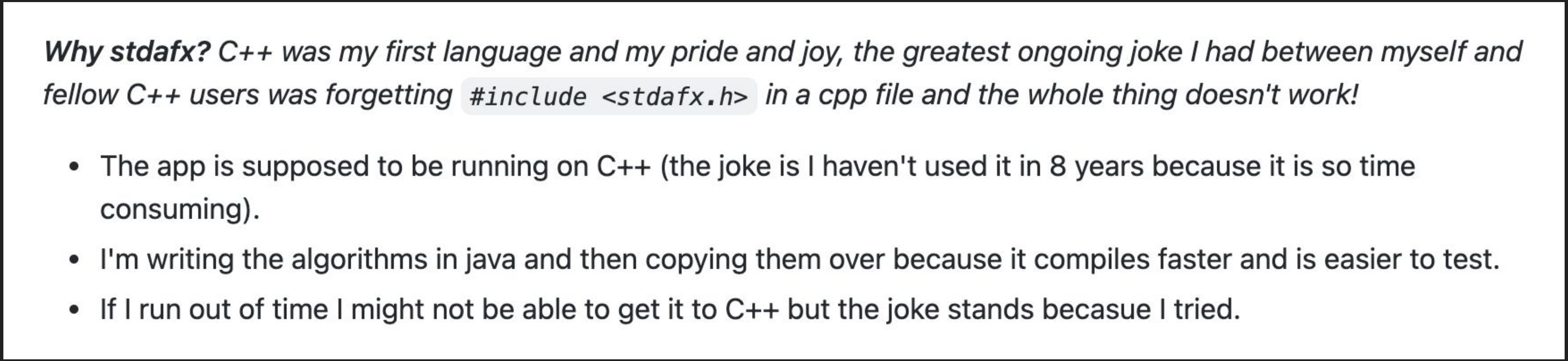
HackerS

***An RPG where you are immersed as a hacker** into a hackathon and must create the best project. Failing to make the best project will result in your peril!*

Categories

- Single Player
- Text-Based
- Funny

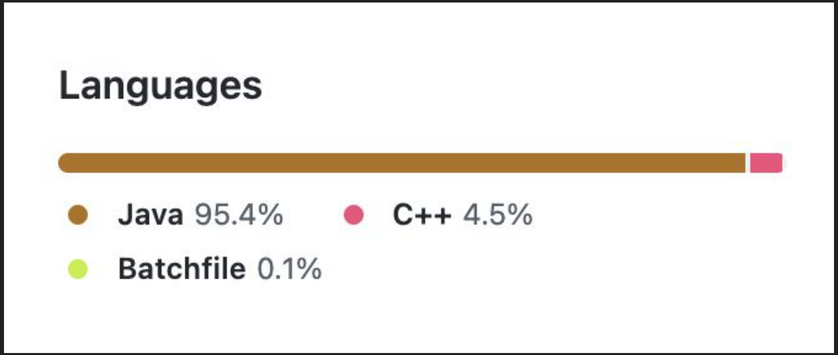
Background - STDAFX



Why stdafx? *C++ was my first language and my pride and joy, the greatest ongoing joke I had between myself and fellow C++ users was forgetting* `#include <stdafx.h>` *in a cpp file and the whole thing doesn't work!*

- The app is supposed to be running on C++ (the joke is I haven't used it in 8 years because it is so time consuming).
- I'm writing the algorithms in java and then copying them over because it compiles faster and is easier to test.
- If I run out of time I might not be able to get it to C++ but the joke stands becasue I tried.

So yeah, that clearly didn't work out as you can see:



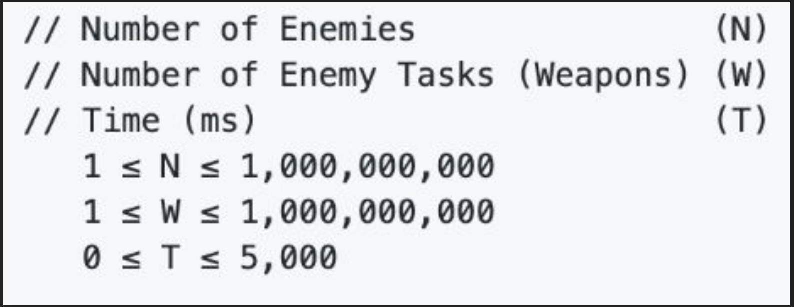
Languages

● **Java** 95.4% ● **C++** 4.5%

● **Batchfile** 0.1%

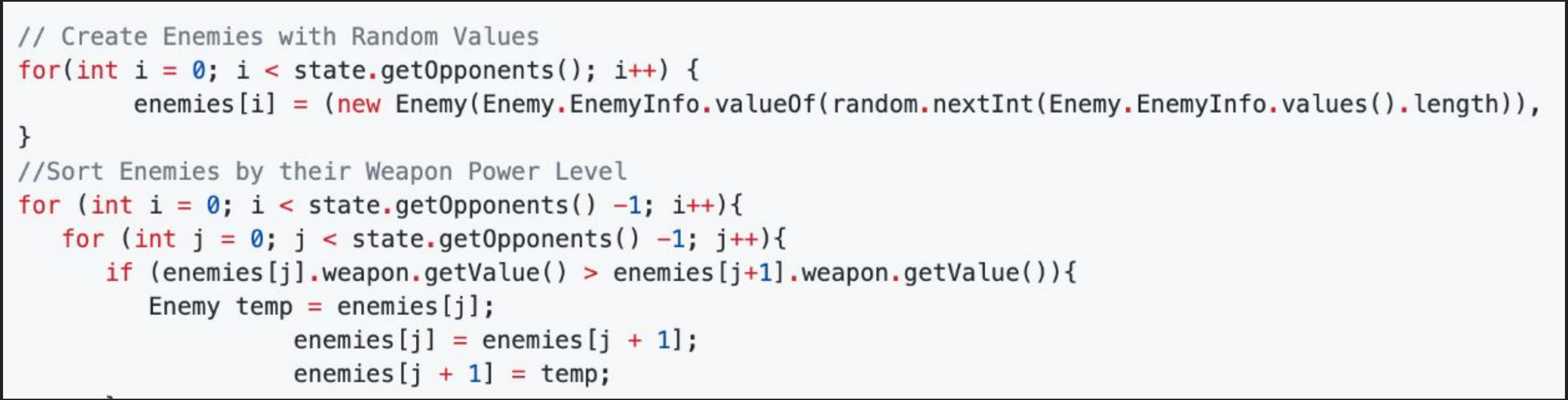
But... it *COULD* happen!

Background - How it works



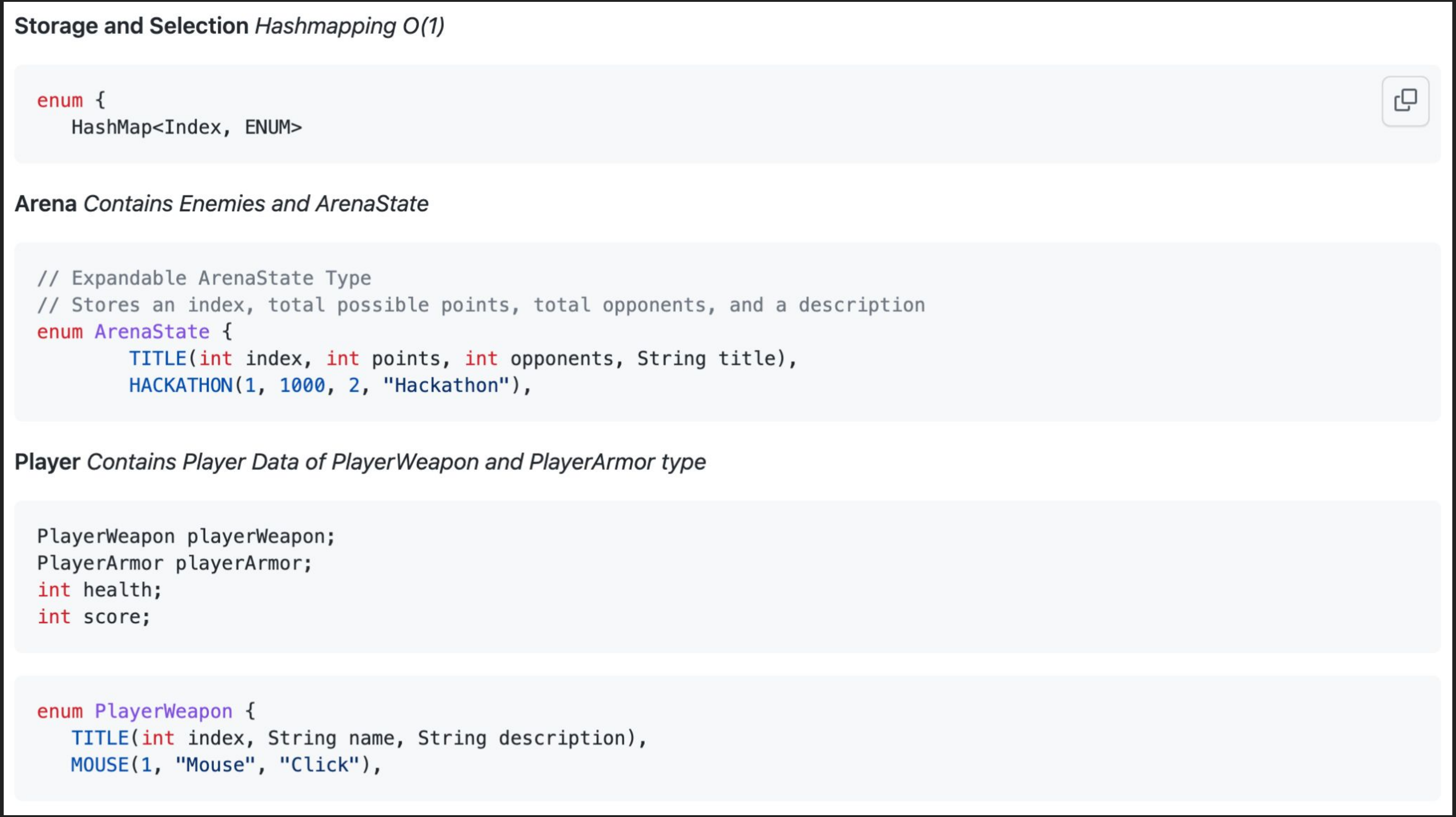
```
// Number of Enemies                      (N)
// Number of Enemy Tasks (Weapons) (W)
// Time (ms)                              (T)
   1 ≤ N ≤ 1,000,000,000
   1 ≤ W ≤ 1,000,000,000
   0 ≤ T ≤ 5,000
```

The Game Controller creates a Player and an Arena. The Player can move through Arenas and face different enemies from a Collection of Enemies.



```
// Create Enemies with Random Values
for(int i = 0; i < state.getOpponents(); i++) {
        enemies[i] = (new Enemy(Enemy.EnemyInfo.valueOf(random.nextInt(Enemy.EnemyInfo.values().length)),
}
//Sort Enemies by their Weapon Power Level
for (int i = 0; i < state.getOpponents() -1; i++){
   for (int j = 0; j < state.getOpponents() -1; j++){
      if (enemies[j].weapon.getValue() > enemies[j+1].weapon.getValue()){
         Enemy temp = enemies[j];
                    enemies[j] = enemies[j + 1];
                    enemies[j + 1] = temp;
```

Data



Storage and Selection *Hashmapping O(1)*

```
enum {
   HashMap<Index, ENUM>
```

Arena *Contains Enemies and ArenaState*

```
// Expandable ArenaState Type
// Stores an index, total possible points, total opponents, and a description
enum ArenaState {
        TITLE(int index, int points, int opponents, String title),
        HACKATHON(1, 1000, 2, "Hackathon"),
```

Player *Contains Player Data of PlayerWeapon and PlayerArmor type*

```
PlayerWeapon playerWeapon;
PlayerArmor playerArmor;
int health;
int score;
```

```
enum PlayerWeapon {
   TITLE(int index, String name, String description),
   MOUSE(1, "Mouse", "Click"),
```

Why Hash Mapping?

- Search happens in O(1) time.
- Sorting (Generating Arenas) relies on search
  - Sorting happens in O(n) time but must meet T < 5000.
  - Arena Generation happens in O(2n + 1) time.*
- ENUM can be indexed or sought by ENUM value.
- Data is expandable to 1,000,000,000 of each instance
  - PlayerWeapon, EnemyInfo, EnemyWeapon, ArenaInfo
  - With mapping: O(2n + 1)
  - Without mapping: O(3n)

Expansion

- Currently in Java
- Created for duality (Java or C++) so it can be easily translated to support more systems.
- STATIC FRAMEWORK
  - The framework does not need to be modified to add new adventures, missions, fights, enemies, weapons, or anything.
  - Support for data import can be added to framework (working model)
- ENUM
  - All data: EnemyInfo, EnemyWeapon, Arena, PlayerWeapon, PlayerArmor can be expanded up to 1,000,000,000 instances.
  - Easily add new items to improve the game!