**Final Equation (Bold = Conditional):**

* **Average:** (Character’s Name) the (Class)’s Max HP is:[(Max of Class hit die + CON mod) + (Level - 1 \* Avg class hit die) + (Level \* CON mod (this is decided with an array)) **+ [(Level \* 2 (If they have Tough feat)) + Level (if they are hill dwarf) This is all added onto the total after the base calculations]]**
* **Rolled:** (Character’s Name) the (Class)’s Max HP is: [(Max of Class hit die + CON mod) + (Roll number of appropriate hit dice equal to character level - 1) + (Level \* CON score (this is decided with an array)) **+ [(Level \*2 (If they have Tough feat)) + Level (if they are hill dwarf) This is all added onto the total after the base calculations]]**

Dictionary will contain the class and their appropriate hit die {“Barbarian”, 12} that compares the inputted class with those in the list to get the hit die.

**Pseudo Code:**

Code Starts  
 Public Bool hasTough;

Public Bool isHillDwarf;

Public Bool RolledHP;

Public String class;

Public int level;

Public int con;

Public String Name;

*//Establishes the strings, bools, and ints used in code*

Private Dictionary <String, double> hitDie{Class, hitDie};

Private int[] conMod { -5….+10};

Private hp;

*//Creates a dictionary for the class and respective hit die, an array for the constitution modifier, and an int to take the returned value of CalcHp()*

Start(){

hp = CalcHp();

Debug.Log(DisplayMessage());

}

Private int CalcHp(){

int maxHP;

hitDie(Class, maxHP = hitDie);

maxHP += conMod[CON-1];

*//Gives the initial max HP for the given class and the constitution modifier, making it possible to have a level 1 character in the code*

If (level>1){

for(levels>1){

maxHP += RolledHP ? Random.next(1, hitDie.Value +1) : (hitDie.Value+1 / 2) +.5);

maxHP += ConMod[CON-1];

}

}

*//Calculates the rest of the HP for character’s level, taking account if average or rolled HP was selected*

If (hasTough){

maxHP += 2 \* level;

}

*//Adds amount of HP needed if Tough feat is selected*

If (isHillDwarf){

maxHP += level;

}

return maxHP;

*//Adds amount of HP needed if character is a Hill Dwarf*

}

Private String DisplayMessage(){

Return ($“{Name} the {Class}’s Max HP is {hp} at level {level} with a CON score of {CON} and {isHillDwarf ? “is” : “is not”} a Hill Dwarf {hasTough ? “with” : “without”} the Tough Feat. I want the HP {RolledHP ? “rolled” : “averaged”}.”

}

*//States the information of the character with the given values*