Lab 4

Summer Smith, Chandler Guzman, Lauren Holloway

Variables

Variable	Possible Values	Data Type
characterName	any	string
characterLevel	1-20	int
conScores	1-30	Dictionary(int, int)
conScore	1-30	int
characterRaces	Player's handbook races (See table below)	Dictionary (String, Int)
characterRace	Player's race	String
characterClasses	Provided classes (See table below)	Dictionary (String, int)
characterClass	Player's class	String
toughFeat	true/false	boolean
stoutFeat	true/false	boolean
diceAveraged	Averaged (true), rolled (false)	boolean
hitPoints	-	int
roll	1-12 (based on dice type)	int
totalRoll	1-240 (based on total from hit die rolls)	int

Methods

Method Name	Purpose	Input	Return
setup	Declares the conScore, characterRace, and characterClass Dictionaries.	N/A	void
calculateHP	Calculates the player's HP based on level, constitution, race, feats, roll type, and class.	N/A	int
checkValues	Ensures all values are within constraints as listed above for inputs	N/A	void
output	Prints the user's character stats as specified below and total HP points	N/A	void

Race Dictionary

Race (Key)	Extra HP (Value)
Aasmar	-
Dragonborn	-
Dwarf	+2
Elf	-
Gnome	-
Goliath	+1
Halfling	-
Human	-
Orc	+1
Tiefling	-

All those without extra HP would be assigned a value of 0.

```
Dictionary<String, int> characterRaces = new Dictionary<String, int>()
{
        { “Aasmar”, 0 },
        { “Dragonborn”, 0 },
        { “Dwarf”, 2 },
        { “Elf”, 0 },
        { “Gnome”, 0 },
        { “Goliath”, 1 },
        { “Halfling”, 0 },
        { “Human”, 0 },
        { “Orc”, 1 },
        { “Tiefling”, 0 },
};
```

Class Dictionary

Class (Key)	Hit Die (Value)
Artificer	d8
Barbarian	d12
Bard	d8
Cleric	d8
Druid	d8
Fighter	d10
Monk	d8
Ranger	d10
Rogue	d8
Paladin	d10
Sorcerer	d6
Wizard	d6
Warlock	d8

```
Dictionary<String, int> characterClasses = new Dictionary<String, int>();
{
        { “Artificer”, 8 },
        { “Barbarian”, 12 },
        { “Bard”, 8 },
        { “Cleric”, 8 },
        { “Druid”, 8 },
        { “Fighter”, 10 },
        { “Monk”, 8 },
        { “Ranger”, 10 },
        { “Rogue”, 8 },
        { “Paladin”, 10 },
        { “Sorcerer”, 6 },
        { “Wizard”, 6 },
        { “Warlock”, 8 },
};
```

Constitution Dictionary

Constitution Score (Key)	Modifier (Value)
1	-5
2 - 3	-4
4 - 5	-3
6 - 7	-2
8 - 9	-1
10 - 11	+0
12 - 13	+1
14 - 15	+2
16 - 17	+3
18 - 19	+4
20 - 21	+5
22 - 23	+6
24 - 25	+7
26 - 27	+8
28 - 29	+9
30	+10

For this, we can use a for loop that starts at 1 and ends at 30. If the number is even (int % 2 == 0), the modifier value goes up by 1.

HP Calculation

Needed for calculation:

- characterLevel
- Constitution Score
- Race
- Feat
- Averaged or Rolled

```
if (diceAveraged)
{
	int averagedRoll; // This is turned into an int so that it can be rounded up.
	switch (characterClasses[characterClass])
	{
		case 6:
			averagedRoll = 4;
			break;
		case 8:
```

```
                averagedRoll = 5;
                break;
            case 10:
                averagedRoll = 6;
                break;
            case 12:
                averagedRoll = 7;
                break;
        }

        hitPoints = (characterLevel * averagedRoll) + (characterLevel * conScores[conScore]) + (characterLevel *
characterRaces[characterRace]);
}
        else
{
        int randomRoll = Random.Range(1, characterClasses[characterClass] + 1);
        hitPoints = (characterLevel * averagedRoll) + (characterLevel * conScores[conScore]) + (characterLevel *
characterRaces[characterRace]);
}
```

Output Statement

```
if (toughFeat && stoutFeat) {
        Debug.Log("My character " + characterName + " is a level " + characterLevel + " " + characterClass + " with a
CON score of " + conScore + " and is of " + characterRace + " race and has Tough and Stout feats. I want the HP " +
rollType);
} else if (toughFeat) {
        Debug.Log("My character " + characterName + " is a level " + characterLevel + " " + characterClass + " with a
CON score of " + conScore + " and is of " + characterRace + " race and has Tough feat. I want the HP " + rollType);
} else if (stoutFeat) {
        Debug.Log("My character " + characterName + " is a level " + characterLevel + " " + characterClass + " with a
CON score of " + conScore + " and is of " + characterRace + " race and has Stout feat. I want the HP " + rollType);
} else {
        Debug.Log("My character " + characterName + " is a level " + characterLevel + " " + characterClass + " with a
CON score of " + conScore + " and is of " + characterRace + " race and has no feats. I want the HP " + rollType);
}
```

Lab 5

Summer Smith, Chandler Guzman, Lauren Holloway

Variables

Variable	Class	Possible Values	Data Type
characterName	Character	any	string
characterLevel	Character	1-20	int
conScores	DungeonMaster	1-30	Dictionary(int, int)
conScore	Character	1-30	int
characterRaces	DungeonMaster	Player's handbook races (See table below)	Dictionary (String, Int)
characterRace	Character	Player's race	String
characterClasses	DungeonMaster	Provided classes (See table below)	Dictionary (String, int)
characterClass	Character	Player's class	String
toughFeat	Character	true/false	boolean
stoutFeat	Character	true/false	boolean
diceAveraged	Character	Averaged (true), rolled (false)	boolean
totalHP	Character	-	int
rollType	Character	Averaged, rolled	string
rollHP	Character	1-240 (based on total from hit die rolls)	int

Methods

Method Name	Class	Purpose	Input	Return
setUp	DungeonMaster	Declares the conScore, characterRace, and characterClass Dictionaries.	N/A	void
calculateHP	Character	Calculates the player's HP based on level, constitution, race, feats, roll type, and class.	N/A	int
checkValues	DungeonMaster	Ensures all values from Character are within constraints as listed above for inputs	characterLevel, conScore, characterRace, characterClass	void
output	Character	Prints the user's character stats as specified below and total HP points	N/A	void
rollDice	Character	Rolls dice for calculateHP if the character would like their dice rolls randomized	N/A	void

DungeonMaster.cs

- Will define all the dictionaries needed (classes, race, constitution)
- Will check the values for Character to ensure they align with dictionaries

Character.cs

- Will store character information (HP, race, class, etc.)
- Will calculate HP
- Will roll dice (if needed)
- Will output