Pathfinder Character Sheet

By Peter Wright

# Introduction

Pathfinder is a tabletop role-playing game based off the popular franchise Dungeons and Dragons. The major differences between Dungeons and Dragons and Pathfinder are that Pathfinder has a larger amount of customization because of its large number of extra rules and content. This application is a form that is modeled after the Pathfinder character sheet (a paper reference for a player’s character in a Pathfinder game). This Pathfinder character sheet allows you to store and retrieve characters you have created as well as update information on the sheet based on certain important fields. This application was created using C#’s .NET forms application in visual studio and can be easily run on any Windows PC.

# Installation

The program is stored as an executable file. The installation process is simply the downloading of the .exe file and running the program. It helps to have text files for your saved character sheets already created in the directory this file is located inside, but it is not necessary. This program cannot be run on Mac OS without third party software.

# User’s Manual

## Introduction

This application is essentially a collection of fields that are used to store information about your Pathfinder character in a quick and easy way. In this documentation we will go over what each section of the fields represents and what their different elements mean.

## Save, Load, and Update

At the top of the application there is a text field labeled “Save Path:”. This field is used to get the file path to a text file to either read or write your character information to. The choice in whether to read or write the information to this text file is indicated by the “Save” and “Load” buttons respectively. These buttons are located to right of the text field. Lastly, you will notice a larger “Update” button located here and at the top of the second page. This button updates the character sheet to automatically include certain important values (ability scores, base attack, etc.) to other labels on the sheet. It also updates the underlying model of the application. This does not mean that the update button will save your character.

## Character Description

This section of the application includes the defining traits of the character. This includes their name, physical description, religion, player, etc. All these fields are stored as strings in the model, even if they hold numeric, Boolean, or other values.

## Ability Scores

This section holds the ability scores of the character as well as their bonus and modifiers to those scores. Ability score modifiers are calculated with the formula: ability score – 10 / 2. The “Ability Score” field holds the base ability score value while the “Temp Adjustment” field holds the value of a bonus given to the base ability score. These fields store integer values and display an error message in the “Save Path:” field when they don’t hold numeric values.

## Defenses

This section includes the fields related to hit points, initiative, armor class, saving throws, damage reduction (DR), and spell resistance. The hit points part includes the “Total” field, which holds the max hit points of the character, the “Wounds/Current HP” field, which holds the current hit points of the character, and the “Nonlethal Damage” field, which holds the character’s non-lethal damage. These fields store integer values and display an error message in the “Save Path:” field when they don’t hold numeric values.

The next part of this section is the initiative section. This only stores modifiers to initiative as the only other modifier to initiative is the character’s dexterity. This field stores an integer value and displays an error message in the “Save Path:” field when they don’t hold numeric values.

The next part is the armor class section. This holds the modifiers to the character’s armor class, which is used to tell if an attack hits or misses the character. These bonuses include armor bonus, shield bonus, size bonus, natural armor bonus, deflection bonus, and miscellaneous bonus. Touch armor class is armor class without the armor and shield bonuses added, while flat-footed armor class is armor class without the character’s dexterity added. These fields store integer values and display an error message in the “Save Path:” field when they don’t hold numeric values.

The last part of the defenses section is the saves section. This includes the base and bonuses to fortitude, reflex, and will saves. These bonuses include constitution, dexterity, and wisdom modifiers respectively, as well as their magic, miscellaneous, and temporary modifiers. These fields store integer values and display an error message in the “Save Path:” field when they don’t hold numeric values.

Lastly, damage reduction and spell resistance are the character’s resistance to certain types of type and are stored as strings.

## Combat Maneuvers

This section holds the base attack bonus and bonuses to combat maneuver checks and defenses. The CMB (combat maneuver bonus) is used for combat maneuvers like trip, grapple, etc. while CMD (combat maneuver defense) is your resistance to such attacks. Both of these sections only have a size bonus text field for input. These fields store integer values and display an error message in the “Save Path:” field when they don’t hold numeric values.

## Weapon Attacks

This section holds 5 entries for the attacks the character can make. Each weapon attack has a field for the name of the attack, the attack bonus to hit, the critical hit range, the type of attack it is, the range of the attack, the amount of ammunition left if the attack uses if any, and the damage the attack does. These values are stored as strings.

## Speed

This section holds the speed values of the character. These speeds describe how fast the character moves (in feet) in different situations. These include the base speed, speed with armor, speed while flying, speed while swimming, speed while climbing, and speed while digging. These fields store integer values and display an error message in the “Save Path:” field when they don’t hold numeric values.

## Skills

This section holds a list of all the skills the character can become proficient in. These skills have attributes that include: whether it is a class skill, an ability score bonus, number of ranks put into the skill, and miscellaneous modifiers. Certain skills also have a name field because there are different types for this skill. The class skills gain a bonus of +3 if the character has ranks in that skill. The total skill bonus is equal to class skill bonus + ability modifier + ranks + miscellaneous modifiers. The ranks and miscellaneous modifier fields store integer values and display an error message in the “Save Path:” field when they don’t hold numeric values. The name fields hold string values.

## Conditional Modifiers and Languages

This section is a small section that holds information about what buffs/debuffs the character currently has and the languages the character speaks. These fields store their values also strings.

## AC Items

This section holds information about items the character holds that given it bonuses to its armor class. Each item has seven properties that can be explained on this chart: the item’s name, bonus to armor class, type of the item, armor check penalty, spell failure chance, weight of the item, and other properties. Name, type, and properties are all stored as string while the other fields are stored as integers. Numeric fields also have totals given at the bottom of the chart. These fields display an error message in the “Save Path:” field when they don’t hold numeric values.

## Gear

This section holds a list of all the items held by the character as well as the weight of each item. The total weight of all the items is then displayed at the bottom of the list. The name of each item is stored as a string while their weights are stored as integers and display an error message in the “Save Path:” field when they don’t hold numeric values.

## Money

This section holds the different types of money the character has. These types include: copper pieces, silver pieces, gold pieces, and platinum pieces. These fields are stored as integers and display an error message in the “Save Path:” field when they don’t hold numeric values.

## Feats and Special Abilities

This section holds the feats and special abilities taken by the character. These are abilities the character can gain as they level up, from their class, from their race, or through some other source. These values are stored as strings.

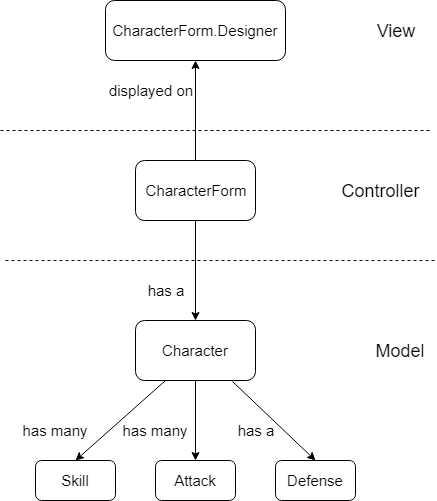
## Experience

This section holds the amount of experience the character possesses and the number needed to reach the next level. These fields are stored as integers and display an error message in the “Save Path:” field when they don’t hold numeric values.

## Modifiers

Throughout the application there are also modifiers sections. These are used to keep track of bonuses to different aspect of the character. They store string information as they are mostly used for note taking.

# Design



## Summary

In creating this application, I used Pathfinder’s standard character sheet as a reference. I’ve seen many third-party character sheets with different benefits, but using the standard form was in my opinion the simplest solution. However, in doing this I had to simplify my original design. A pathfinder character can be broken down into many distinct parts: race, class, skills, ability scores, defenses attacks, etc. Originally many of these parts had their own class associated with them. However, as I began to move toward a model that mimicked Pathfinder’s standard character sheet I began to realize that each part didn’t need its own class. This lead to my current design. There are only 4 main sections of my character sheet that are separated into classes. Theses sections are Attack, Defense, Skills, and general character information. All elements on the form fall into one of these four groups and as such can be easily tracked and managed.

## Data Structures

A lot of the complexity of this project came from two distinct data structures: lists and matrixes. List C# objects are primarily used over arrays in this program as the biggest list, the skill list, holds objects instead of lower level data. However, this does not mean that arrays are never used. There are multiple arrays in the program used to hold string and integer data. The matrices are also made with arrays, not lists. This is because matrices were only used to represent tables on the character sheet. These tables had a set number of rows and columns and held only lower level data, so there was no need to dedicate a list object to them. There was no real list or matrix arithmetic involved above simple addition since the Pathfinder system was designed to be easy to use.

One important data structure that was constantly used throughout the program is C#’s Properties. A property works like an accessor and mutator function rolled into one. As such each variable that is connected to a text field in the form has an associated Property. This allows for easy input and output from the form to the class objects.

## File Structures

The file architecture was automatically created by visual studio. As such all C# files are stored in the main directory. This includes the class files as well as the client and designer files used to run and create the form. The images used to give the character sheet its look can be found in the “img” directory and the images follow the pattern of “SheetPart#.jpg”. The executable and serialization files can be found in the “bin/Debug” directory. This application uses serialization files to store previous characters created. This is done through the “save” and “load” methods of the PlayerCharacter class. In these methods the serialization files are searched and if found read/written in the proper format using a StreamWriter/Reader C# object. These serialization files must be text files with the correct formatting or an error will be thrown.

## Classes

### Skill

The Skill class is used to represent the skills the character knows and can be trained in during the game. Skills have six distinct attributes: name, associated ability score, class skill, ability score bonus, ranks, and other bonuses. These attributes are all represented by member variables with associated properties. The only functions given to this class are a constructor and a function to calculate the total bonus to the skill roll.

### Attack

The Attack class is used to represent the attacks the character can make. These attacks have seven attributes associated with them: name, range, attack type, critical hit range, damage, bonus to hit rolls, and ammunition. Each of these attributes is saved as a member variable. This class only has a constructor function as it is only used to store attack data.

### Defense

Unlike the other classes the Defense class doesn’t represent a single aspect the character has. Instead, it holds all defensive information pertaining to the character. This includes hit points, armor class, saves, resistances, and armor class items. While many of the of these are stored as simple member variables in the class, others are stored as matrices (armor class items) and arrays (saves). The functions found in the Defense class include the constructor as well as classes to get total bonus for saves, armor class, and armor class items attributes.

### PlayerCharacter

The PlayerCharacter class represents the whole of the character. This means it has objects for each of the previously mentioned classes. However, it also stores the information about the character that does not fit into the other classes. Attributes like money, experience, physical traits, etc. are all stored in the Character class in member variables. The functions in the PlayerCharacter class include the constructor, functions that return totals of certain attributes, and the serialization methods. The serialization methods take a string file path as a parameter and use a StreamReader/Writer to transfer information to and from the file, outputting an error if an exception is thrown.

### CharacterForm

The CharacterForm class represents the form the user views and uses to edit their character. The CharacterForm class is a child of the C# Form class and has a designer attached to it that holds all the text fields, labels, buttons, pictures, and events used in the application. This means that while the only member variable seen is the PlayerCharacter object, it actually has hundreds more variables at its disposal. This is especially evident in updateCharacter and updatePage functions that update the PlayerCharacter object to match the form and the form to match the PlayerCharacter object respectively. Other than these two functions the only non-inherited functions are the constructor and the event listener functions.

## Code

The code for this program can be found in a separate file called “code.txt” as it is too large to put in this section.

# Test Plan

Most of the testing for this project is done through the use of the serialization files as they offer a quick and easy way to test if any of the functions, properties, or classes are not working. The serialization files are easy to manipulate as their format has a high focus on readability. Therefore, to test any aspect of the code one must simply go into a serialization file and find the aspect they wish to change. A file named “testSave.txt” was created expressly for this purpose. Any changes/errors can be seen when the application loads the specified file.

# Conclusion

Pathfinder is considered one of the most popular tabletop role-playing games in the world and is played by me and my friends at least once a week. My love for this game inspired this project and made the entire coding experience a lot more fun. I originally created this project with the intention to make it easier to make characters in pathfinder for me and my friends. Now I plan on uploading this to an opensource site for other people to use and hopefully find the same love for Pathfinder as I did. This program was a test in using my newly refined programming skills to innovate something that is dear to me.

# Bibliography

Microsoft. “Form Class”. MSDN Microsoft. <https://msdn.microsoft.com/en-us/library/system.windows.forms.form(v=vs.110).aspx> (accessed June 20, 2018).

OGN SRD. “d20PFSRD”. Pathfinder SRD. <https://www.d20pfsrd.com/> (accessed June 22, 2018).

Pathfinder. “PathfinderRPGCharacterSheet.pdf”. Pathfinder RPG Character Sheet. <https://www.wou.edu/~joles12/CS196/PATTH/PathfinderRPGCharacterSheet.pdf> (accessed July 1, 2018).

Microsoft. “StreamWriter Class”. MSDN Microsoft. <https://msdn.microsoft.com/en-us/library/system.io.streamwriter(v=vs.110).aspx> (accessed June 20, 2018).

Microsoft. “StreamReader Class”. MSDN Microsoft. <https://msdn.microsoft.com/en-us/library/system.io.streamreader(v=vs.110).aspx> (accessed June 20, 2018).