# RPG Phase 1 – The Hero

In case you hadn’t heard, I’m a HUGE RPG (role-playing game) nerd. I’m not talking about Final Fantasy VII or Skyrim. I’m talking about Dungeons & Dragons, The Wheel of Time, Dead Reign, and other tabletop RPGs you play using dice, a pencil, and a piece of paper. One of my favorite things to do is to create tools and utilities to simplify the more tedious tasks of playing RPGs.

To practice designing and building your own classes, we’re going to build a simple RPG program.

## Requirements

In traditional role-playing games, you have a Game Master (GM) who is responsible for telling the story, challenging players, and adjudicating outcomes of combat and looting! For the purposes of this lab, the GM will have dice and keep them in a bag. The GM won’t have much of a game, though, if he/she doesn’t have heroes to go questing. For this lab, let’s just focus on creating one hero. The GM also knows all of the rules to the game being played. So, let’s build a tiny game.

1. The program will use the console for input and output. The game will be entirely text based.
2. At the start of the game, the GM will automatically create a randomly generated hero (based on criteria listed later). Once the hero is displayed to the user (complete with attributes and bonuses), the user will be asked if he/she wishes to “re-roll” the character. If “yes”, then the ENTIRE character is built again from scratch. The process recycles until the user approves the current character.
3. Once the user accepts a character build, the program will prompt the user to give the character a name.
4. You must have ConsoleUI, GameMaster, Hero, DiceBag, and Die classes in your project. Below are the descriptions for each.

ConsoleUI

This class is responsible for all interactions with the user, including both input and output. However, this class should have no “business intelligence” outside of input validation. This class will be told to output data to and gather input from the user. This class will be controlled by the GameMaster class.

**SPECIAL NOTE:** If you completed the ConsoleIO lab with full credit (no bugs), you may use that library INSTEAD of creating this class.

GameMaster

The GM has a DiceBag that he/she uses to do just about everything, including create the Hero character. The GM is responsible for keeping track of the Hero character. The GM is also the brains of the operation and will use an instance of the ConsoleUI class to communicate with the user.

DiceBag

The DiceBag is able to produce *any* kind of die needed for *any* kind of roll. When the GM needs to roll dice, it tells the DiceBag how many dice need to be rolled, how many sides each die has, and whether there is a modifier (+2, -7, etc). While the modifier is optional, every die roll will have at least one die which will have at least two sides. The GM may ask for any number of dice with any number of sides. This means that 12d34 (meaning 12 dice, each containing 34 sides) is a completely legit request. The DiceBag’s job is to take the request, create the appropriate die (or dice), roll it/them the correct number of times, add the modifier (if any) to the total, and return the result.

Die

Every die has a set number of sides, can be rolled (select a random value in the appropriate range), and keeps track of the value of the last roll it made. Once set, the number of sides for that instance of Die should NEVER be able to change.

Hero

The Hero is the most complicated class in this lab. The Hero is identified by his/her name and has three attributes (discussed shortly), bonuses based on the attributes, base and current Health Points (HP), and base and current Magic Points (MP). In general it is better to have higher values in each of these stats

The attributes are Strength, Dexterity, and Intelligence. Strength affects how much HP you have and the damage you deal in combat, Dexterity indicates your ability to successfully land attacks as well as dodge those of your enemies, and Intelligence determines your Magic Points and Spell Strength.

* For each attribute Roll 3d6 (that’s three six-sided dice) for a possible value range of 3-18.
  + If your total is 16 or higher, roll an additional 1d6
    - If that bonus die results in a 6, roll another bonus die. Regardless of the result, stop rolling
      * This means that the max value a starting attribute may have is 30
* Base HP is initially equal to Strength x 10. When the character is first created, the Current HP is equal to the Base HP
* A Damage Bonus will apply to high Strength characters. The Damage Bonus is equal to every point of Strength over 15. For example, if a character has a Strength attribute of 19, the Damage Bonus would be 4.
* A negative Damage Bonus applies for every point below 10.
* A Strike Bonus is awarded for high Dexterity characters. The Strike Bonus gains +1 for every TWO points above 14. A character with a Dexterity attribute of 15 has a Strike Bonus of +0, but a character of Dexterity 16 gains a Strike Bonus of +1. A character of Dexterity 20 would have a Strike Bonus of +3
* A negative Strike Bonus applies for every TWO points below 10, giving a character with a Dexterity of 6 a Strike Bonus of -2.
* A Dodge Bonus is awarded to high Dexterity characters as well. For every TWO points over 15, the character gains a +1 to his/her Dodge Bonus. This means characters with a Dexterity of 17 or 18 gain a Dodge Bonus of +1.
* A negative Dodge Bonus applies for every TWO points below 11. Characters with a Dexterity of 7 or 6 both have a Dodge Bonus of -2.
* A character‘s Base MP is determined by multiplying his/her intelligence by 5. When a character is first created, the Current MP is always set equal to the Base MP.
* A character with a high Intelligence attribute also gains a spell strength bonus. For every point over 15, his/her Spell Strength gains +2.
* Likewise, a negative penalty is applied to Spell Strength when the character has a low Intelligence. Spell Strength receives a -2 for every point below 10.

Finally, the Hero class needs to implement an override of the toString() method. This method will create a string containing all of the Hero’s information formatted in a useful way. Think “label & value” for each point of data in the Hero. The format is up to you, but you will be assessed on readability, clarity, and thoroughness.

It should be known that you will add to and refactor the code you produce for this lab. In the future, you will need to be able to change the values of the Attributes of a Hero and, thus, change the associated stats and bonuses. Keep this in mind when designing and implementing your classes. Also, ALL classes must practice proper encapsulation (all data fields are private; public getters and setters are provided, but only as needed).

## Rubric

**AUTOMATIC ZERO:** Your application uses the main method for logical code (it must ONLY start the program), any class fails to practice proper encapsulation, or a character cannot be created by the GM.