Contents

Description

Below is the combined descriptions of each phase of the "Character creation" project. If working through this project for the first time, it is encouraged to work on one phase at a time, ensuring your execution of the project is correct by comparing with our solution before moving on.

Phase 1

Purpose

This first project is designed to teach you how to devise, implement, and submit solutions to programming problems. As you will be expected to complete projects not only this semester, but throughout your studies, one of our primary goals is to familiarize you with the submission process. This task will also ensure that you understand the basic concepts we have been studying thus far.

Skills

For this project, you will need:

- Exhibit an understanding of IDE features pertaining to object-oriented development,
- Analyze a simple design document,
- Implement a simple class containing multiple attributes and methods, and
- Write a test program for your implementation.

Knowledge

This assignment will familiarize you with the following important skills:

- How to read a UML diagram,
- How to interpret a series of simple instructions, and
- How to manipulate objects.

Tasks

Challenge

In short In this project, you are asked to develop a `Character` class, based (loosely) on a Dungeons and Dragons character sheet.

In more details Consider the following UML diagram for a `Character` class:

In order, the attributes will store:

- The name of the character,
- The race of the character,
- The gender of the character (M = male, F = female, O = other, H = hermaphroditic, etc.)
- The level of the character,
- The hit points (HP) of the character,

The methods to be implemented are as follows:

- Getters for the `charName`, `race`, `gender`, `level`, and `hitPts` attributes,
- Public setters for the `charName`, `race`, `gender`, and `level` attributes,
- Private setter for `hitPts`
  - health should always update when level updates (health is 10 at level 0 and 3 additional hit points per level; i.e. level 3 character will have `10 + (3 * 3) = 19` health),
- A `LevelUp` method that increases `level` by 1,
- A ToString method neatly displaying all the attributes in the method

Your tasks are to:

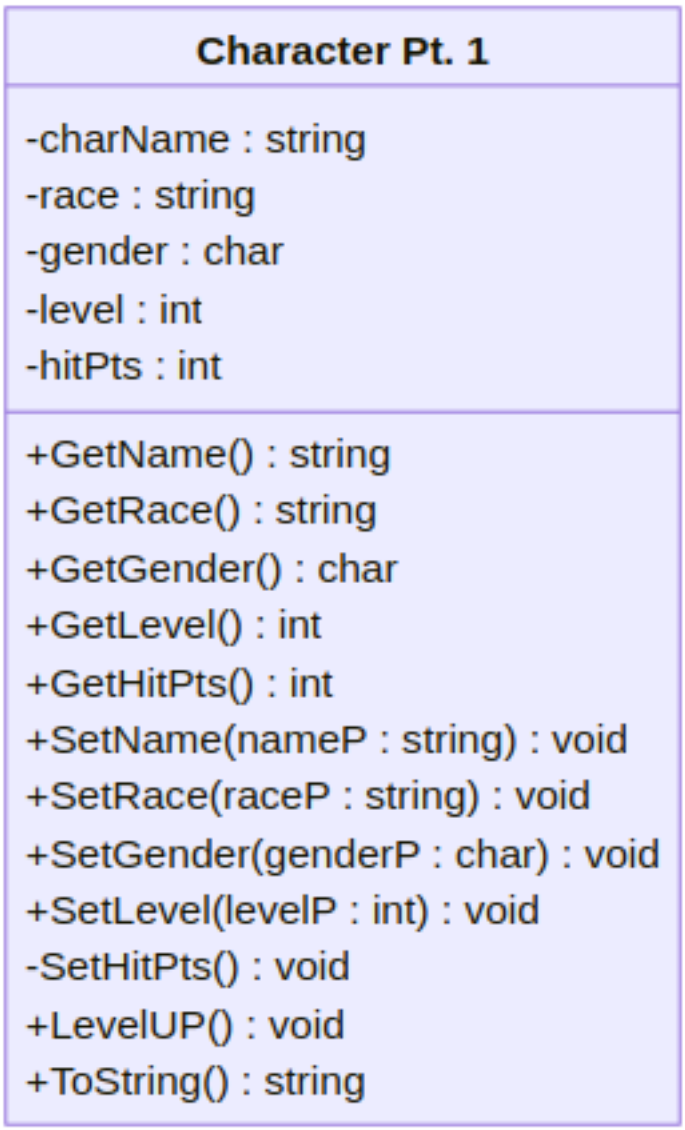

Figure 1: A UML diagram for the Character("Character Pt. 1") class (text version[2])

- Create a new project called "Character",
- Implement an `Character` class as described above, and
- Write a `Program` class containing a `Main` method that tests your implementation.

Note that **only your `Character` class will be assessed**; your `Program` class's role is to enable *you*, the programmer, to test your implementation. It should help you detect bugs or missing features. However, we will not review it when grading your work. If you decided to implement additional methods in your `Character` class to help you in debugging, that is fine, but they will *not* be reviewed either.

Finally, write your name and the date in a delimited comment at the beginning of the `Character.cs` file.

Example My character, *Totally McRealguy*, is a level 5 male orc. This is what my `ToString` method displays to the console:

```
Character Name: Totally McRealguy
Race: Orc
Gender: M
Level: 5
HP: 25
```

Oh wow, *Totally McRealguy* leveled up! Let's see what the console says now:

```
Character Name: Totally McRealguy
Race: Orc
Gender: M
Level: 6
HP: 28
```

Notice the increase in HP when he leveled up.

Bonus Extra points will be given if you implement both an empty and normal `Constructor` methods for your `Character` class.

Solution

Project 2 Phase 1 Solution (LINK HERE)

Phase 2

Purpose

This first project is designed to teach you how to devise, implement, and submit solutions to programming problems. As you will be expected to

complete projects not only this semester, but throughout your studies, one of our primary goals is to familiarize you with the submission process. This task will also ensure that you understand the basic concepts we have been studying thus far.

Skills

For this project, you will need:

- Exhibit an understanding of IDE features pertaining to object-oriented development,
- Expand on the class you made last time with new methods
- Construct a user interactive program
- Validate inputs from the user using boolean systems
- Make an object from the class you constructed

Knowledge

This assignment will familiarize you with the following important skills:

- How to manipulate objects
- How to use boolean systems (If/switch, loops)
- How to validate user input

Tasks

Challenge

In short In this project, you are asked to add two new methods to your Character class, then make a program asking the user to give you the values you need to make a new character with input validation.

In more details First, download the Project 2 Phase 2 Template (LINK HERE)

Consider the following UML diagram for the updated `Character` class:

In Character.cs:

- Add an empty constructor with the following default values
  - charName: "N/A"
  - race: "N/A"
  - gender: 'Z'
  - level: 0
  - hitPts: defined by `SetHitPts`
- Add a constructor with parameters (assign as normal)
  - Note: This should also use `SetHitPts`

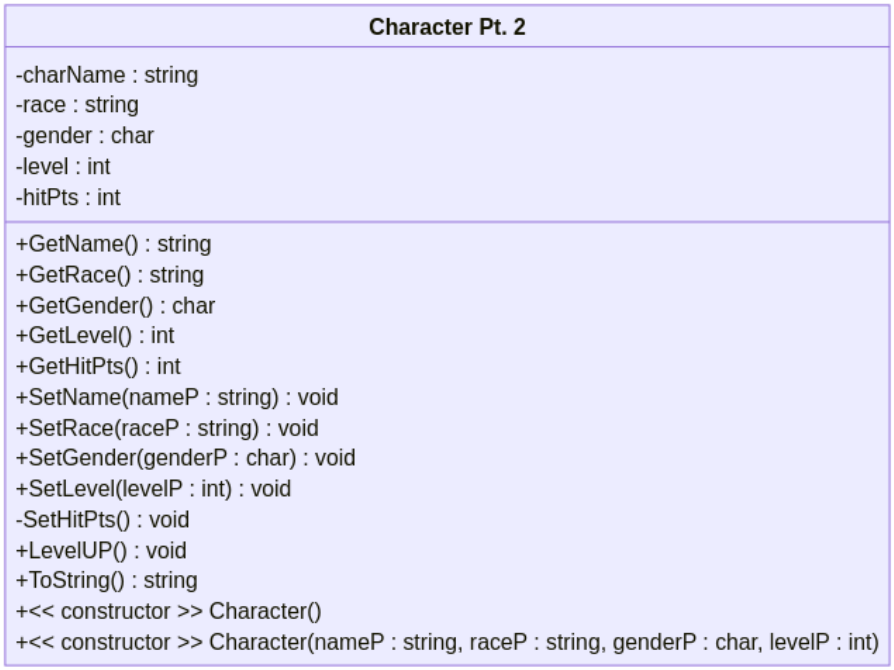

Figure 2: A UML diagram for the Character("Character Pt. 2") class (text version[4])

In Program.cs:

- Create a program that prompts the user to input the values to create the character.
- Each value will be restricted to inputs that fit these conditions:
  - name: no input validation needed
  - race: must be one of the following races: Human, Elf, Orc, Dragonborn, Tiefling (case insensitive)
  - gender: Possible values are (M)ale, (F)emale, (O)ther, (I)ntersex (case insensitive)
  - level: must be an int and between 0 and 30 (0/30 included)
- Create a new Character object and display it to the console.

Your tasks are to:

- Modify Character to add two constructors,
- Create a input validated program to get the character values from the user, and
- Create and display a new character object using the collected values.

Finally, write your name and the date in a delimited comment at the beginning of the `Program.cs` file.

Example The program starts by welcoming the user and asking for the character's name:

```
========================================
| Welcome to the DnD Character Creator! |
========================================

Please enter your character's name:
Totally McRealguy
What is Totally McRealguy's race? (Choose between Human,
 ↪  Elf, Orc, Dragonborn, or Tiefling
no
That is not a valid response, try again.
What is Totally McRealguy's race? (Choose between Human,
 ↪  Elf, Orc, Dragonborn, or Tiefling
orc
```

Notice that it repeats itself when given a wrong input. Additionally, it accepts lowercase values.

Continuing to gender:

```
What is Totally McRealguy's gender? (Choose between
 ↪  (M)ale, (F)emale, (O)ther, (I)ntersex)
no
```

```
That is not a valid response, try again.
What is Totally McRealguy's gender? (Choose between
 ↪  (M)ale, (F)emale, (O)ther, (I)ntersex)
why
That is not a valid response, try again.
What is Totally McRealguy's gender? (Choose between
 ↪  (M)ale, (F)emale, (O)ther, (I)ntersex)
m
```

And level:

```
What is Totally McRealguy's level? (Choose a number no
 ↪  greater than 30)
-1
That is not a valid response, try again.
What is Totally McRealguy's level? (Choose a number no
 ↪  greater than 30)
55
That is not a valid response, try again.
What is Totally McRealguy's level? (Choose a number no
 ↪  greater than 30)
3

****************************************

Thank you, here is your character:
Character Name: Totally McRealguy
Race: orc
Gender: M
Level: 3
HP: 19
```

Bonus Add these additional restrictions to your program:

- Tieflings can only be (I)ntersex
- Always store gender as Uppercase

Solution

Project 2 Phase 2 Solution (LINK HERE)

Phase 3

Purpose

This final phase is designed to use all of the programming strategies and concepts taught throughout the semester, along with arrays and poten-

tially for loops, to finalize the three-phase project you've been working on throughout the semester. This project is a bit more complex than the other two, so please **read carefully** and allocate enough time to work on it.

Skills

For this project, you will need:

- Exhibit an understanding of IDE features pertaining to object-oriented development,
- Expand on the class you made last time with new methods,
- Expand on the user interactive program you made last time to utilize the new methods,
- Work with arrays and loop structures,
- Validate inputs from the user using boolean systems,
- Make an object from the class you constructed

Knowledge

This assignment will familiarize you with the following important skills:

- How to manipulate objects
- How to use boolean systems (If/switch, loops)
- How to create/use arrays
- How to validate user input

Tasks

Challenge

In short In this project, you are asked some new attributes to your `Character` class, along with additional methods to use those attributes. Additionally, you should extend your user interactive program to use these new methods.

In more details First, download the Project 2 Phase 3 Template. (LINK HERE)

(Note: You may only use your phase 2 implementation **if you received full points for that phase**.

Consider the following UML diagram for the updated `Character` class:

In `Character.cs`:

- Add two new attributes:

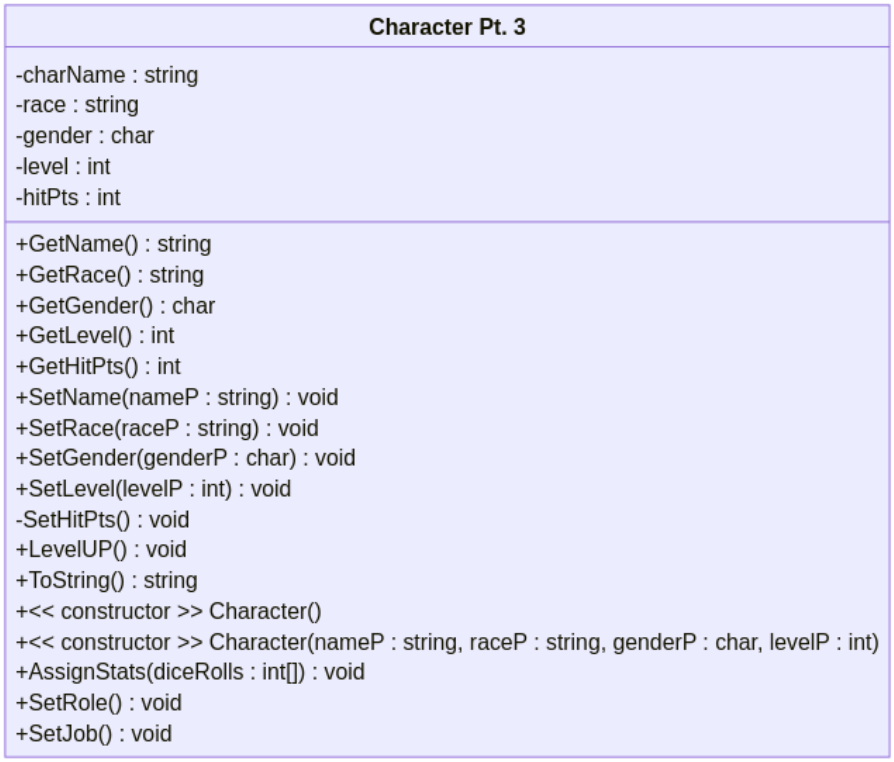

Figure 3: A UML diagram for the Character("Character Pt. 3") class (text version[6])

  - `stats`: an array of integers of size 6, where the integer at each index holds the value of a specific stat
    - 0: Strength
    - 1: Dexterity
    - 2: Constitution
    - 3: Intelligence
    - 4: Wisdom
    - 5: Charisma
  - `role`: a string (default value is "Peasant")
- Modify ToString to include the attributes above (Remember, you must display every cell in the array, not the array itself)
- AssignStats
  - Takes an array of "dice rolls" (That is, 6 randomly generated values between 1 and 20; done in `Program.cs`) as input
  - If the length of the array mentioned above does not equal 6, exit the method
  - Asks the user to assign each dice roll to a specific stat
- SetRole
  - Based on the stats assigned with AssignStats, this method should determine the potential roles the character can be:
    - If character has >15 in a specific stat, a role related to that stat should be available for selection
      - Strength: Barbarian
      - Dexterity: Rogue
      - Constitution: Sorcerer
      - Intelligence: Wizard
      - Wisdom: Cleric
      - Charisma: Bard
    - If character has >15 in two stats, then they should also be offered additional roles
      - Strength and Dexterity: Fighter
      - Strength and Charisma: Paladin
      - Dexterity and Wisdom: Monk
      - Intelligence and Wisdom: Druid
      - Wisdom and Charisma: Warlock
    - If no stats are >15, their role is Peasant
  - Array of all role options should be set at the beginning in order of stat.
    - Example Array Structure: (str, str/dex, str/cha, dex, dex/wis, con, ...)
  - Once determining what roles are available to them, user must pick which role they want. Their choice should be validated.
  - Once validated role is provided, value is stored in `role`, and method ends.

In `Program.cs`

- After the code written in Phase 2, add the following:
  - Create an array of length 6 to store dice rolls
  - Fill array with 6 randomly generated integers between 1 and 20 (inclusive)
    - This is to mimic rolling 6 20-sided dice
  - Call AssignStats, passing in the array of dice rolls
  - After `AssignStats`, display character again, then call `SetRole`

Your tasks are to:

- Modify `Character` to add two attributes,
- Modify the `ToString` method,
- Add two new methods,
- Modify the `program.cs` file to use the new methods.

Finally, write your name and the date in a delimited comment at the beginning of the `Program.cs` file.

Example Let's assume that the following character has already been created:

```
Character Name: Totally McRealguy
Race: orc
Gender: M
Level: 3
HP: 19
Strength: 0
Dexterity: 0
Constitution: 0
Intelligence: 0
Wisdom: 0
Charisma: 0
Role: Peasant
```

Notice that the ToString has now changed to show the new, currently unassigned attributes.

Now, after randomly generating 6 dice values, we call `AssignStats` and pass in the values. The following is what proceeds:

```
Stats:
Strength: 0
Dexterity: 0
Constitution: 0
Intelligence: 0
Wisdom: 0
Charisma: 0
```

```
Current Dice Roll: 3

Which stat would you like to assign this dice roll to?
1 - Strength
2 - Dexterity
3 - Constitution
4 - Intelligence
5 - Wisdom
6 - Charisma

no
That value is invalid, try again.
Which stat would you like to assign this dice roll to?
1 - Strength
2 - Dexterity
3 - Constitution
4 - Intelligence
5 - Wisdom
6 - Charisma
```

Notice how the user is reprompted when given a wrong value. Let's assign the 3 roll to wisdom:

```
5

****************************************

Stats:
Strength: 0
Dexterity: 0
Constitution: 0
Intelligence: 0
Wisdom: 3
Charisma: 0

Current Dice Roll: 14

Which stat would you like to assign this dice roll to?
1 - Strength
2 - Dexterity
3 - Constitution
4 - Intelligence
5 - Wisdom
6 - Charisma

5
This value is already assigned.
```

```
Current Dice Roll: 14

Which stat would you like to assign this dice roll to?
1 - Strength
2 - Dexterity
3 - Constitution
4 - Intelligence
5 - Wisdom
6 - Charisma
```

If we attempt to assign the new stat to wisdom, it tells you that the stat has already been assigned to.

Now let's go ahead and assume the user assigned all 6 values to the 6 stats, showing us this:

```
Character Name: Totally McRealguy
Race: orc
Gender: M
Level: 5
HP: 25
Strength: 14
Dexterity: 12
Constitution: 12
Intelligence: 2
Wisdom: 3
Charisma: 5
Role: Peasant
```

Next, we should call the `SetRole` method, which gives us the following:

```
Choose from the following roles:
Your stats are too low. You are assigned the role
↪  "Peasant."
```

Notice that our stats are too low to be assigned a stat, so we get "Peasant"

Let's try some different stats:

```
Character Name: Totally McRealguy
Race: orc
Gender: M
Level: 5
HP: 25
Strength: 20
Dexterity: 20
Constitution: 20
Intelligence: 20
Wisdom: 20
```

```
Charisma: 1
Role: Peasant
```

I got pretty lucky on this one! Let's see what the program shows me this time…

```
Choose from the following roles:
0: Barbarian
1: Fighter
3: Rogue
4: Monk
5: Sorcerer
6: Wizard
7: Druid
8: Cleric
```

Notice how the Bard and Warlock roles are still blocked, since my charisma stat is too low. If I try to select one of those…

```
2
This is not a valid choice, try again.
```

I am blocked. However, if I pick one that is available…

```
3
Character Name: Totally McRealguy
Race: orc
Gender: M
Level: 5
HP: 25
Strength: 20
Dexterity: 20
Constitution: 20
Intelligence: 20
Wisdom: 20
Charisma: 1
Role: Rogue
```

It works! And now my character is complete!

Bonus

- For your dice rolls, instead of generating a single value between 1 and 20, generate four values between 1 and 6, drop the lowest roll, and add them together.
    1. Go from rolling a sinlge 20-sided die to four 6-sided dice, keeping the top three rolls.
    - Ex: If you rolled 4, 5, 2, and 6, you should keep 4, 5, and 6.

2. Add these 3 together, saving this sum to your diceRolls array. This makes it impossible to roll something as low as a 1 or 2 in any stat.

- `job/SetJob`
  - Add a new attribute: job
    - string variable, default value is "Unemployed"
  - Modify ToString to Display job
  - Create new SetJob method
    - User picks jobs available to them based on stats
      - Strength: Greater than 10: Soldier; Greater than 15: Mercenary
      - Dexterity: Greater than 10: Armorer; Greater than 15: Blacksmith
      - Constitution: Greater than 10: Miner; Greater than 15: Gladiator
      - Intelligence: Greater than 10: Mathematician; Greater than 15: Scholar
      - Wisdom: Greater than 10: Historian; Greater than 15: Philosopher
      - Charisma: Greater than 10: Merchant; Greater than 15: Bartender
    - Array of all options should be set at the beginning (str, str, dex, dex, con, con, ...)
    - Only show jobs that are available to user (like `SetRole`)
    - User should pick job (with validation)

Solution

Project 2 Phase 3 Solution (LINK HERE)