CHARACTER CLASH!

FINAL PROJECT

Time to get creative and apply your programming and object oriented skills to a new context!

Your task is to create a software application that is similar to the Pokemon game. It should include a number of characters that can clash with each other, or against computer opponents, in battles. The winner of each battle will be determined by a number of things including each character's attributes, attacks and defenses.

Image of a wide variety of characters including a ghost, a shark, a bee, an alien and a robot.

You will use many of the concepts that have been introduced throughout the course to establish attributes for each character and to determine hit points and health status. You will use object oriented programming concepts to implement your program.

You will follow the Applied Design stages and begin by interviewing an end-user to determine their interests, then you will design a Pokemon-like game that they would enjoy playing. The end-user will be involved in the design and testing of the game.

Your software application should include: * A number of characters * A number of common attributes for each character * A variety of power levels for each attribute of a given character * A range of attacks that each character can perform * Formulas to determine the effectiveness of each character's attack in given situations (this may involve some randomness) * The ability for the user to list all data related to the characters
* The ability for the user to select and play the game as one of the characters * The ability for the user to select a variety of attacks and defenses * Game play that progresses the user through the game and determines whether or not the user defeats other characters to win the game, or is defeated and therefore loses.

You will need to carefully select your characters and a context in which they can do battle. This will require you to understand the interests of your end user, but it will also require you to be creative and to apply many of the concepts your have learnt throughout the course.

These characters or groups will be able to "battle" each other or they can work together to battle "computer opponents". The results of each battle will be dependent on the statistics of each of the characters or groups in the battle, as well as the attacks and defenses chosen by the user. You can also add some randomization into the battle as well.

The end-user will provide you context for the program by indicating their interests and likes/dislikes. The following list provides you with some quick ideas to help you start thinking of some possibilities: * A program that allows various bugs to battle each other, each bug has different attributes and strengths/weaknesses * A program that allows different basketball teams to compete against each other, each team has different strengths and weaknesses (three point shooting, defense, endurance). The game allows the user to play as a coach, selecting what to do in specific situations * A program that allows different African animals to battle for "Savanah Supremacy". * A program that simulates a number of different cars being able to race each other on different tracks (gravel, pavement, desert, rainy, etc). * A program that simulates a battle between vegetables, fighting to be recognized as the healthiest thing to eat. * A program that uses characters that you have made up yourself, with attributes, that battle for some type of championship. * A program that simulates a wide range of pets (cats, dogs, birds, lizards, etc) who are defending their owners home from a burglary or defending themselves from being petnapped!

You will have to be creative as you complete the program, and you will also have to carefully design and test the game play and logic of the battles. You want to ensure that the formulas for hit points and health make sense, and that the game includes an appropriate level of difficulty.

Example… The following is just an example to illustrate the various components of the program.

Talia is a student in the course. She interviews her little sister as the end-user of her game and determines that her little sister loves aliens. Talia brainstorms some ideas and presents the following idea to her sister:

A game that includes 8 aliens, one from each of the planets in the solar system. Each alien has five attributes including oozePower, hop-ability, laserShot, protectCharms and powerPunch. These attributes are rated on a scale of 1-5, with each character possessing a total score of 17.

As an example, one of Talia's characters is called MartyMarsian. He has the following levels for each of the attributes: * oozePower: 3 * hop-ability: 4 * laserShot: 5 * protectCharms: 2 * powerPunch: 3

Notice that the values add up to 17 and no value is above 5 or below 0. This is a design feature that Talia wanted to include in her game.

Talia meets with her little sister after the initial design stage and she loves Talia's ideas. They work together to determine how each character will do battle and decide that the user will get to select which alien they play as. They will then have to battle the seven other aliens to be crowned Champion of the Solar System. The alien will receive extra points after each victory and Talia will design formulas that make the game challenging, but winnable.

Image of the planets

Talia designs a superclass and subclasses for her aliens and creates formulas (that include some random values) to determine the amount of health lost in each battle. Talia creates user menus that allow the user to navigate the program and to allow the user to see and sort the characteristics of each alien.

Talia tests the program with friends and relatives to obtain further feedback and suggestions. She then implements a few changes and submits her finished project to her teacher!

Implementation Requirements

Complexity and Creativity

Your final project should be sufficiently complex and large-scale to push your limits as a programmer, but not so sophisticated that you are not able to complete it in the time allotted. The complexity in your project should come from the design and the algorithms and not from the code. (That is, you cannot meet the complexity requirement simply by writing a lot of code. Your code must be challenging or interesting in some meaningful way.) In addition, you should not add complexity by introducing peripheral elements, such as graphics or sound effects. (Your program can certainly have these, but they will not be considered in determining the projects complexity.) In addition, one of the main goals of this project is to allow you to unleash your creativity and allow you to create something of interest to you. To achieve this, your project must show some level of creativity or personalization that makes it your own. Simply creating your own version of some existing application will not fully meet this requirement. For both the complexity and creativity requirements, you should talk to the instructors early and often to ensure your project is in line with our expectations.

Documentation and Style

As with all previous projects, your program must be well-written, well-documented, and readable. Writing code with good style is always a good idea, but in a project of this size and scope, following style guidelines will help you keep your thoughts organized and make it easier to keep track of your progress, pick up where you left off each day, and find and fix bugs.

Resources: Throughout this course, you have completed a number of small programs that implemented a Pokemon game. You now have the opportunity to take these skills and apply them to a new context. Be sure to review course resources so that you fully understand how this type of game can be implemented. You can alter and change some of the ideas presented in the course to facilitate the completion of your own game. The following are just a few resources that you might want to review:

- 3.03 – Pokemon and returning values
- 3.06 Hitpoints and formulas
- 3.07 Pokemon Battle Program
- 3.14 Random Numbers
- Unit 5 Object Oriented Programming
- Unit 6 Inheritance and Polymorphism

These resources will help you understand how characters and attributes are implemented in Pokemon: * Basic Overview of the Game: http://www.pokemon.com/us/parents-guide/ * Advanced Individual Value & Stat Calculator: https://legendarypkmn.github.io/javacalc.html * Gameplay: https://youtu.be/DlEbXH8eUTk?t=1m26s * Types of Pokemon: http://www.pokemon.com/us/pokedex/ * Pokemon with Stats: http://tinyurl.com/no4mzic

STEP 1 – UNDERSTANDING CONTEXT

Conduct user-centered research to understand design opportunities and barriers.

Select an end-user for whom you will design and create this program (this can be a friend, classmate, relative, etc). Create interview questions that will allow you to understand the end-user's interests and likes/dislikes. Provide these questions to your teacher and then use them to interview your end-user.

Record all of the end-user's responses to your questions.

STEP 2 – DEFINING AND IDEATING

Choose a design opportunity and point of view, make inferences about premises and boundaries, take creative risks to identify gaps to explore, generate ideas to create a range of possibilities, prioritize ideas for prototyping and designing with users

Using the responses from your end-user interview, begin to develop a plan for your program. List: • the types of characters included in your program • their names • their attributes and relative "power" for each • their strengths and weaknesses • any other details that might be relevant for the characters • any formulas that will determine hit points and health values during a battle • a logical game play format

Share these ideas with your end user. Record their comments, suggestions and feedback and note any changes that you may make as a result of this interview.

Identify any issues or problems that might arise as you begin to program (what areas might require more information or programming solutions? Where can you find this information?)

STEP 3 – PROTOTYPING AND TESTING

Construct prototypes, making changes to code as needed.

Program your game. Be sure to review course notes and activities to make sure that you effectively implement object oriented programming design for your characters. Decide how battles will take place, how each character will have a chance to attack and defend, and how points will be awarded during battles. Be sure to program your game in small steps, ensuring that each step is fully functioning before adding complexity.

When you are ready, create a short test plan and test your program. Include expected output and actual output and include details related to any fixes that needed to be made.

Have your end-user try a working version of your program. Note their suggestions, comments and feedback. Indicate any changes that you made to the program, based on the user's feedback.

STEP 4 – SHARING, TESTING AND FINAL ITERATION

Gather feedback from users over time to critically evaluate design and make changes to product design or processes Identify new design issues

Share your work with classmates, friends or family. Record any feedback, suggestions and comments and use this information to make the final iteration of your program.

Grading Scheme/Rubric	
Implementation	
Project is appropriately complex and creative	4 points
Program is well-documented and shows good style	10 points
Final product meets all requirements and goals laid out in specifications	8 points
Program uses programming concepts effectively, including all required elements with an appropriate level of complexity	4 points

Grading Scheme/Rubric	
Object Oriented Programming concepts are effectively applied with an appropriate level of complexity	6 points
Game play is logical and effective	3 points
Formulas for damage/health are logical and effective	3 points
Checkpoint 1	2 points
Checkpoint 2	2 points
Checkpoint 3	2 points
TOTAL	44 points
—	—
Applied Design Steps and Log	
Understanding Context components are complete and thorough	3 points
Defining and ideating components are complete and thorough	3 points
Prototyping and testing components complete and thorough	2 points
Sharing, Testing and Final Iteration components are complete and thorough	2 points
TOTAL	10 points
—	—
TOTAL	54 points