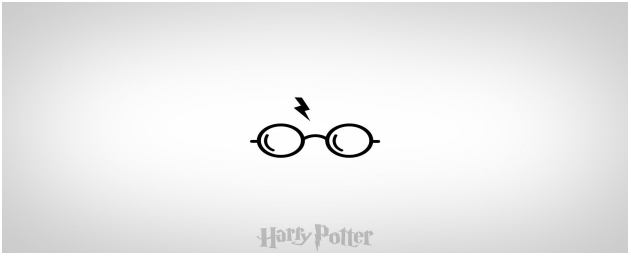
PROGRAMMING PROJECT
PROGRAM DEVELOPMENT
2019/20 Course

Computer Science Degree:

Software Engineering

Computer Engineering



Potter's Dare (PD)[1]

Delivery (Thursday, December 5th, 1:30 PM)

This document specifies the main actions that must be carried out in order to implement the project of the Program Development.

1.- Introduction

The objective of the project is the development of a system that allows doing a simulation based on the fantastic novels "*Harry Potter*"[2] saga of . In the case of the subject, we have used some names of characters, magic wands, potions, etc. But we have made our own adaptation and created a fictional context to adapt it to the contents of OOP taught in PD.

Thus, the project consists of simulating a *competition of duels* among the houses of the *Hogwarts* castle (initially *Gryffindor*, *Hufflepuff* and *Slytherin*) in which there will be characters that belong to each of them and that will compete in turns until there is a champion or the simulation reaches a maximum number of turns.

2.- Description of the entities of the "Potter's Dare (PD)" project.

2.1.- *Hogwarts*

It represents the castle where the characters compete.
Its main mission is to store information about houses, characters, duels and their results.
To do this, in each turn it will request to each house that sends one of its characters to maintain a duel with the characters of each of the other houses.

Once each character from each house arrive at Hogwarts, the duels will start. In these duels, the

[1] We thank the Student Council for providing the original idea for this project.
[2] https://www.wizardingworld.com

character with **less energy** (see Character) will be the first to face the rest of the characters **one by one.** Thus, the character with less energy will perform the action of attacking (fight) one by one the other characters. Once the first one have finished, the second one will also attack all the rivals and so on until all the characters have attacked once to the rest. Keep in mind that the duel will only occur if both characters ("attacker" and "attacked") still have energy (that is, a value greater than zero in that variable).
At the end of all attacks, only those characters that still have energy will come back to their respective houses.
Those without energy will be sent to a dungeon in the castle in order of arrival.
The dueling turns will be repeated until there is a winner or a maximum number of turns is reached. The specific details on this final situation are explained below in the detailed requirements.
In addition, at the end of the simulation, the application will display the information that summarizes the actions that have happened as well as statistical information of the characters that have participated.

Detailed requirements of *Hogwarts:*

- There will be a **single instance** of *Hogwarts* in the simulation.
- *Hogwarts* class will store the following minimum information:
    - A collection that stores the different houses of the simulation. This structure will allow efficient access to each house using its name (whose values will be unique) as the key.
    - The characters sent by each house to participate in the duel will be stored in a linear collection in which the characters will be sorted according to the criteria of **least to greatest** energy.
    - *Hogwarts* will store a collection of "new" magic wands. These wands can be of any of the types specified in the "Magic Wands" section and will be stored in a structure that allows them to be maintained in ascending order by the specific name assigned to each wand (not to be confused with its type). So, there may not be two wands with the same name. These wands are stored to be delivered to the each character (that still has enough energy) after her turn dueling while there are still wands stored. So, if the magic wands run out, the character will continue during the rest of the duels with the last wand she used.
    - In addition, after a turn of duels *Hogwarts* will not return the characters without energy to their house and, instead, will send them to the dungeon, that will be a linear collection in which the characters will be sorted in order of arrival.

Hogwarts **functionality** will be:

- In each turn, while the simulation does not reach the final situation:
    - Ask each house that they send a character.
    - Sort them from least to greatest energy.
    - Perform the duels:
        - First, the character with less energy will perform the action of attacking (fight) one by one the other characters in case both characters involved in the duel still have energy.
        - Continue with the rest of characters attacking all the rivals, one by one, in case both characters still have energy, until each character has used her turn of attacking.
    - At the end of the turn of duels:

        - It will send the characters that still have energy to their houses and those without energy to the dungeon.
        - Before sending the characters with energy to their house, and *Hogwarts* still has magic wands without delivering, *Hogwarts* will provide a new magic wand to the characters, in the same order that they are stored, that is, in ascending order by the wand name.
    - Display the results of each duel together with the rest of the requested information.
    - **The final situation happens when there is only one character with energy or 10 turns have been completed.**
- At the end of the simulation when the final situation is reached:
    - Display the final results.

2.2.- *House*

It represents the houses where the different characters are grouped so that each character must belong to one and only one of the houses. His main task is to hold the group of its characters and send each one of them in each turn to the duel in *Hogwarts*.

Detailed requirements of *House*:

- Each object of this type will store, at minimum, the different characters that belong to it. This structure will allow the **initial** ordering of the characters according to the criteria explained in the functionality section.

House **functionality** will be:

- **Before the beginning of the first turn of duels**, each house will sort its characters. This **initial** criteria order will be:
    - House *Gryffindor*: it will sort their characters from less to more defensive points.
    - House *Hufflepuff*: it will sort their characters from less to more offensive points.
    - House *Slytherin*: it will sort their characters from more to less offensive points.
- When *Hogwarts* asks for a character to a house, it will send the character in the first position for the duel.
- The characters sent to the duels that later return to the house will be inserted in the last position of the character collection.

2.3- *Character*

This class defines the common features of the characters of the simulation that are explained below.

Detailed requirements of *Character:*

- Each object of this type will store, at least:
    - Name.
    - *Energy* points (20.0 by default)
    - *Offensive* points (20.0 by default)
    - *Defensive* points (20.0 by default)
    - *Magic Wand*:
        - Any character may use any *Magic Wand*.
        - The characters should be able to change their *Magic Wands*.
- Moreover, the characters are divided into the next types:
    - *Defensives*: they have 5.0 extra *defensive* points.
    - *Offensives*: they have 5.0 extra *offensive* points.

Characters **functionality** will be:

- Each character will provide, at least, the next methods:
    - *fight* method: it will be used during the duels. This method will check the *attack points* returned by the Magic Wand for the character using it in an offensively way and it also will check the *resistance points* returned by the Magic Wand for the opponent using it in an defensively way and then:
        - if the attack points of the character are greater than the resistance points of the opponent (*attack points* > *resistance points*), the *energy* of the opponent after the attack will be the result of subtracting to the energy of the opponent before the attack the difference between the *attack points* and the *resistance points* .
        - if the attack points of the character are lesser or equal than the resistance points of the opponent (*attack points* <= *resistance points*), the energy of the opponent remains the same.
    - r*esistance* method: it will be used by the opponent to counteract the attack. It will return the *resistance points* returned by the Magic Wand for the character using it in a defensively way.
    - *Change Wand* method. THe method used by *Hogwarts* after a turn of duels to change the wand of the character before sending her to her house (if there are still wands unassigned).
        - For instance, if a character has fighted during a turn using the *HollyDefensive Wand,* Hogwarts could assign her a *HawthornOffensive Wand.*
    - In addition, each character will also provide the rest of methods needed to carry out the functionality required to accomplish the simulation ("*gets" methods*, "*sets" methods*, etc.).

2.4- *Magic Wands*

Each character uses a Magic Wand during the duels to generate her *attack points* as well as her *resistance points*. The difference between each wand depends on the kind of wood used that produces a different result if they are used in an offensively way or in a defensively way.

Detailed requirements of *Wand:*

- A wand is used during the duels by the characters both in an offensively way, returning *attack points,* and in a defensively way, returning *resistance points*.

Character **functionality** depends on the kind of wood:

- Initially, there will be the next kind of Magic Wands with their different behaviour:
    - *OffensiveHawthorn*: it is a kind of wand focused on attacking. So, when a character use this wand:
        - In a offensive way: the *attack points* returned by this wand will be: 0.6 times her *energy* points plus 1.4 times her *offensive* points;
        - In a defensive way: the *resistance points* returned by this wand will be: 0.8 times her *energy* points plus 0.2 times her *defensive* points;
    - *HollyDefensive*: it is a kind of wand focused on defending. So, when a character use this wand:
        - In a offensive way: the *attack points* returned by this wand will be: 0.7 times

her *energy* points plus 0.3 times her *offensive* points;
    - In a defensive way: the *resistance points* returned by this wand will be: 0.9 times her *energy* points plus 1.1 times her *defensive* points;

2.5- *Extra potions*

Any character could drink different extra potions along the competition. These potions increase the offensive points and/or the defensive points of the characters who drink them. A character could not use ever one of these potions, use one of them, use two of them or even use all of them during the same competition.
So, initially, there will be the next kind of extra potions:

- *InvigorationPotion*: it is a kind of wand focused on defending. So, it increases 1.4 times the *defensive points* of the character who drinks it.
- *FelixFelicisPotion*: it is a kind of wand focused on attacking. So, it increases 1.3 times the *offensive points* of the character who drinks it.

3.- Requirements and Assessment

This section explains the **minimum requirements** and the **extra requirements of the project,** together with the **Assessment** of the project.
In case **the project does not meet any of the minimum requirements, the grade of the project will be a 3 at most** without the possibility of increasing this grade implementing the extra requirements.
In case the project does meet <u>completely</u> the minimum requirements, the grade will be a 5 at least with the possibility of **implementing extra requirements that allow getting 5 extra points** in the final grade of the project.

a. Minimum requirements

- The project **must provide the functionality detailed in the previous sections** of this document. To do this, it should be possible to check that everything works well using the ***"HogwartsDemo"*** class provided in the initial ***"PottersDare"*** project (and using the ***"InitData"*** class to load the initial data) or a similar class developed by each group. So:
    - The system always works with an adequate performance with respect to time and use of resources.
    - The system is able to manages all possible malfunction, notifying adequately to the user what was its cause.
- To check the output of each group, initial data is provided to be loaded and a output will be created that must be called **<u>output.log</u>** and must **<u>strictly follow (in English) the format</u>** . Both the initial data to do the simulation and the template of the output.log file are provided in Annex I.
- The execution of the simulation will generate an output file called **<u>output.log</u>**. This file must **<u>strictly follow (in English) the specific template shown in Annex I</u>**.
    - In addition, in Annex I, an example in pseudocode of possible initial data is shown together with the output.log file that this initial data would produce following this template for the first turn and for the last turn of the simulation (The complete output

is shown in the file output.log, located in the project folder of the subject).
    - The project delivered by each group should be able to work with any set of initial data and generate the appropriate output.
    - In a few days, the teachers will provide a second set of initial data:
        - The output generated by this second set of data will be the one that each group will have to include in the delivery of the project.
        - This second set of initial data will be the one taken as a starting point for the defense of the project.
- The **terminal output will have the same format as the output.log file**.
- The project should run from the beginning to the end **without needing to ask for user intervention**.
- **Good code style.** The source code meets all the recommendations and specifications explained in the subject, such as:
    - Correct indentation of all the source code.
    - Use of an unique style in the name of classes, variables and methods as well as in comments, file names, loops, specification of conditions and initialization of variables.
    - The different iteration structures are used appropriately. Thus, a for loop is only used to go through the whole collection whereas while loop is used to searches that could end before reach the last element of the collection or for the removal of elements from a Collection (in this case in conjunction with Iterator).
- **Correct application of the concepts class, encapsulation and composition**:
    - The required classes for the project are well defined, specifying adequately their public functional interface..
    - All classes correctly apply the encapsulation principle without exposing details of implementation of their data in their public functional interface.
    - Composition is properly used in the definition of project classes.
- **Correct application of the concepts inheritance and polymorphism:**
    - All class hierarchies required for the project are defined correctly.
    - The public functional interface of the hierarchy's base class is correctly defined in order to take advantage of polymorphism.
    - All polymorphic algorithms and all polymorphic data structures needed by the project are defined, making the most of polymorphism and avoiding in all cases the repetition of code
- **Correct use of Collections: List, Set y Map**:
    - They will be used correctly, together with the methods they provide, **at least once each**, using the most appropriate in each case, depending on their characteristics.
- **Correct application of the Singleton pattern** in the appropriate class according to the characteristics of this pattern.
- Generation of the **external documentation** of each class:
    - with the explanation of all its methods (and its inputs and outputs), by means of the correct use of JavaDoc functionality, that is, correctly including in the internal project documentation the corresponding JavaDoc annotations (@param, @return, etc.) without the system returning any type of error in its generation.

b. Extra requirements

As additional contributions to the delivery, each group could do:

- **JUnit tests** (up to **2 points**)**:**
  - A whole set of tests has been developed for each of the classes of the project, providing, at least, a functional test for each public member method of the class..
  - That is, these tests will contain a test method together with its corresponding assertion / s for each method of the class to be tested.
  - In addition, the creation of the object / s necessary for the tests must be included in the *setUp()* method of the test class and not repeated continuously within each method.
- **Application of more design patterns apart from the Singleton pattern** indicated in the minimum requirements (up to **2 points**)**:**
  - They must be applied correctly and appropriately.
  - This requirement will only be taken into account if no anti-pattern is detected in the rest of the project.
- **Correct use of comparator libraries** (up to **0,5 points**)**:**
  - Adequate use is made of the definition of comparators for customization of sorting behavior.
- **Correct use of Iterator** (up to **0,5 points**):
  - It will be used correctly, **at least once**, in the functionality appropriate to its characteristics, together with the methods it provides..
- **Correct use of exceptions handling** (up to **0,5 points**)**:**
  - Exceptions are used correctly, so that the correct operation of all project routines is ensured.

4.- About the format of the delivery and its deadline

Each group of students should **deliver their project before Thursday, December 5th at 13:30**. The source code of the delivery **will exclusively contain the needed code to meet the requirements of the project** and nothing else. In order to deliver the project, **a task on the Virtual Classroom will be enabled**.

Code requirements:

- In addition to the important consideration of "´**Minimum Requirements**", the students also must take into account:
  - The project should be opened by BlueJ 4.2.1 or later with Java 11 without any extra action is needed in order to run the project and its tests.
  - The .TXT file with the summary of the project and its main class as well as each of the source code files of the project must include at the beginning:
    Group name:
    Students' First Names and Surnames:

The folder containing the project must be delivered as a **.zip compressed file.** The name of the file will be composed of the surname and name of each member of the group separated by "_". For instance, the file for a three student group would be:
"ec_surnamestudent1_namestudent1_surnamestudent2_namestudent2_ec_surnamestudent3_na

mestudent3.zip”.
The file must be delivered by <u>ALL THE STUDENTS</u> of the group.

Annex I.

Template of output.log file

As specified, once the initial data is loaded, a maximum number of turns will be carried out in the simulation. The template of the information shown in each turn, that will also be stored in a file called **output.log**, is as follows:

houses:

(for each house in hogwarts)
house:<name>
(for each character in the house)
character:<name> <e: energy> <o: offense><d: defense> <wand: wandName (wandType)>

new wands:

(for each new wand in hogwarts)
wand:<name (wandType)>

(for each turn)
turn:<turnNumber>

characters who are going to duel:

(for each character in hogwarts (fighters))
character:<name> <e: energy> <o: offense><d: defense> <wand: wandName (wandType)>

(for each duel in this turn)
duels:

(for each duel between characters)
<*name char1*> *is dueling against* <*name char2*>
attack points of <*name char1*> are: <attack points>
resistance points of <*name char2*> are: <resistance points>
the remaining energy of <*name char2*> after the duel are: <energy>

(at the end of duels in this turn)
duel results:
---
character:<name> <e: energy> <o: offense><d: defense> <wand: wandName (wandType)> [returns to the house] [goes to dungeon]
[new wand assigned: <wandName (wandType)>]

```
(at the end of the simulation)
end of the simulation:
-------------------------

houses:
-----------
(for each house in hogwarts)
house:<name>
(for each character in the house)
character:<name> <e: energy> <o: offense><d: defense> <wand: wandName (wandType)>

new wands:
---------------
(for each new wand in hogwarts)
wand:<name  (wandType)>

dungeon characters:
-------------------------
(for each character in the dungeon)
character:<name> <e: energy> <o: offense><d: defense> <wand: wandName (wandType)>

(winner house (house with the largest number of characters))
the winner house is:
-----------------------
house:<name>
(for each character in the house)
character:<name> <e: energy> <o: offense><d: defense> <wand: wandName (wandType)>
```

Example of Initial Data

As an example of initial data in the simulation, the InitData and the HogwartsDemo classes are shown below (in pseudocode). The first one will load the initial data and the second one will contain the main method of the simulation. The pseudocode shown in both classes should be adapted to the implementation of each group.

Remember that the output.log file provided in this document corresponds to the output generated for this initial data, but the project delivered by each group will be adapted to any set of initial data to generate the appropriate output.

```
/**
 * Class to init data of the simulation.
 * It could be used alto to write finals results.
 *
 * @author profesores DP
 * @version 19/20
```

```
 */
public class InitData
{
    private Hogwarts hogwarts = … ////hogwarts must be a unique instance

    /**
     * Constructor for objects of class InitData
     */
    public InitData()
    {
        initData();
    }

     private void initData()
    {
        //New wands
        //-----------------------
        hogwarts.newWand(new HollyWandBehavior("Larch") … );
        hogwarts.newWand(new HollyWandBehavior("Rowan") … );
        hogwarts.newWand(new HawthornWandBehavior("Hazel") … );
        hogwarts.newWand(new HollyWandBehavior("Tamarack") … );
        hogwarts.newWand(new HawthornWandBehavior("Spruce") … );
        hogwarts.newWand(new HawthornWandBehavior("Laurel") … );


    //Griffindor's characters
    //----------------------------
    House gryffindor = new House("Gryffindor" … );
    Character harry = new DefensiveCharacter("Harry Potter"  … );
        ->assign Wand (name: “HarryW”, type: “HollyWandBehavior”) to harry
        ->assign  InvigorationPotion to harry
        Character hermione = new OffensiveCharacter("Hermione Granger"  … );
        ->assign  InvigorationPotion to hermione
        Character ron = new DefensiveCharacter("Ron Weasley" … );
        ->assign Wand (name: “RonW”, type: “HawthornWandBehavior”) to ron
        ->assign  FelixFelicisExtraPotion to ron
        Character neville = new OffensiveCharacter("Neville Longbottom" … );
         gryffindor.addMember(harry);
         gryffindor.addMember(hermione);
         gryffindor.addMember(ron);
         gryffindor.addMember(neville);
         ->sort gryffindor’s members
         hogwarts.newHouse( gryffindor);

         //Hufflepuff's characters
         //----------------------------
         House hufflepuff = new House("Hufflepuff" … );
```

```
        Character cedric = new OffensiveCharacter("Cedric Diggory"  … );
       ->assign  FelixFelicisExtraPotion to cedric
        Character nymphadora = new DefensiveCharacter("Nymphadora Tonks" … );
        Character pomona = new OffensiveCharacter("Pomona Sprout" … );
       ->assign Wand (name: “PomonaW”, type: “HollyWandBehavior”) to pomona
       ->assign  FelixFelicisExtraPotion to pomona
        Character rolf = new DefensiveCharacter("Rolf Scamander" … );
        hufflepuff.addMember(cedric);
        hufflepuff.addMember(nymphadora);
        hufflepuff.addMember(pomona);
        hufflepuff.addMember(rolf);
        hufflepuff.sortMembers();
        ->sort hufflepuff’s members
        hogwarts.newHouse(hufflepuff);

        //Slytherin's characters
        //----------------------
        House slytherin = new House("Slytherin" … );
        Character draco = new OffensiveCharacter("Draco Malfoy" … );
       ->assign  FelixFelicisExtraPotion to draco
        Character dolores = new DefensiveCharacter("Dolores Umbridge" … );
        Character pansy = new OffensiveCharacter("Pansy Parkinson" … );
       ->assign Wand (name: “PansyW”, type: “HawthornWandBehavior”) to pansy
       ->assign  FelixFelicisExtraPotion to pansy
        Character albus = new DefensiveCharacter("Albus Severus Potter" … );
       ->assign Wand (name: “AlbusW”, type: “HollyWandBehavior”) to albus
        slytherin.addMember(draco);
        slytherin.addMember(dolores);
        slytherin.addMember(pansy);
        slytherin.addMember(albus);
        ->sort slytherin’s members
        hogwarts.newHouse(slytherin);
    }

}


/**
 * Main simulation class.
 * First, initial data are loaded.
 * Then, while is not the end of the simulation, the actions will be performed.
 *
 * @author profesores DP
 * @version 19/20
 */
public class HogwartsDemo
{
```

```
  public static void main(String[] args) {
      int turnNumber = … ;  //define the current turn number
      int totalTurns = … ;    //define the total number of turns
      boolean endSimulation = false;   //define the end of the simulation
      InitData initdata = new InitData(); //init data of the simulation

      //this pseudocode is not a fixed version, it can be modified
      ->call to method that shows the initial information
      while(condition to continue in the simulation) {
         System.out.println("\nturn: <"+turnNumber+">");
         -> call to method that obtain the characters from the houses
         -> call to method that performs the duel in this turn
         turnNumber++;
      }
      -> call to method that write the final results
   }
}
```

Example of output

Given the initial data shown in the previous section, the output that those initial data generate (in the format of the output.log file) are shown below for the initial turn and for the final turn of the simulation. The complete output is shown in the file output.log, located in the project folder of the subject.

```
houses:
----------
house:<Hufflepuff>
character:<Cedric Diggory> <e: 20.0> <o: 32.5> <d: 20.0> <wand: CedricW (HollyWandBehavior)>
character:<Pomona Sprout> <e: 20.0> <o: 32.5> <d: 20.0> <wand: PomonaW (HollyWandBehavior)>
character:<Nymphadora Tonks> <e: 20.0> <o: 20.0> <d: 25.0> <wand: NymphadoraW (HollyWandBehavior)>
character:<Rolf Scamander> <e: 20.0> <o: 20.0> <d: 25.0> <wand: RolfW (HawthornWandBehavior)>

house:<Slytherin>
character:<Dolores Umbridge> <e: 20.0> <o: 20.0> <d: 25.0> <wand: DoloresW (HollyWandBehavior)>
character:<Albus Severus Potter> <e: 20.0> <o: 20.0> <d: 25.0> <wand: AlbusW (HollyWandBehavior)>
character:<Pansy Parkinson> <e: 20.0> <o: 32.5> <d: 20.0> <wand: PansyW
```

(HawthornWandBehavior)>
character:<Draco Malfoy> <e: 20.0> <o: 32.5> <d: 20.0> <wand: DracoW (HawthornWandBehavior)>

house:<Gryffindor>
character:<Harry Potter> <e: 20.0> <o: 20.0> <d: 35.0> <wand: HarryW (HollyWandBehavior)>
character:<Hermione Granger> <e: 20.0> <o: 25.0> <d: 28.0> <wand: HermioneW (HawthornWandBehavior)>
character:<Ron Weasley> <e: 20.0> <o: 26.0> <d: 25.0> <wand: RonW (HawthornWandBehavior)>
character:<Neville Longbottom> <e: 20.0> <o: 25.0> <d: 20.0> <wand: NevilleW (HollyWandBehavior)>

new wands:
---
wand: <Hazel (HawthornWandBehavior)>
wand: <Larch (HollyWandBehavior)>
wand: <Laurel (HawthornWandBehavior)>
wand: <Rowan (HollyWandBehavior)>
wand: <Spruce (HawthornWandBehavior)>
wand: <Tamarack (HollyWandBehavior)>

turn: <0>
characters who are going to duel:
---
character:<Cedric Diggory> <e: 20.0> <o: 32.5> <d: 20.0> <wand: CedricW (HollyWandBehavior)>
character:<Dolores Umbridge> <e: 20.0> <o: 20.0> <d: 25.0> <wand: DoloresW (HollyWandBehavior)>
character:<Harry Potter> <e: 20.0> <o: 20.0> <d: 35.0> <wand: HarryW (HollyWandBehavior)>

duels:

<Cedric Diggory> is dueling against <Dolores Umbridge>
attack points of <Cedric Diggory> are: <23.75>
resistance points of <Dolores Umbridge> are: <45.5>
the remaining energy of <Dolores Umbridge> after the duel are: <20.0>

<Cedric Diggory> is dueling against <Harry Potter>
attack points of <Cedric Diggory> are: <23.75>
resistance points of <Harry Potter> are: <56.5>
the remaining energy of <Harry Potter> after the duel are: <20.0>

<Dolores Umbridge> is dueling against <Harry Potter>
attack points of <Dolores Umbridge> are: <20.0>
resistance points of <Harry Potter> are: <56.5>

```
the remaining energy of <Harry Potter> after the duel are: <20.0>

<Dolores Umbridge> is dueling against <Cedric Diggory>
attack points of <Dolores Umbridge> are: <20.0>
resistance points of <Cedric Diggory> are: <40.0>
the remaining energy of <Cedric Diggory> after the duel are: <20.0>

<Harry Potter> is dueling against <Cedric Diggory>
attack points of <Harry Potter> are: <20.0>
resistance points of <Cedric Diggory> are: <40.0>
the remaining energy of <Cedric Diggory> after the duel are: <20.0>

<Harry Potter> is dueling against <Dolores Umbridge>
attack points of <Harry Potter> are: <20.0>
resistance points of <Dolores Umbridge> are: <45.5>
the remaining energy of <Dolores Umbridge> after the duel are: <20.0>


duel results:
------------
character:<Cedric Diggory> <e: 20.0> <o: 32.5> <d: 20.0> <wand: CedricW (HollyWandBehavior)> returns to the house
new wand assigned: <Hazel (class HawthornWandBehavior)>
character:<Dolores Umbridge> <e: 20.0> <o: 20.0> <d: 25.0> <wand: DoloresW (HollyWandBehavior)> returns to the house
new wand assigned: <Larch (class HollyWandBehavior)>
character:<Harry Potter> <e: 20.0> <o: 20.0> <d: 35.0> <wand: HarryW (HollyWandBehavior)> returns to the house
new wand assigned: <Laurel (class HawthornWandBehavior)>

…

turn: <9>
characters who are going to duel:
-----------------------------------------
character:<Hermione Granger> <e: 11.0> <o: 25.0> <d: 28.0> <wand: Tamarack (HollyWandBehavior)>
character:<Albus Severus Potter> <e: 15.0> <o: 20.0> <d: 25.0> <wand: Spruce (HawthornWandBehavior)>
character:<Cedric Diggory> <e: 20.0> <o: 32.5> <d: 20.0> <wand: Hazel (HawthornWandBehavior)>


duels:
-------
<Hermione Granger> is dueling against <Albus Severus Potter>
attack points of <Hermione Granger> are: <15.2>
```

resistance points of <Albus Severus Potter> are: <17.0>
the remaining energy of <Albus Severus Potter> after the duel are: <15.0>

<Hermione Granger> is dueling against <Cedric Diggory>
attack points of <Hermione Granger> are: <15.2>
resistance points of <Cedric Diggory> are: <20.0>
the remaining energy of <Cedric Diggory> after the duel are: <20.0>

<Albus Severus Potter> is dueling against <Cedric Diggory>
attack points of <Albus Severus Potter> are: <37.0>
resistance points of <Cedric Diggory> are: <20.0>
the remaining energy of <Cedric Diggory> after the duel are: <3.0>

<Albus Severus Potter> is dueling against <Hermione Granger>
attack points of <Albus Severus Potter> are: <37.0>
resistance points of <Hermione Granger> are: <40.7>
the remaining energy of <Hermione Granger> after the duel are: <11.0>

<Cedric Diggory> is dueling against <Hermione Granger>
attack points of <Cedric Diggory> are: <47.3>
resistance points of <Hermione Granger> are: <40.7>
the remaining energy of <Hermione Granger> after the duel are: <4.0>

<Cedric Diggory> is dueling against <Albus Severus Potter>
attack points of <Cedric Diggory> are: <47.3>
resistance points of <Albus Severus Potter> are: <17.0>
the remaining energy of <Albus Severus Potter> after the duel are: <0.0>

duel results:
---
character:<Cedric Diggory> <e: 3.0> <o: 32.5> <d: 20.0> <wand: Hazel (HawthornWandBehavior)> returns to the house
character:<Albus Severus Potter> <e: 0.0> <o: 20.0> <d: 25.0> <wand: Spruce (HawthornWandBehavior)> goes to dungeon
character:<Hermione Granger> <e: 4.0> <o: 25.0> <d: 28.0> <wand: Tamarack (HollyWandBehavior)> returns to the house

end of the simulation:
---
houses:
---
house:<Hufflepuff>
character:<Pomona Sprout> <e: 7.0> <o: 32.5> <d: 20.0> <wand: Rowan (HollyWandBehavior)>
character:<Cedric Diggory> <e: 3.0> <o: 32.5> <d: 20.0> <wand: Hazel (HawthornWandBehavior)>

house:<Slytherin>
character:<Pansy Parkinson> <e: 20.0> <o: 32.5> <d: 20.0> <wand: PansyW (HawthornWandBehavior)>
character:<Dolores Umbridge> <e: 8.0> <o: 20.0> <d: 25.0> <wand: Larch (HollyWandBehavior)>

house:<Gryffindor>
character:<Hermione Granger> <e: 4.0> <o: 25.0> <d: 28.0> <wand: Tamarack (HollyWandBehavior)>

new wands:
---

dungeon characters:
---

character:<Ron Weasley> <e: 0.0> <o: 26.0> <d: 25.0> <wand: RonW (HawthornWandBehavior)>
character:<Rolf Scamander> <e: 0.0> <o: 20.0> <d: 25.0> <wand: RolfW (HawthornWandBehavior)>
character:<Harry Potter> <e: 0.0> <o: 20.0> <d: 35.0> <wand: Laurel (HawthornWandBehavior)>
character:<Nymphadora Tonks> <e: 0.0> <o: 20.0> <d: 25.0> <wand: NymphadoraW (HollyWandBehavior)>
character:<Neville Longbottom> <e: 0.0> <o: 25.0> <d: 20.0> <wand: NevilleW (HollyWandBehavior)>
character:<Draco Malfoy> <e: 0.0> <o: 32.5> <d: 20.0> <wand: DracoW (HawthornWandBehavior)>
character:<Albus Severus Potter> <e: 0.0> <o: 20.0> <d: 25.0> <wand: Spruce (HawthornWandBehavior)>

the winner house is:
---
house:<Hufflepuff>
character:<Pomona Sprout> <e: 7.0> <o: 32.5> <d: 20.0> <wand: Rowan (HollyWandBehavior)>
character:<Cedric Diggory> <e: 3.0> <o: 32.5> <d: 20.0> <wand: Hazel (HawthornWandBehavior)>