CS162-400

Assignment 4- Polymorphism + Linked Lists Tournament

Step 1: Identifying Classes and Objects — Finding the nouns:

Describe the problem: Building off of our previous project of creature vs. creature, we now need to create a full tournament environment. We need to use linked lists, using both stacks and queues to hold creature objects, and store them appropriately. Also, I will be adding a function that allows the winner of each battle to have the possibility to regain strength when placed back into the lineup (FAQ*). In addition, I will keep track of points each team generates in a fashion other than win/loss (FAQ*). The first, second, and third place creatures will be displayed with their corresponding team. The user will be able to determine how many, and what type of creatures will be in each teams lineup.

FAQ:

How do you rank player and team scores?: I will base a scoring system, both for individual characters and teams, based upon research I completed in the previous project. Here is my previous graph:

	Reptile	Barbarian	Goblin	The Shadow	Blue Man	Victories Out of 100
Reptile	☑	R:10 B:00	R:10 G:00	R:08 S:02	R:01 Bl:09	29
Barbarian		☑	B:06 G:04	B:00 S:10	B:00 Bl:10	6
Goblin			☑	G:01 S:09	G:00 Bl:10	5
The Shadow				☑	S:02 Bl:08	23
Blue Man					☑	37

I have decided to determine points based on the specific creature lineup, and opponents difficulty. Minimum points (10) and Max is (60). If creature won 10 out of 10 battles against certain opponent, then if they beat that opponent in lineup they will only earn 10 points. Points earned= (10 + (10-wins)*5). A matchup vs. a similar creature will get 35pts in all cases.

POINTS EARNED!!!!!!	Reptile	Barbarian	Goblin	The Shadow	Blue Man
Reptile	35pts	Reptile:10pts Barbarian:60pts	Reptile:10pts Goblin:60pts	Reptile:10pts The Shadow:60	Reptile:55pts Blue Man:15pts
Barbarian		35pts	Barbarian:30 Goblin:40	Barbarian:60 The Shadow:10	Barbarian:60 Blue Man:10
Goblin			35pts	Goblin:60 The Shadow:10	Goblin:60 Blue Man:10
The Shadow				35pts	The Shadow:50 Blue Man:20
Blue Man					35pts

When a creature wins and is put in the back of the lineup, does he recover (how?):

Yes. A creature recovers up to 20% (rounded down by integer values) of their strength points (if and only if) they have lost at least 20% of their strength points. This will occur after the battle has ended, and essentially as the winner is being added to the back of the lineup.

How are the lineup's stored? How about the loser list?:

The tournament lineup is stored in a FIFO (queue) linked list. There will be two of these lists, one to represent each team. This is only a singly linked list. The winner will be added to the back, and removed from the front. A loser will be added to a FILO (stack), that keeps track of all the losers.

What method do you use to gather information about the winners? How do you know who is on what team? Can there be ties?:

A team score is kept track by a variable in the battleField class. Namely two integer variables called team1 and team2. They will be tallied up with the result of corresponding battles. When the tourney is over, accessing these variables directly should give us direct result of who won or lost (or if there is a tie).

As for individual successors, I will display a list of all character's and their scores. This will be done by storing creatures into a dynamically created array. I will then implement section sort based upon the individual creatures strength and sort them inside of the array. As creatures are placed into the dynamically created array, they will be removed from the linked list (queue or stack). There may in fact be a tie, but the ranking should be displayed for all creatures, and the "top three" or top three scores can be counted as the first second and third place creatures. I imagined a top score type list you see on an arcade game, so I will allow for all of our fighter's score to be displayed.

Will you be able to see the current "batting order":

Yes! I have decided to create recursive functions in the classes queue and stack. These functions will give you an up-to-date list of the lineup of each team's lineup. I will display these lists, as well as the loser stack list, after the end of a match. The creatures will have been moved around corresponding to the result of the match, and then displayed. It sounded like a good challenge to make this recursive.

I have drawn out what I believe to be a program that will allow all of what needs to be accomplished. Let me discuss my approach to major design implementation:

New classes: We are bringing in two classes that represent a stack and queue of singly linked lists. We will need to slightly modify them, and they will have composition of the creature class. They will contain a pointer to the abstract creature class. The peek and getFront functions will need to return a creature* now also. We will be able to utilize the creature's stored in list, by assigning a temporary creature pointer to the value, and return it back to the list when we are done.

New variables needed:

— **Creature class variables**:

- private: string person — This will let user name each individual creatures names in their lineup. This allows for more personal feel and also gives a better picture of who wins in the end. Now, you could have a lineup of all the same creature type, but give them separate names so they may be distinguished as the game progresses, and at the scoreboard at the end.

- private: bool tag- I believe a tag variable will come in handy when completing the scoreboard. This way, we can cycle through the creatures and find one that has the highest score. If they do, they can be tagged so they are not re-entered into the score board.

- private: string team- This will assign a creature a team name when they are initialized.

- private: int initstr — Stands for initial strength, stores what the creatures initial strength is to use in restore function.

- private: int points — Give's points to individual creature.

—**battleField class variables**:

- std::string name — This will allow us to store, validate, and send a name to the creature class upon adding creatures to the lineup.

- private: cretque team1, team2 — Two queue classes are initiated to store the creatures for each team.

- private: creatstk loserpile — Creates a stack for the loser pile.

- private: std::string t1name, t2name — We will add these variables to the constructor of battleField, and initialize them to be either "Team 1" or "Team 2".

- private: int points — Stores how many points to give to creature and teams.
- private: std::string cr1, cr2 — This will get the names from the creatures after they battle, and compare them in order to determine how many points are earned int he function getpoints();.
- private: int team1p, team2p; — Keeps track to points earned by teams.
- private: exit3 — Allows for another exit option

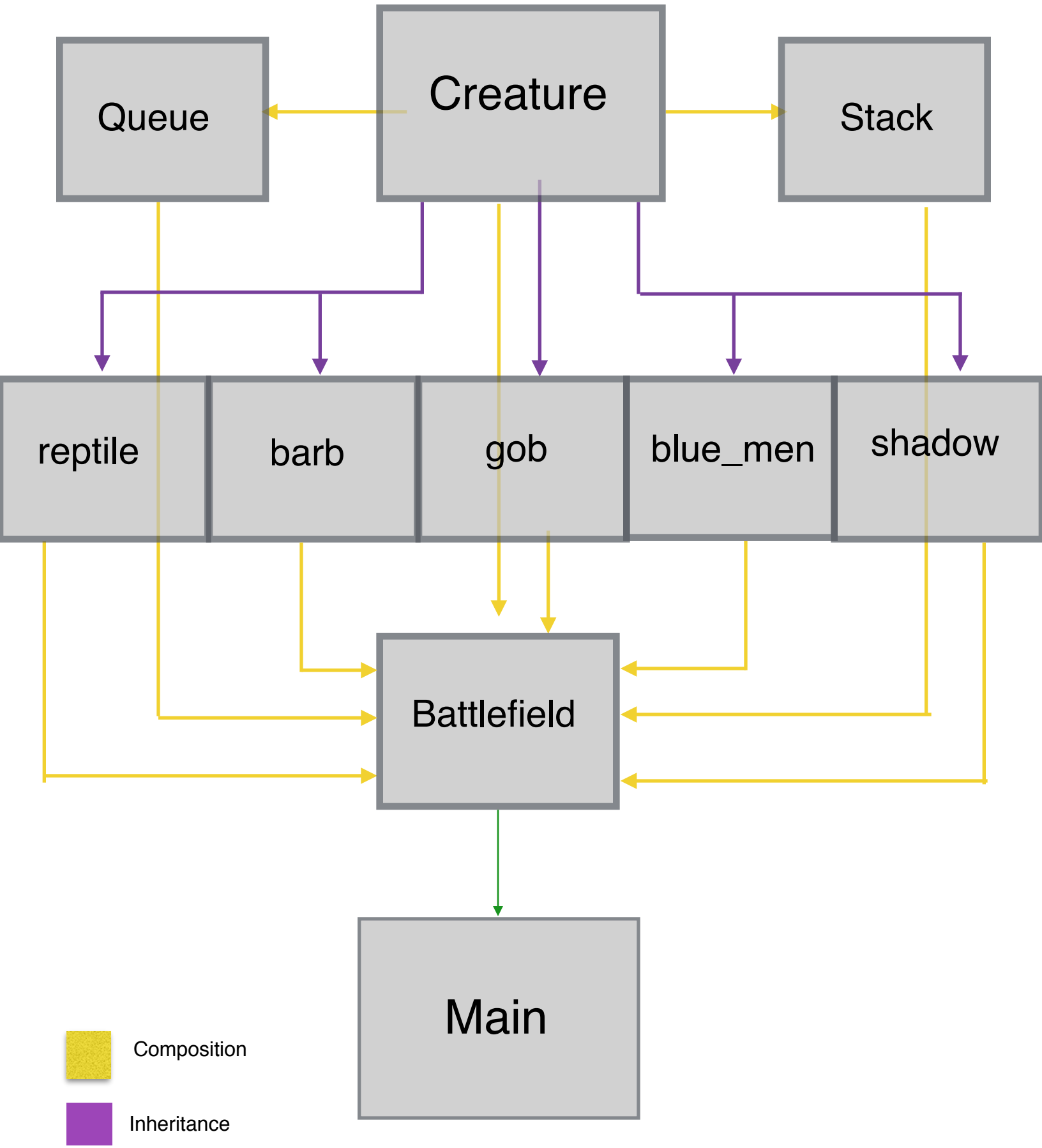
New function needed:

—**Creature class functions**:

- void restore(); — As stated in the FAQ*, the winner will gain a 20% regain in health if they are down by that much. This function will accomplish that.
- std::string getPerson(); — Returns the personal name given to individual creature.
- std::string getTeam(); — Returns the team of character.
- (We will need to add team and person in the constructor of the creature class).
- void setnameteam(); — Allows the private member variables of creature (person and team) to be set.

—**battleField class functions**:

- bool validateString(); —
- void choose(std::string); — I have decided to do some general clean up in the mainMenu, and divert larger pieces of code to their own functions. The choose function will let users choose which creature they want .
- void how_many(); — User selects how many character's they want in their lineup. Stored in int count variable.
- void onceagain(); — To do some more clean up work, I will move the option to play again in it's own function.
- int getpoints(); — Returns an integer value of stored points.
- void setpoints(); — Sets the creature variable int points.
-
- void reset(); — Resets all variables for next run through.


Queue
Creature
Stack
reptile
barb
gob
blue_men
shadow
Battlefield
Main
Composition
Inheritance

Final Considerations — Modifications — Notes:

- **CHANGE:** Added creature *fighter2 variable to the battlefield class in order to store fighters after battles.
- **CHANGE:** I did not anticipate using selectionSort to sort out all of the creatures in the end, to display their rankings. I did realize quickly that many creatures would in fact 'tie', and only displaying the top three would be a bit deceiving. I decided to convert that creatures into an array, and then to sort that array using selection sort. To do this I made a pointer to a pointer of type creature in the battleField class. I had to add variable called two count, which takes the variable count*2, so we can create a dynamic array the size of the total number of creatures.
- **CHANGE:** I went ahead and pulled out the unix function sleep, that created a delay in the program. While this did give it a more realistic gaming type feel, it took too long. With a game of only randomness, the excitement goes away rather quickly. I still left in the forced user input to continue in order to slow things up a bit. Feel free to power through a tourney by pressing 1 enter 1 enter 1 enter….., for testing purposes.
- **CHANGE:** I added a static function in order to change the status of the goblin special. The special now gets reset after each match.
- **CHANGE**: At first, working off my previous program, I took out the random roll at the beginning to see who goes first. I should have taken the time to incorporate this into my new design because I found it unfair to let Team1 have the first attack every turn. I had to go back and change a number of things in order to make the order random. Still, a worthwhile addition.