Contents

[Algorithm: 1](#_Toc19046793)

[Generalized-character-class characteristics: 2](#_Toc19046794)

[Generalized-tool-class characteristics: 3](#_Toc19046795)

[Program characteristics: 4](#_Toc19046796)

# Algorithm:

1. Greet user
2. Ask user to enter character name and select character class.
3. Repeat 3. for another user.
4. Ask if would like to add more players.
5. If yes, do 3.
6. If no, break;
7. Until one player is left:
8. Ask p1 what would like to do:
9. Choose tools unique to class, then attack
10. Health pack then attack?
11. Defence boost then attack?
12. Attack.
13. Execute what the player chooses.
14. Record order of deaths for leaderboard at end of game.
15. Display leaderboard.
16. Game over.

# Generalized-character-class characteristics:

1. Note that the base class will undergo polymorphism.
2. Health, attack and defence (private).
3. Has perceived health (public).
4. Percieved health = health + defense.
5. New health = perceived health – enemy attack.
6. Has pointers to addresses of its tools (private).
7. Has a pointer to itself (public to allow easy interaction).
8. Ability to create own tools and link itself to those tools upon creation.

Code:

aCharacterType(){//begin constructor

createTools(Character\*self){

ClassTools\*one;

ClassTools\*two;

this->pointerToTool1=one;

this->pointerToTool2=two;

}

}//end constructor

1. Ability to use tools on any character.

Code:

useTool(Character\*target){

if(this->selectedToolPointer!=NULL){

(this->selectedToolPointer)->toolUse(target);

}

}

1. Ability to take damage on behalf of any character except the one that it is attacking or its only enemy.
2. Ability to give its tools to any character.

Code:

giveTool(Character\*target){

target->extraToolPointer=this->selectedToolPointer;

this->selectedToolPointer=NULL;//ensure that donor cannot use this tool by //employing an if loop to check if selected tool to be used is NULL before use.

}

# Generalized-tool-class characteristics:

1. Note that the base class will undergo polymorphism.
2. Has a characteristic modifier (amount) and target value to be modified (ex: health [perceived], attack etc), that is unique to each morph of the base class.
3. Has a Boolean variable that is changed to false once used by owner object.
4. Code for when used:

useTool(Character\*target){

target->affectedTargetCharacteristic=(target->percievedTargetCharacteristic) – (amount deducted by that morph of tool);

//note that if there is no perceived characteristic, then the realAndCurrentCharacteristic is //used

}

# Program characteristics:

1. Saves all written output for display at end of program (with timestamps) and to be saved in a text file called “log.txt”.
2. Clearly separates each action visually (via ----------).
3. Clearly separates each round visually (via ===========).