Tatyana Vlaskin

Assignment 3 – Inheritance  and Files

**Design:**

1. First I need to create a parents class, lets call it Character:

class Character{

protected:

string name;

int attack,

int defence;

int armor;

int strength\_point;

public:

Character( ); // default constructor

Character(string &newname, int &armor1, int &strength\_point1): name(newname), armor(armor1), strength\_point(strength\_point1) {}

void setAttack(int attack\_new);

void setDefence(int defence\_new);

void setArmor(int armor\_new);

void setStrengthPoint(int strength\_new);

int getAttack();

int getDefence();

int getArmor();

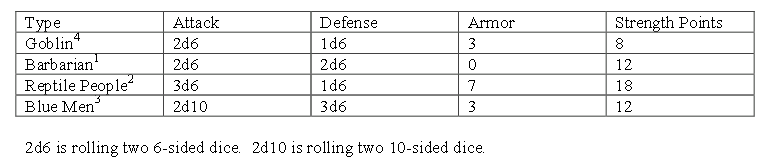
int getStrengthPoint();

void Wound(int damage, int defence);

1. void Wound(int damage, int defence, Character Character1,Character Character2 ); will look somewhat like this:{

this->Character1.strength\_points = this->Character1.strength\_points – Character1.getAttack()+Character2.getDefence();} // the set defense function will include the armor component- see below

1. Next step will be to create a subclass for each of the creatures. There are 4 creatures, so I need to create 4 subclasses



So subclasses will look somewhat like this:

**class Goblin : public Character {**

public:

Goblin( );

Goblin(string &newname, int &armor1, int &strength\_point1, int attack1, int defence1): Character (newname, armor1, strength\_point1), attack(SetAttack(attack1)), deffence(SetDeffence(defence1)) {}

GOBLIN: ATTACK 2D6 AND DEFFERNCE 1D6

voidSetAttack (int attack\_new){

attack1 = 1 + rand % 6;

attack2 = 1 + rand % 6;  
 attack = attack2+attack2;

attack\_new = attack  
}

//TOTAL DEFENSE WILL BE CALCULATED using the following formula:

// Defence = DefenceDiceRolls + Armor Vlue

void setDeffence(int defence\_new){

deference = 1 + rand % 6;

//also armor will be added to the defense as well

defence = deffence +armor;

defence = defence\_new;}

**class Barbarian: public Character {**

public:

Barbarian( );

Barbarian(string &newname, int &armor1, int &strength\_point1, int attack, int deffence): Character (newname, armor1, strength\_point1), attack(SetAttack(attack1)), deffence(SetDeffence(defence1)) {}

GOBLIN: ATTACK 2D6 AND DEFFERNCE 2D6

voidSetAttack (int attack\_new){

attack1 = 1 + rand % 6;

attack2 = 1 + rand % 6;  
 attack = attack2+attack2;}

//TOTAL DEFENCE WILL BE CALCULATED using the following formula:

// Defence = DefenceDiceRolls + Armor Vlue

void setDeffence(int defence\_new){

defence1= 1 + rand % 6;

defence2= 1 + rand % 6;

defence = defence2+defence1;

//also armor will be added to the defense as well

defence = defence +armor;}}

**class Reptile: public Character {**

public:

Reptile( );

Reptile(string &newname, int &armor1, int &strength\_point1, int attack, int deffence): Character (newname, armor1, strength\_point1, attack(SetAttack(attack1)), deffence(SetDeffence(defence1)) {}

REPTILE: ATTACK 3D6 AND DEFFERNCE 1D6

voidSetAttack (int attack\_new){

attack1 = 1 + rand % 6;

attack2 = 1 + rand % 6;

attchak3 = 1 + rand % 6;  
 attack = attack1+attack2+attack3;  
}

void setDeffence(int defence\_new){

defence= 1 + rand % 6;

//also armor will be added to the defense as well

defence = deffence +armor;

}

**class BlueMan: public Character {**

public:

BlueMan( );

BlueMan(string &newname, int &armor1, int &strength\_point1,int attack, int defence): Character (newname, armor1, strength\_point1, attack(SetAttack(attack1)), deffence(SetDeffence(defence1)) {}

BLUE MAN: ATTACK 3D6 AND DEFFERNCE 1D6

voidSetAttack (int attack\_new){

attack1 = 1 + rand % 10;

attack2 = 1 + rand % 10;

attack = attack1+attack2;  
}

void setDeffence(int defence\_new){

defence1= 1 + rand % 6;

defence2= 1 + rand % 6;

defence3= 1 + rand % 6;

defence = defence1+ defence2+ defence3 + armor;

1. When the game starts, the user will be asked what kind of character they want to be:

void characterSelection(){

cout<< endl << "Choose your figher class. " << endl;

cout<< "1. Goblin." << endl << "2. Barbarian "<<endl << "3. Reptile" << 4. Blue Man “ << endl;

int result;

cin >> result;

switch(result){

case 1:

return Goblin

case 2:

return Barbarian

case 3:

return Reptile

case 4:

return Blue Man

}

Depending on the selection a corresponding class will be instantiated.

1. Next step, the user will be asked which monster, they want to fight. In real fantasy, it would make sense to randomly chose a monster to fight; however, it will make the testing very difficult and I do not think that I have time for that.

So the user will be asked which monster they want to fight. There will be a switch statement similar to the one what I provideD above for character selection. Ill place this switch statement into a function and call it, lets say void monsterSelection().

This function will be called multiple times, inside the

void characterSelection() function{

cout<< endl << "Choose your figher class. " << endl;

cout<< "1. Goblin." << endl << "2. Barbarian "<<endl << "3. Reptile" << 4. Blue Man “ << endl;

int result;

cin >> result;

switch(result){

case 1:

return Goblin

monsterSelection()

case 2:

return Barbarian

monsterSelection()

case 3:

return Reptile

monsterSelection()

case 4:

return Blue Man

monsterSelection()

}

1. Once we have 2 characters, we start a combat.

Ill make a function like:

Void combat(){

while (Character1. strength\_point >0 && Character2. strength\_point >0){

Character1.SetAttack();

Character2.wound(); // function was described previously

Character2.SetAttack();

Character1.wound();

}

if (Character2. strength\_point<=0){

cout <<endl<< "Congratulations! You killed the monster!" << endl;}

if (Character1. strength\_point <=0){ // please note that character1 is the user

cout << "You are dead! You lost." << endl;

}

}

**Testing:**

For testing I will chose

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Whats are we testing | How we are testing | What is expected | What is the output | PASS/FAIL |
| Make sure that radom number of attack points is generated each time | cout << randomAttackPoints << " sum+=randomAttackPoints; | Each time there is a different attack points | Each time there is a different attack points | pass |
| Make sure Goblin rolls the dice x2 for attack | There is a for loop with the :  cout << randomAttackPoints << " sum+=randomAttackPoints; | 2 values for attack should be displayed | 2 values for attack are displayed | pass |
| Make sure Goblin gets values from 1-6 for each dice roll | There is a for loop with the :  cout << randomAttackPoints << " sum+=randomAttackPoints; | Values of attack1 and attack 2 between **1-6** | Values of attack1 and attack 2 between **1-6** | pass |
| Make sure Goblin attack1 and attach2 value add up and assigned to total attack points | There is a for loop with the :  cout << randomAttackPoints << " sum+=randomAttackPoints; | Attack will be equal to sum of 2 dice roles | Attack will be equal to sum of 2 dice rolls | pass |
| Do similar tests described for Goblin for Barbarian | See Goblin | See goblin | See goblin | pass |
| Do similar tests described for Goblin for Reptile | Similar to Goblin, the difference is that, there have to be 3 rolls of dice and dice values are from 1 to 3. There is a for loop with the :  for(int i=0; i<atackNumberOfDiceRoles; i++){  cout << randomAttackPoints << " sum+=randomAttackPoints | Make sure that are attack1, attack2 and attack3 have value and they are in the range **1-3** | There are 3 values for attack and values are between 1-3 | pass |
| Do similar tests described for Goblin for Blue Man | Similar to Goblin, the different is that, there have to be 2 rolls of dice and dice values are from 1-10. | Make sure that are value for attack 1 and attack2 and values are between **1-10** | There are 2 values for attack and values are between 1-10 | pass |
| Make sure Goblin rolls the dice x1 for defense | for(int i=0; i<defenceNumberOfDiceRoles;  randomDeffencePoints = rand()%defenceDiceSides+1; cout << randomDeffencePoints << " "; sum+=randomDeffencePoints;  } | Defense have value | There is a value | pass |
| Make sure Goblin gets values from 1-6 for each dice roll for defense | for(int i=0; i<defenceNumberOfDiceRoles;  randomDeffencePoints = rand()%defenceDiceSides+1; cout << randomDeffencePoints << " "; sum+=randomDeffencePoints;  } | Values of defense is between **1-6** | Value is between 1 and 6 | pass |
| Do similar tests described in Goblin for Barbarian. The difference is that barbarian needs to roll dice x2 for defense | for(int i=0; i<defenceNumberOfDiceRoles;  randomDeffencePoints = rand()%defenceDiceSides+1; cout << randomDeffencePoints << " "; sum+=randomDeffencePoints;  } | Values of defense is between **1-6** | Values of defense is between **1-6** | pass |
| Make sure barbarian defence1 and defense values add up and to total defense points | for(int i=0; i<defenceNumberOfDiceRoles;  randomDeffencePoints = rand()%defenceDiceSides+1; cout << randomDeffencePoints << " "; sum+=randomDeffencePoints;  } | defense will be equal to sum of defense1 and defense | defense is equal to sum of defense1 and defense | pass |
| Do similar tests described for Goblin for Reptile (defense) | See golbin | See golbin | See goblin | pass |
| Do similar tests described for Goblin for Blue Man | Similar to Goblin, the different is that, there have to be 3 rolls of dice | Make sure that are value for defense1, defense2 and defense3 and values are between **1-6** | that are value for defense1, defense2 and defense3 and values are between **1-6** | pass |
| Make sure that appropriate class is instantiated when you use switch statement | cout statements to display the name, strength\_ponts and armor for each character type. Strength will be displayed at the beginning of the battle. Armor will be displayed during the battle. Number of dice roles will be displayed during the battle | Compare values that were displayed to the value provided in the table- see above | Values compares | pass |
| Make sure that resulted attack points are calculated correctly. | int netAttackPoints=player.attack()-enemy.deffence();  cout << endl<< "Total damage - armor is " << netAttackPoints << " - " << enemy.getarmor();  netAttackPoints= netAttackPoints - enemy.getarmor();  cout << " = " << netAttackPoints <<end | Values that were denerated by rolling dice (see above mentioned tests) are used in the formula, appropriate armor value is subtracted | Values that were denerated by rolling dice (see above mentioned tests) are used in the formula, appropriate armor value is subtracted | pass |
| Make sure that strength points do not go up during the battle when armor exceeded netAttackPoints. | if(netAttackPoints<0){  netAttackPoints=0;  }  There is a cout statement for netstrength | Strength never go up | Strength points never go up | pass |
| Make sure that resulted strength points are calculated correctly | Cout statement with the net Strength points | At the end of the each battle – each time your press f, the resultant strength is calculated using strength- attack | At the end of the each battle – each time your press f, the resultant strength is calculated using strength- attack | Pass |
| Check that winner is announced when one of the characters runs out of strength\_point | cout statement that print sstrength\_points after each attack and defense round. | When health is 0 or less, winner/looser message is displayed | When health is 0 or less, winner/looser message is displayed | pass |
| Check that whoever runs out of strength point is labeled as looser | cout statement that print sstrength\_points after each attack and defense round. | Whoever runs out of strength\_point first lost of the game | Whoever runs out of strength\_point first lost of the game | Pass |
| Test Goblin/ Goblin combination that its possible to win and its possible to loose for a player | Choose goblin for a player and chose goblin for an enemy | Player can win and player can loose | Player can win and player can loose | Pass |
| Test Goblin/ barbarian combination that its possible to win and its possible to loose for each character | Choose goblin for a player and chose barbarian for an enemy or vise versa | Goblin can win or loose  Barbarin can lose or win | Goblin looses all the time, once golbin won – it kept loosing all the time and then all of a sudden it won | Pass  goblin will be able to win, but I do not have time for that. |
| Test Goblin/ reptile combination that its possible to win and its possible to loose for each character | Choose goblin for a player and chose reptile for an enemy or vise versa | Goblin can win or loose  reprile can lose or win | Goblin looses all the time | Pass?  Reptile has a lot of strength points, so I think its normal for goblin to loose. Maybe if I test it million times, the goblin will be able to win, but I do not have time for that. |
| Test Goblin/ blue men combination that its possible to win and its possible to loose for each character | Choose goblin for a player and chose blue men for an enemy or vise versa | Goblin can win or loose  Blue men can lose or win | Goblin looses all the time | Pass?  Blue man has a lot of strength points, and extermly strong attack function, I do not think that goblin will ever be able to win blue man, even if I test the program million times. |
| Test barbarian/reptile combination. Maybe sure that each character can win/loose | Choose barbarian for a player and chose replite for an enemy or vise versa | barbarian can win or loose  replite can lose or win | Barbarian always looses and reptile always win | Pass?  Per discussion board. This is done intentionaly to make TAs life easy. |
| Test barbarian/blue combination. Maybe sure that each character can win/loose | Choose barbarian for a player and chose blue man for an enemy or vise versa | barbarian can win or loose  blue can lose or win | BlueMan beats barbarian all the time | Pass?  Blue man has a lot of strength points, and extermly strong attack |
| Test barbarin/ barbarian combination that its possible to win and its possible to loose for a player | Choose barbarian for a player and chose barbarin for an enemy | Player can win and player can loose | Player can win and player can loose | Pass |
| Test reptile/blue combination. Maybe sure that each character can win/loose | Choose reptile for a player and chose blue man for an enemy or vise versa | reptile can win or loose  blue can lose or win | BlueMan beats ReptilePerson all the time | Pass?  Blue man has a lot of strength points, and extermly strong attack. This is ok per discussion board |
| Test reprile/ reprile combination that its possible to win and its possible to loose for a player | Choose reptile for a player and chose replite for an enemy | Player can win and player can loose | Player can win and player can loose | Pass – but this battle takes really long time |
| Test blue man/ blue man combination that its possible to win and its possible to loose for a player | Choose blue man for a player and chose blue man for an enemy | Player can win and player can loose | Player can win and player can loose | Pass – but this battle takes really long time |

REFLECTION:

There are few changes in my design. First of all I decided to make my variables more description- changes in name. And the main change is that I decided to add additional variable into the class character.

class CHARECTOR{

protected:

string name;

int atackNumberOfDiceRoles;

int atackDiceSides;

int armor;

int defenceNumberOfDiceRoles;

int defenceDiceSides;

int damage;

int strength;

int ststore;

public:

int getarmor(){return(armor);}

int getstrength(){return(strength);}

void setstrength(int s){

if(s>ststore){

strength=ststore;

}

else{

if(s<=0){

strength=0;

}

else{

strength=s;

}

}

}

int getstnstore(){return(ststore);}

int getdamage(){return(damage);}

string getname(){return(name);}

};

Specifically, int atackNumberOfDiceRoles, int atackDiceSides; int defenceNumberOfDiceRoles, int defenceDiceSides;

My initial plan was have attack and deffernse function in each of the 4 child classes that looked like that:

voidSetAttack (int attack\_new){

attack1 = 1 + rand % 6;

attack2 = 1 + rand % 6;  
 attack = attack2+attack2;

attack\_new = attack  
}

However, I decided that there will be a lot of redundancy in code. At the same time, during the design process I overlooked the sentence that: For pur poses right now each subclass will vary only in the values in the table. Which basically, is an indication that child classes can have only constructor and inherit all functions from the parent character class.

This in turn led to another change. I got rid of all, but one set functions. The only set function that is left is void setstrength(int strength) that will reset strength after each battle. As for the rest of the rest variables, I incorporated everything in the constructors in the children classes and changed defense and attack functions to make when reusable for each character type, which in turn eliminated redundancy in code ( now I do not need to have 4 attack functions and 4 defense functions). Specifically, int atackNumberOfDiceRoles, int atackDiceSides; int defenceNumberOfDiceRoles, int defenceDiceSides variable let me do that:

int attack(){

int sum=0;

int randomAttackPoints

randomAttackPoints = rand()%atackDiceSides + 1;

sum+=randomAttackPoints;

}

return(sum);

}

int deffence(){

int sum=0;

int randomDeffencePoints

for(int i=0; i<defenceNumberOfDiceRoles;

randomDeffencePoints = rand()%defenceDiceSides+1

sum+=randomDeffencePoints;

}

return(sum);

This change in turn, allowed me to change the subclasses for each character, into classes that contain only constructors and inherit all other functions form the parents class Character.

///CLASS GOBLIN

class Goblin:public CHARECTOR{

public:

Goblin(){name="Goblin";armor=3;strength=ststore=8;atackNumberOfDiceRoles=2;atackDiceSides=6;defenceNumberOfDiceRoles=1;defenceDiceSides=6;}

};

///CLASS BARBARIAN

class Barbarian:public CHARECTOR{

public:

Barbarian(){name="Barbarian";armor=0;strength=ststore=12;atackNumberOfDiceRoles=2;atackDiceSides=6;defenceNumberOfDiceRoles=2;defenceDiceSides=6;}

};

///CLASS REPTILEPEOPLE

class ReptilePeople:public CHARECTOR{

public:

ReptilePeople(){name="ReptilePeople";armor=7;strength=ststore=18;atackNumberOfDiceRoles=3;atackDiceSides=6;defenceNumberOfDiceRoles=1;defenceDiceSides=6;}

};

///CLASS BLUEMEN

class BlueMen:public CHARECTOR{

public:

BlueMen(){name="BlueMen";armor=3;strength=ststore=12;atackNumberOfDiceRoles=2;atackDiceSides=10;defenceNumberOfDiceRoles=3;defenceDiceSides=6;}

};

I’ve also got rid of the void Wound(int damage, int defence, Character Character1,Character Character2 ) function. I had no idea how to create 2 instances of the child class in the parent class. I wasted a lot of time trying to find some information online, but has now luck. As a result of this, I moved the code in the main function:

PLAYER IS ATTACKING

int netAttackPoints=player.attack()-enemy.deffence();

netAttackPoints= netAttackPoints - enemy.getarmor();

enemyNewStrength=enemy.getstrength()-netAttackPoints;

ENEMY IS ATTACKING:

netAttackPoints=enemy.attack()-player.deffence();

netAttackPoints= netAttackPoints - player.getarmor();

playerNewStrength=player.getstrength()-netAttackPoints;

During the testing stage, I’ve noticed that strength points were going up, when armor exceeded netAttackPoints. This does not make any sense logically, so I’ve added this code:

if(netAttackPoints<0){

netAttackPoints=0;

}

This solved the problem.

My next challenge was to get different numbers during the random dice rolls. First I tried to pause my system, but it was anointing to wait for a long time. After doing some additional reading I came across the srand(time(0)), which we covered last quarter, but I forgot about it.

That did not fix the problem. So I came up with the pause function:

void pause(int dur)

{

int temp = time(NULL) + dur;

while(temp > time(NULL));

}

Its annoying, but after each dice roll, the system will pause for a sec, before it moves on to the next dice role. It was taking so long to test this program. I found, a post on the discussion board:

[Eric Stevens](https://piazza.com/class/hux8wmp35ta6d9?cid=89) [1 day ago](https://piazza.com/class/hux8wmp35ta6d9?cid=89)

Great Mr Rooker, thank you for the confirmation.

This was my suspicion but it is a relief to here that from you. I actually was able to one instance of the Goblin beating the Barbarian. After long hours of battling i realized that if I made rounds go too fast that every number that came form the same dice roll(ie if they were both 2d6) the numbers would be the same. I post this here to maybe give someone else insight on this problem. DONT PUT SRAND(TIME) SEEDS THAT OCCUR EVERY TIME A DICE FUNCTION IS CALLED>>>

Put one strand in main and be done with it.

I tried that and it works, it is such a relief.