CSCI3180 Assignment 3 Report

Name : Mao Yuxuan SID : 1155062018

In task2, I use the dynamic scoping characteristic of perl language to minimize the use of extra parameters. In my implementation, after an attack is performed, the program will call the "**be_attacked**" subprogram to modify the hp. **Since different attack method have different harm values, we need to modify the harm value before each "be_attacked" is called**.

```
our $harm = 1000;
```

(indicating the harm value of different attack method)

```
sub attackWithSword{
    my $self = shift;
    my $enermy = shift;
    local $harm = 800;
    $enermy->be_attacked();
```

(in attack with sword, it change the harm value and call the "be_attacked" method)

```
local $harm = 600;
$enermy->be_attacked();
```

```
$harm = 1000;
$enermy->be_attacked();
```

(similarly, in attack with magic and black magic sword, change the value and call method)

```
sub be_attacked{
    my $self = shift;
    my $actual_harm = 0;
    $actual_harm = $harm - $self->{'dp'};
    $self->{'hp'} -= $actual_harm;
    print "the remain hp of the enermy is : ",$self->{'hp'},"\n";
}
```

(in "be_attacked" method, use the harm to calculate what value to be deducted)

If one is implemented this method with static scoping, he/she may need to pass a parameter "harm" to the "be_attacked" method, which makes more work to do.

To me, there are 2 advantages of dynamic scoping. On the one hand, it **can reduce the use of parameters, and makes the code cleaner** (when the parameters became 4 or 5 or more, the effect is significant), On the other hand, **in some cases, it makes the code more reasonable.** For instance, for an attack, the person to attack should have the information of the attack (like harm of the attack), which is shown in "attackWithSword". For the person being attacked, it only takes the harm, he doesn't need to be told (be passed with a parameter) about how much harm in that attack.