

[Dragon Battle]

Authored by Kisora Thomas and Zachary Collins

The Algorithm **Dragon Battle** will do exactly what it sounds like. You, and your computer opponent will control a dragon which will duke it out until one of them ‘faints’ from exhaustion, AKA when the HP hits zero. You will use global variables to declare ‘User’ Dragon, and ‘Computer’s’ Dragon, and use the dragon’s **skillset** to do the battle against the foe. That said, you’ll have to implement a Computer to control the behaviour of their dragon, randomly selecting moves each **Turn**.

Implement a Dragon within the **class** Dragon, whose **Object** holds the following values:

* **Health Points** - (HP) Randomly generated upon the calling of the **new** constructor function. This random integer value is between 70 -100 points. Create the total health points as a reference. Create a current HP variable for the battle.
* **Attack Factor** - (AF) The base value of the attack is multiplied by the Attack Factor, a value randomly generated between 0-1.
* **Defense Factor** - (DF) The calculated amount of damage done by an attack after the Attack Factor has been calculated is shielded by the Dragon’s Defense Factor. The Defense factor can be between 0-1, randomly generated upon construction.
* **Mana Points** - (MP) Mana is used as the price for Magical Skills. It is a random amount generated from 5-15 points. Save the total MP in a variable for reference, and the current amount for the Battle.
* **Gem** - Dragons are given a color to refer to them. Consider the color the name.
* **Skillset** - The four moves that A dragon is able to use. Can be implemented as a Vector<String>. When a String is selected, it calls the Behaviour of that skill, passing the name as the value.

**Skillsets** are the moves or **abilities** a Dragon can select one of on any given turn. When a Dragon Object is created, it is populated with **FOUR** Skills, each randomly selected from the global Skillset Vector<String>, containing all the moves available for the game. Create a **class Skills** that handles the passing of a string of the skill, and calls the **Behaviour** of the skill to affect the enemy. The Skills class is static, and you can add as many skills as you’d like. **Skill Creation Theory** is provided below, when coming up with Skills:

* **Name** - When the dragon uses the move, it should refer to this skill’s name.
* **Attack or Defense**- Attack moves lower the HP or MP of the foe, and Defend Skills lower Attack Factors or Defense Factors of the opponent.
* **Physical** or **Magical** - Skills that are physical consume no Mana points, but are generally weaker than Magical Skills. Magical Skills consume a random amount of Mana Points on each use, but are generally stronger than physical moves.
* **Behaviour** - The actual effect of the skill, passing an enemy Dragon into its method. It modifies the Dragon of target based on the previously described abstract identity of the Skill.

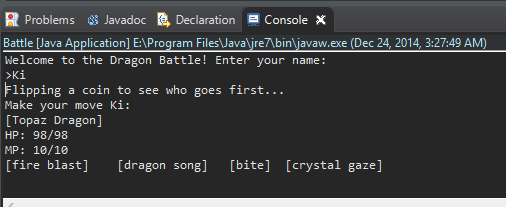
A minimum of four skills must be designed for the Battle to commence. Examples include :

* **Roar** - Lower attack Factor of foe by 10-30%
* **Flirt**  - Lower defense factor of foe by 10-30%
* **Dragon Dance** - Restore 15-35% of Mana Points
* **Bite** - Reduce enemy HP by 10 - 15 Points
* **Fire Blast** - Consume 50-100% of total Mana Points, to reduce enemy HP by 15-25 Points. If the number of mana points required exceeds the number remaining, then the attack fails, and the turn is wasted.
* **Heat Wave** - Increase attack factor of self by 5-30% Consumes 25-50% total Mana and causes 5-10% total health damage;
* **Crystal Gaze** - Reduce foe’s Mana Points by 5-30% of total MP, and add half of Mana stolen to your own Mana pool.
* **Dragon Song** - Regain Health Points equal to 5-15% of total HP
* **Shred** - Reduce enemy HP by 5-10 points, and lower attack factor of enemy 5-10%
* **Blood Sacrifice** - Consume 10-20% of total Mana Points, and sacrifice half of current health to reduce enemy health by half of their current health
* **Berserk** - Lower self Defense factor by 50%, reduce all stats of foe by 5%.

Revisions:

* percentages are annoying. Just use natural numbers

A generic battle should implement a command line tool and look something like this:

  
 .contains() //leave this if you would

//

changed crystal gaze up a bit and added one new skill<--Great improvement!

finished the skills and debugged mine a bit more

//ideas;; instead of having mana be percentages im going to set mana to 100 upon creation, and then depending on color of the dragon make either specific attacks or attributes higher and lower. Like brown dragon has more strength so its shred and bitem moves are slightly stronger. or blue is a caster dragon, so it has another 20 mana

switch (move) {

case "roar": roar(self,enemy);break;

case "bite": bite(self,enemy);break;

case "dragon dance": Dragon dance(self);break;

case "shred": shred(self,enemy);break;

what does that do <--- Just like if, else if, else if. The ‘else’ statement in a switch-case version is called ‘default:’

Oh, and the break statements are imperative.

i was thinking of assigning each skill to a number and then rand.nextint()<-- Consider using strings.

Enter your name:

> Kisora

Player Kisora summons a new Emerald Dragon.  
 Player Computer summons a new Topaz Dragon.

Flipping coin to see who goes first.   
 It’s heads-- you go first.

Emerald Dragon’s Moves:   
 [Fire Blast] [Dragon Dance]

[Shred] [Flirt]

> flirt

Emerald Dragon used Flirt. It reduced Topaz Dragon’s defense by

14%!

Computer’s turn.

Topaz Dragon uses [Crystal Gaze]!  
 It reduced Emerald Dragon’s Mana Points by 20%!

Your turn.

....Continue until someone’s HP is at zero.

Final Note: I’ve got more than one solutions in mind for this algorithm. For guidance, come to me with your questions. Kisora Thomas - **419-450-6085**. Cheers! Let’s battle!

**import** java.util.\*

**public** class Dragon{

public int totalHP;

public int currentHP;

public double attackFactor;

public double defenseFactor;

public int manaPoints;

public Vector<String> skillSet = **new** Vector<String>();

public Dragon(){

Random rand = new Random();

this.totalHP = rand.nextInt(30)+70;

this.currentHP = totalHP;

this.attackFactor = rand.nextDouble();

this.defenseFactor = rand.nextDouble();

Returns the next pseudorandom, uniformly distributed double value between 0.0 and 1.0 from this random number generator's sequence. // just delete whenever you see it

this.skillSet = Skills.createNewSkillset();

}

public void listSkills(){

for(int i =0;i<skillSet.size();i++){

print(“[“+skillSet.get(i)+”]\t”+ //...yada yada...);

}

}

public void act( String move,Dragon self, Dragon enemy){

for(int i=0;i<skillSet.size();i++){

if(skillSet.get(i).contains(move)//move is valid

Skills.execute(move,self,enemy);

}

else print(“Try another move.”)

}

}

public class Skills{

**public void** execute(String move,Dragon self, Dragon enemy){

switch(move){

case “roar”: roar(self,enemy);**break**;

case “bite”: bite(self,enemy);**break**;

case “dragon dance”: dragonDance(self);**break**;

case “shred”: shred(self,enemy);**break**;

}

}

**public void** bite(Dragon self, Dragon enemy){

Random rand = new Random();

int bitePower = rand.nextInt(15)+5;

int finalBitePower=(self.attackFactor \* bitePower)+bitePower;

finalBitePower = finalBitepower-(finalBitePower

\*enemy.defenseFactor);

enemy.currentHP = enemy.currentHP - finalBitePower;

}

}

//Code to select random string

**public static** String Goodbye() {

String Goodbye[] = { "Goodbye now!", "Talk to you later!", "See

ya!",

"A tout a l'heure!", "Have a good one!", "Later!",

"Nice talking with you!", "Until we meet again!",

"Until another day!", "Bye-bye!", "Ciao!",

"Farewell.",

"Later!", "Peace out.", "Take care!", "Bye! Stay

inspired!",

"Bye! Keep your head up!", };

String word = (Goodbye[new Random().nextInt(Goodbye.length)]);

**return** word;

}

public class Battle{

Dragon userDragon, computerDragon;

public int main(String[] args){

print(“Welcome to the dragon battle! Enter your name:”);

String name = input.nextLine();

userDragon = new Dragon(); //Calls the constructor

computerDragon = new Dragon();

while(!(userDragon.currentHP ==0)|| !(computerDragon.currentHP ==0)){

print(“Make your move:)

print(userDragon.printSkills());

String move = input.nextLine().toLowerCase().trim();

userDragon.do(move,computerDragon);

}

}

for(i=0;i<dragonArmySize;i++)

Vector dragonArmy.add( new Dragon());

print(dragonArmy.get(i).color or .HP or .AttackFactor or this or that...

EX:  
  
public class Dragon{

string name = “Argor”;

public Dragon(){

**this**.name = selectRandomName(“Argor”, “Syvern”, “Draxxe”);

}

public String printName(){

print(name);

}

}

main(){

Dragon d1 = new Dragon();

Dragon d2 = new Dragon();

Dragon d3 = new Dragon();

print(d1.printName() + d2.printName() + d3.printName());

}

Output: “Argor Argor Draxxe ”

Vector<String> names = new Vector<String>();

names.add(“Zac”);

names.add(“Kisora”);

String firstName = names.get(0);

print(firstName);

Output: Zac

names.remove(0); //Shifts elements left one space

firstName = names.get(0);

print(firstName);

Output: Kisora

names.add(“Draxxe”,3); //Inserts at specified index point, creates it if it doesn’t exist yet.

print(name.get(0) + names.get(1) + names.get(2)+ names.get(3));

Output: Kisora null null Draxxe //Dynamically updates size