DavidValsan2404871932024Mini8

March 16, 2025

1 Mini-project Assessed Exercise

2 Level 8

2.1 David Valsan

2.2 240487193

2.3 24 October 2024

2.4 Version 3

- Added file input & output
- Record turned into ADT
  - Accessor methods added for record
- feed(), play() and train() methods removed in favor of accessor methods with the same effect
- currentEnemyID now stored in the player creature record instead of main() method so it
- Added separate method for generating random numbers
- Added message argument to input method, to be displayed beforehand
- Split main into smaller methods
- Fixed bug where invalid input to the question of fighting the next enemy immediately, was treated as valid input "y"

2.5 Summary of the Question

Write a procedural program that simulates bio-engineered mythical creatures that must be cared for.

2.6 Justification that the program passes this level

This is based on tick boxes at start of the question description for this exercise. - My program is literate - Program clearly decomposed into multiple smaller methods, doing distinct jobs - Methods take arguments and return results - Excellent style over comments, indentation, variable usage, final variables - Excellent use of methods - Includes - Accessor methods - File input and output - Records - Loops within loops - Array accessed by a loop - Loops - Decision statements - Variables, assignments and expressions - Screen output, keyboard input - Variable names give clear indication of their use - Uses array arguments - Indentation makes structure clear - Methods have comments indicating their use - Final variables used for literal constant - All variables defined within methods and have minimum scope

2.7 The literate program development

2.7.1 PlayerCreature and EnemyCreature

Implementation (how it works) Player creature contains data about the player's creature, while enemy creature, the enemy creatures are the ones the player has to fight to win, thus they do not need other attributes like hunger, fun and a name. Magic will be used to decide a victor of a battle. The species is needed as to create an image in the mind of the player, and make each battle feel more unique and so the player feels they are making progress. currentEnemyID was moved to allow for easier file input and output. A function can only return one variable so instead of making separate save and load functions for the player creature record and for the ID, it was easier to make it an attribute in the record and load it with the rest of the variables.

- Added 2 new boolean fields, technically these were moved from main() to the PlayerCreature record. This was done for the purpose of splitting up main(), and since these variables were directly changed alongside the player creature record, and in Java 2 variables cannot be returned at once, they now exist here, to be returned within the record.

[52]:
```
class PlayerCreature
    {
        String species;
        String name;
        int hunger;
        int fun;
        int magic;
        int currentEnemyID;
        boolean gameOver;
        boolean gameWon;
    }

class EnemyCreature
    {
        String species;
        int magic;
    }
```

2.7.2 inputString

What it does Allows player to input a string.

Implementation (how it works) Declares a scanner, and returns whatever the player inputs. Created to make input easier during coding, and to avoid declaring scanner every function it is needed. Argument passed in to be displayed as message before input is taken.

[12]:
```
//takes string from user and returns it
//makes program easier to read
//scanner does not have to to be declared every function that takes player input
//in this program the player only needs to input a string
public static String inputString(String message)
```

```
{
    Scanner scanner= new Scanner(System.in);
    System.out.println(message);
    return scanner.nextLine();
}
```

Testing

[31]:
```
inputString("Message?");
```

```
Message?

 Feed
```

[31]:
```
Feed
```

2.7.3 genRandomInt

What it does Generates random integer, with possible value from 0 to limit-1.

Implementation (how it works) Declares a "random" object, and uses its nextInt() method, alongside the limit passed in as an argument to generate a random number whose value ranges from 0 to value of limit -1. e.g. limit=4 ; range= 0-3

[39]:
```
//general method
//uses Random() to generate a random integer
//limit is passed in as argument- generate number from 0 to (limit-1)
public static int genRandomInt(int limit)
{
    Random random= new Random();
    return random.nextInt(limit);
}
```

Testing

[40]:
```
genRandomInt(10);
```

[40]:
```
1
```

[41]:
```
genRandomInt(10);
```

[41]:
```
4
```

[42]:
```
genRandomInt(10);
```

[42]:
```
8
```

2.7.4 displayActions

What it does Displays player's available actions each turn, and allows them to input any of the ones mentioned. #### Implementation (how it works) Each action is neatly printed line by line. By calling inputString(), it allows player to input any of the above actions, but check occurs in main(), where the input is returned to. Each of the printed actions can be done by the player and has their own method. Also leaves a line blank at the beginning making it easier to tell each different round apart from each other.

Since inputString() takes a message now as argument, a blank string is passes in leaving a row between printed action and player's input. All the action could be passed in as the message using backslash-"n", however displaying them on different lines is more legible for programmers.

[49]:
```
//displays every action the player can take (input)
//normal round
public static String displayActions()
{
    System.out.println("\n ");
    System.out.println(">Feed");
    System.out.println(">Play");
    System.out.println(">Train");
    System.out.println(">Check stats");
    System.out.println(">Change name");
    System.out.println(">New creature");
    System.out.println(">Battle");
    System.out.println(">Save");
    System.out.println(">Load");
    System.out.println(">Quit");
    return inputString("");
}
```

Testing

[50]:
```
displayActions();
```

```
>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit


 Feed
```

[50]: Feed

2.7.5 inputSpecies

What it does Asks user to input the species of their creature and double checks their decision with them, as it cannot be changed.

Implementation (how it works) Double checks with the player and only allows "y" or "n" to be entered. Uses 2 local string variables to ensure this(one holds the species, while the other the player's decision). Uses the inputString() function to ask for the species of the creature and get input from the player. Logic expressions were changed to ONLY allow "y" or "n" as a response, before it used to loop as long as "y" was not entered.

Name changed from "select" to "input" as it is a more accurate descriptor.

[13]:
```
//allows player to input their creature's species
//double checks with them as the species cannot be later changed
public static String inputSpecies()
{
    String temp="", speciesAnswer="";
    while(temp.equals("n") | !temp.equals("y"))
        {
            temp="";
            speciesAnswer=inputString("What species is your creature?");
            while(!temp.equals("y") & !temp.equals("n"))
                {
                    temp=inputString("Are you sure? You cannot change this␣
 ↪later.(y/n)");
                }

        }
    return speciesAnswer;
}
```

Testing

[34]:
```
inputSpecies();
```

```
What species is your creature?
 Goblin
Are you sure? You cannot change this later.(y/n)
 fnsonf
Are you sure? You cannot change this later.(y/n)
 slnl
Are you sure? You cannot change this later.(y/n)
```

```
 n
What species is your creature?
 Goblin
Are you sure? You cannot change this later.(y/n)
 y
```

[34]:
```
Goblin
```

2.7.6 inputName

What it does Similar to inputSpecies(), asks user for the name of their creature. There is no double-check as logically there should not be a problem with changing their creature's name later.

Implementation (how it works) Preserved from older version. Returns the return of inputString() in which it passes a message asking for the name of the creature to be input. Name changed to inputName(), to match inputSpecies, similarly "input" is a more accurate descriptor.

[14]:
```
//calls method to take input from the user
//asks for name
//no double-check as this can be changed
public static String inputName()
{
    return inputString("Name you creature.(This can be changed later)");
}
```

Testing

[36]:
```
inputName();
```

```
Name you creature.(This can be changed later)
 Bob
```

[36]:
```
Bob
```

2.7.7 createPlayerCreature

What it does Creates new record for a player creature.

Implementation (how it works) Species and name are passed in as strings from their respective functions(player input), every other attribute is by default set to 0 or false.

[6]:
```
//creates record for a new player creature
public static PlayerCreature createPlayerCreature(String nSpecies, String nName)
{
    PlayerCreature nCreature= new PlayerCreature();
    nCreature.species=nSpecies;
```

```
    nCreature.name=nName;
    nCreature.hunger=0;
    nCreature.fun=0;
    nCreature.magic=0;
    nCreature.currentEnemyID=0;
    nCreature.gameOver=false;
    nCreature.gameWon=false;

    System.out.println(nCreature.species+"\n"+nCreature.name+"\n"+nCreature.
 ↪hunger+"\n"+nCreature.fun+"\n"+nCreature.magic+"\n"+nCreature.
 ↪currentEnemyID+"\n"+nCreature.gameWon+"\n"+nCreature.gameOver);
    //line used for testing, deleted in actual code
    return nCreature;
}
```

Testing

[12]:
```
createPlayerCreature("Goblin", "Bob");
```

```
Goblin
Bob
0
0
0
0
false
false
```

[12]:
```
REPL.$JShell$19C$PlayerCreature@5a534634
```

2.7.8 createEnemyCreature

What it does Creates new record for an enemy creature.

Implementation (how it works) Species and magic values are passed in from the initiateEnemies() method, from the final arrays. Used to create a new enemy creature record. Separate from player creature as PlayerCreature holds more data, which an enemy creature would not need hence being a waste.

[47]:
```
//creates record for a new enemy creature
public static EnemyCreature createEnemyCreature(String nSpecies, int nMagic)
{
    EnemyCreature nCreature= new EnemyCreature();
    nCreature.species=nSpecies;
    nCreature.magic=nMagic;

    System.out.println(nCreature.species+"\n"+nCreature.magic);
    //used for testing, removed in actual program
```

```
    return nCreature;
}
```

Testing

[14]:
```
createEnemyCreature("Goblin", 127);
```

```
Goblin
127
```

[14]:
```
REPL.$JShell$20$EnemyCreature@27be2561
```

2.7.9 displayStatus

What it does Upon request returns displays the player creature's species, name, hunger, fun and magic stats. This allows the user to monitor their creature's status to ensure it does not turn on them. Similarly allows them to see their magic progress.

Implementation (how it works) Hunger,fun, magic and name variables are all passed in, to be displayed in the statements as seen below. Also leaves a blank line at the top so the user can more easily read the console output and tell apart this section from the possible actions.

[53]:
```
//displays current stats and their values
public static void displayStatus(PlayerCreature creature)
{
    System.out.println("\n ");
    System.out.println("Name: "+getName(creature));
    System.out.println("Hunger: "+getHunger(creature));
    System.out.println("Fun: "+getFun(creature));
    System.out.println("Magic: "+getPlayerMagic(creature));
    return;
}
```

Testing

[54]:
```
displayStatus(createPlayerCreature("Goblin","Bob"));
```

```
Goblin
Bob
0
0
0
0
false
false
```

```
Species: Goblin
Name: Bob
Hunger: 0
Fun: 0
Magic: 0
```

2.7.10 Get Accessor Methods

What it does Used to make records into ADTs. Returns value of specified field of the passed in record.

Implementation (how it works) All fields need to be read at one point or another hence there is one method for each field. Player and Enemy creatures have 2 fields named the same, in which case the name of the method adds "player" or "enemy" in order to clearly keep them separated.

[11]:
```
//getter accessor methods -player
public static String getPlayerSpecies(PlayerCreature pc)
{
    return pc.species;
}
public static String getName(PlayerCreature pc)
{
    return pc.name;
}
public static int getHunger(PlayerCreature pc)
{
    return pc.hunger;
}
public static int getFun(PlayerCreature pc)
{
    return pc.fun;
}
public static int getPlayerMagic(PlayerCreature pc)
{
    return pc.magic;
}
public static int getCurrentEnemy(PlayerCreature pc)
{
    return pc.currentEnemyID;
}
public static boolean getGameOver(PlayerCreature pc)
{
    return pc.gameOver;
}
public static boolean getGameWon(PlayerCreature pc)
{
    return pc.gameWon;
}
```

```
//getter accessor methods -enemy
public static String getEnemySpecies(EnemyCreature ec)
{
    return ec.species;
}
public static int getEnemyMagic(EnemyCreature ec)
{
    return ec.magic;
}
```

Testing

[37]:
```
PlayerCreature testCreature=createPlayerCreature("Goblin", "Bob");
```

```
Goblin
Bob
0
0
0
0
false
false
```

[38]:
```
getPlayerSpecies(testCreature);
```

[38]: Goblin

[39]:
```
getName(testCreature);
```

[39]: Bob

[40]:
```
getHunger(testCreature);
```

[40]: 0

[41]:
```
getFun(testCreature);
```

[41]: 0

[42]:
```
getPlayerMagic(testCreature)
```

[42]: 0

[43]:
```
getCurrentEnemy(testCreature);
```

[43]: 0

[44]:
```
getGameOver(testCreature);
```

[44]: false

[45]:
```
getGameWon(testCreature);
```

[45]: false

[46]:
```
EnemyCreature testCreature=createEnemyCreature("Goblin", 100);
```

```
Goblin
100
```

[47]:
```
getEnemySpecies(testCreature);
```

[47]: Goblin

[48]:
```
getEnemyMagic(testCreature);
```

[48]: 100

2.7.11 Set Accessor Methods

What it does Changes the value of a record field.

Implementation (how it works) This is done 1 of 2 ways, through either addition or assignment. In case of assignment, it simply assigns the field to the value passed in as an argument. In the case of addition, this is done to more easily allow the hunger,fun and magic stats and enemy ID to be altered. The main worry is when loading values from an external file, however this is prevented by creating a new record(values at 0), setting the values from the file(0+value=value). This allows me to decrease stats by passing in negative values.

For hunger and fun as discussed before I limited these 2 at 10 so the player has to play the game as intended (cannot get high stats early on then stop worrying about them); however due to the way the program changed to allow for ADTs, the checks limiting the values to 10 have been moved to the setter methods.

[17]:
```
//setter accessor methods (player only)
public static PlayerCreature setName(PlayerCreature pc, String nName)
{
    pc.name=nName;
    System.out.println(pc.name);//removed in full program, used for testing
    return pc;
}
public static PlayerCreature setHunger(PlayerCreature pc, int hungerModifier)
{
    pc.hunger+=hungerModifier;
    if(pc.hunger>10)
```

```
    {
        pc.hunger=10;
    }
    System.out.println(pc.hunger);//removed in full program, used for testing
    return pc;
}
public static PlayerCreature setFun(PlayerCreature pc, int funModifier)
{
    pc.fun+=funModifier;
    if(pc.fun>10)
    {
        pc.fun=10;
    }
    System.out.println(pc.fun);//removed in full program, used for testing
    return pc;
}
public static PlayerCreature setMagic(PlayerCreature pc, int magicModifier)
{
    pc.magic+=magicModifier;
    System.out.println(pc.magic);//removed in full program, used for testing
    return pc;
}
public static PlayerCreature setCurrentEnemy(PlayerCreature pc, int nID)
{
    pc.currentEnemyID+=nID;
    System.out.println(pc.currentEnemyID);//removed in full program, used for␣
 ↪testing
    return pc;
}

public static PlayerCreature setGameOver(PlayerCreature pc, boolean nGO)
{
    pc.gameOver=nGO;
    System.out.println(pc.gameOver);//removed in full program, used for testing
    return pc;
}
public static PlayerCreature setGameWon(PlayerCreature pc, boolean nGW)
{
    pc.gameWon=nGW;
    System.out.println(pc.gameWon);//removed in full program, used for testing
    return pc;
}
```

Testing

[52]:
```
PlayerCreature testCreature = createPlayerCreature("Goblin","Bob");
```

```
Goblin
```

```
Bob
0
0
0
0
false
false
```

[56]:
```
setName(testCreature, "Daniel");
```

```
Daniel
```

[56]: REPL.$JShell$19C$PlayerCreature@76fde201

[58]:
```
setHunger(testCreature,7);
```

```
7
```

[58]: REPL.$JShell$19C$PlayerCreature@76fde201

[59]:
```
setHunger(testCreature,12);
```

```
10
```

[59]: REPL.$JShell$19C$PlayerCreature@76fde201

[60]:
```
setFun(testCreature,3);
```

```
3
```

[60]: REPL.$JShell$19C$PlayerCreature@76fde201

[61]:
```
setFun(testCreature,13);
```

```
10
```

[61]: REPL.$JShell$19C$PlayerCreature@76fde201

[62]:
```
setMagic(testCreature,5);
```

```
5
```

[62]: REPL.$JShell$19C$PlayerCreature@76fde201

[63]:
```
setCurrentEnemy(testCreature,1);
```

```
1
```

[63]: REPL.$JShell$19C$PlayerCreature@76fde201

[64]:
```
setGameOver(testCreature,true);
```

```
true
```

[64]: REPL.$JShell$19C$PlayerCreature@76fde201

[65]:
```
setGameWon(testCreature,true);
```

```
true
```

[65]: REPL.$JShell$19C$PlayerCreature@76fde201

2.7.12 feed

What it does Increase the player creature's hunger stat by 3. Prints an appropriate message describing this action.

Implementation (how it works) Most of the method's functionality has been moved to the setter as part of ADT design, however in order to keep the same neat code design, and to still print out a unique message when this is performed(setHunger() is called in other scenarios), the method was preserved and simplified, calling the setter to increment the hunger stat by 3.

[57]:
```
//increase hunger(feed/become less hungry)
public static PlayerCreature feed(PlayerCreature creature)
{
    System.out.println("You fed "+getName(creature) +".");
    return setHunger(creature, 3);
}
```

Testing

[67]:
```
feed(createPlayerCreature("Goblin","Bob"));
```

```
Goblin
Bob
0
0
0
0
false
false
You fed Bob.
3
```

[67]: REPL.$JShell$19C$PlayerCreature@6abb8e09

2.7.13 play

What it does Increase the player creature's fun stat by 3. Prints an appropriate message describing this action.

Implementation (how it works) Most of the method's functionality has been moved to the setter as part of ADT design, however in order to keep the same neat code design, and to still print out a unique message when this is performed(setFun() is called in other scenarios), the method was preserved and simplified, calling the setter to increment the fun stat by 3.

[58]:
```
//increase fun
public static PlayerCreature play(PlayerCreature creature)
{
    System.out.println("You played fetch with "+getName(creature)+".");
    return setFun(creature, 3);
}
```

Testing

[69]:
```
play(createPlayerCreature("Goblin","Bob"));
```

```
Goblin
Bob
0
0
0
0
false
false
You played fetch with Bob.
3
```

[69]:
```
REPL.$JShell$19C$PlayerCreature@6efbcac0
```

2.7.14 train

What it does Increase the player creature's magic stat by 5. Prints an appropriate message describing this action.

Implementation (how it works) As with the 2 preceding methods, most of train() has been moved to the setter, however similarly to those in explicitly increments the passed in player creature's magic by 5.

[59]:
```
//increase magic
public static PlayerCreature train(PlayerCreature creature)
{
    System.out.println("You trained "+getName(creature)+" in the arcane.");
    return setMagic(creature, 5);
}
```

Testing

[71]:
```
train(createPlayerCreature("Goblin","Bob"));
```

```
Goblin
Bob
0
0
0
0
false
false
You trained Bob in the arcane.
5
```

[71]:
```
REPL.$JShell$19C$PlayerCreature@6bc865b2
```

2.7.15 nextRound

What it does After every player action, that makes progress(feed,play,train),this function randomly decreases a stat from 0 to 3. Creates difficulty to the game as player balances fun and hunger stats with their main goal of training magic to its max. Also adds more uncertainty, difficulty and replayability by doing it randomly instead of by a fixed amount, it can also give the player feeling of stress and relief depending on their luck.

Implementation (how it works) Altered to work with ADTs and the generic random method. As before it decreases both or just 1 of hunger and fun depending on the boolean values passed in. It now instead calls the generic genRandomInt() method, passing in the limit of 4 (0-3). A "-" sign is also added to the value in the setter method to indicate it will be decreased.

[60]:
```
//passively decreases the creature's stats every round
public static PlayerCreature nextRound(PlayerCreature creature, boolean␣
 ↪isHunger, boolean isFun)
{
    if(isHunger)
    {
        creature=setHunger(creature, -(genRandomInt(4)));
    }
    if(isFun)
    {
        creature=setFun(creature, -(genRandomInt(4)));
    }


    return creature;
}
```

Testing

[82]:
```
nextRound(createPlayerCreature("Goblin","Bob"), true, true);
```

```
Goblin
Bob
0
0
0
0
false
false
-1
-3
```

[82]:
```
REPL.$JShell$19C$PlayerCreature@70c40569
```

2.7.16 battleEnemy

What it does Lets the player creature fight the next enemy.

Implementation (how it works) The p. creature record, e. creature array and currentEnemyID(index) are passed in and are used to compare the magic of the 2 the one with the higher magic wins and the result is returned. I had it return an integer instead of a boolean as to allow a 3rd possibility of a draw, so instead 1,-1 and 0 are returned representing the result.

[61]:
```
//compares stats of the passed in player and enemy creature records
// return 1,0 or -1 representing the result of the battle
//1= win; 0=draw; -1=loss
//outcome decided by their magic attributes
public static int battleEnemy(PlayerCreature playerCreature, EnemyCreature[]␣
 ↪enemyCreatures, int currentEnemyID)
{
    if(getPlayerMagic(playerCreature)>␣
 ↪getEnemyMagic(enemyCreatures[currentEnemyID]))
    {
        System.out.println(getName(playerCreature)+" defeated the "+␣
 ↪getEnemySpecies(enemyCreatures[currentEnemyID]));
        return 1;
    }
    else if (getPlayerMagic(playerCreature)<␣
 ↪getEnemyMagic(enemyCreatures[currentEnemyID]))
    {
        System.out.println(getName(playerCreature)+" was defeated in combat␣
 ↪with the "+ getEnemySpecies(enemyCreatures[currentEnemyID]));
        System.out.println("You managed to escape while␣
 ↪"+getName(playerCreature)+" became a meal.");
        return -1;
    }
```

```
    else
    {
        System.out.println(getName(playerCreature)+ " and the " +␣
 ↪getEnemySpecies(enemyCreatures[currentEnemyID]) +" were evenly matched.");
        System.out.println("The two of you managed to escape the encounter.");
        return 0;
    }
}
```

2.7.17 displayCreatureAttackCommands

What it does Similar to displayActions(), displays all the player's possible commands, however this displayed instead when the player does not take care proper care of their creature and it turns on them and attacks them. These 2 will similarly have their own methods. By giving the player 2 choices I can add a sense if agency for the player, as even though they are limited by the program they still have a choice to make and strategise accordingly.

Implementation (how it works) Passes in species string to display as part of the message and allows user to input an action, which is returned to main, where the input will be checked. Changed to use the inputString() function, instead of declaring its scanner.

[22]:
```
//display available commands
public static String displayCreatureAttackCommands(PlayerCreature creature)
{
    System.out.println("\n ");
    System.out.println("Your "+getPlayerSpecies(creature)+" is unhappy with␣
 ↪your care.");
    System.out.println(">Run");
    System.out.println(">Fight back");
    return inputString("");
}
```

Testing

[86]:
```
displayCreatureAttackCommands(createPlayerCreature("Goblin","Bob"));
```

```
Goblin
Bob
0
0
0
0
false
false


Your Goblin is unhappy with your care.
```

```
>Run
>Fight back


 Run
```

[86]:
```
Run
```

2.7.18 fightBack

What it does Allows the user to fight back against their creature, the stronger the creature the least likely the user is to survive.

Implementation (how it works) Uses a randomly generated number from 0 to the magic target(max magic) , and compares it to the creature's magic, the stronger the creature(more magic) the less likely will the user be to win.

Has been changed to use the magicMax as the max. value, and the generic genRandomInt() method to get the value that will be compared to the creature's magic.

[24]:
```
//allows user to fight back if attacked by their creature
//but the stronger the creature raised the higher the chance of escaping
public static boolean fightBack(PlayerCreature creature, int magicMax)
{
    if(genRandomInt(magicMax+1)>getPlayerMagic(creature))
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

Testing

[88]:
```
fightBack(createPlayerCreature("Goblin","Bob"), 50);
```

```
Goblin
Bob
0
0
0
0
false
false
```

[88]:
```
true
```

2.7.19 runAway

What it does Alternative to fighting back, instead a coin is flipped to decide whether the player survives or not.

Implementation (how it works) Generates random number from 0 to 1(0 or 1), and if the number is 1, the player wins, if it is 0 they lose.

Chnaged to use the genRandomInt() method instead of its own Random object.

[23]:
```
//alternate strategy
//50% chance of escape
public static boolean runAway()
{
    Random random = new Random();
    if(random.nextInt(2)==1)
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

Testing

[93]:
```
runAway();
```

[93]:
```
false
```

[97]:
```
runAway();
```

[97]:
```
true
```

2.7.20 saveGame

What it does Saves/writes the attributes of the player creature's record fields in an external file, so these values can be retrieved later, so even when data is lost by the program ending or player losing, they can just load this in to resume progress.

Implementation (how it works) Get accessor methods are used to get the fields' values and outputStream is used to store each on a separate line. The text file "MythicCreatureCareSave.txt" will store these, and it should be created in the case it does not exist by default within the same folder.

[17]:
```
//write the values of the player creature record to an external file
//each value is placed on a different row
```

```
//gameOver and gameWon are not saved, as if either is set to true this always␣
 ↪means the game has ended
public static void saveGame(PlayerCreature pc) throws IOException
{
    PrintWriter outputStream = new PrintWriter (new␣
 ↪FileWriter("MythicCreatureCareSave.txt"));
    outputStream.println(getPlayerSpecies(pc));
    outputStream.println(getName(pc));
    outputStream.println(getHunger(pc));
    outputStream.println(getFun(pc));
    outputStream.println(getPlayerMagic(pc));
    outputStream.println(getCurrentEnemy(pc));
    outputStream.close();
}
```

Testing

[18]:
```
saveGame(createPlayerCreature("Goblin", "Bob"));
```

```
Goblin
Bob
0
0
0
0
false
false
```

2.7.21 loadGame

What it does The method that actually takes the data from the external file and assigns it to the corresponding attributes, essentialy "continuing" from the point saved.

Implementation (how it works) The local string variable "temp" is used to store the contents of the first line, this is so if the file is empty(first line is blank), null can be returned instead of an object, which the method calling loadGame() will deal with. Otherwise since the only method writing to the file is saveGame() it means if the first line has data, it will always be the species.

"temp" is used here so the same line is not read twice, and since inputStream.readLine() will move on to the next line, so the data on the first line is saved somewhere. This is also why setCurrentEnemy() operates with addition, rather than a set increment of 1, or assignment; so it can both be increased by 1 when winning a battle, as well as be increased from 0 to the value in the file when loading.

[15]:
```
//if data had been written to the save file, it takes each value from each row␣
 ↪and assigns it to the correct record field
// these will always be in the same order
public static PlayerCreature loadGame() throws IOException
```

```
{
    BufferedReader inputStream = new BufferedReader (new␣
 ↪FileReader("MythicCreatureCareSave.txt"));
    String temp =inputStream.readLine();
    if(temp!=null)
    {
        PlayerCreature playerCreature= new PlayerCreature();
        playerCreature=createPlayerCreature(temp,inputStream.readLine());
        playerCreature=setHunger(playerCreature,Integer.parseInt(inputStream.
 ↪readLine()));
        playerCreature=setFun(playerCreature,Integer.parseInt(inputStream.
 ↪readLine()));
        playerCreature=setMagic(playerCreature,Integer.parseInt(inputStream.
 ↪readLine()));
        playerCreature= setCurrentEnemy(playerCreature,Integer.
 ↪parseInt(inputStream.readLine()));
        inputStream.close();
        return playerCreature;
    }
    inputStream.close();
    return null;
}
```

Testing

[19]:
```
loadGame();
```

```
Goblin
Bob
0
0
0
0
false
false
0
0
0
0
```

[19]:
```
REPL.$JShell$13$PlayerCreature@192a3b1
```

2.7.22 initiateEnemies

What it does Creates the array of enemies, for the player's creature to fight in order to win the game. Uses a final array for the enemy creatures' names, and use the magicMax variable from the previous version to calculate their magic.

Implementation (how it works) magicMax is passed in from main and used to get values for the enemies' magic, getting progressively stronger as they go on. By using the magicMax value I can more easily test the program by changing these values automatically. Each record in the array gets the respective species name and magic value from the respective array(same index in their arrays). Occurs once at the beginning of the program.

[48]:
```
//run once, at the beginning of the program
//initialises an array of pre-determined enemies for the player's creature to␣
 ↪fight
public static EnemyCreature[] initiateEnemies(int magicMax)
{
    final String[] enemySpecies={"Goblin", "Siren", "Minotaur", "Giant",␣
 ↪"Dragon"};
    final int[] enemyMagic={(int)(Math.ceil(0.1*magicMax)), (int)(Math.ceil(0.
 ↪2*magicMax)), (int)(Math.ceil(0.5*magicMax)), (int)(Math.ceil(0.
 ↪7*magicMax)), magicMax};
    EnemyCreature[] enemyCreatures=new EnemyCreature[5];

    for (int i=0; i<=enemySpecies.length-1; i++)
        {
            enemyCreatures[i]= new EnemyCreature();
            enemyCreatures[i]= createEnemyCreature( enemySpecies[i],␣
 ↪enemyMagic[i]);
        }
    return enemyCreatures;
}
```

2.7.23 creatureRabid

What it does Runs when the player has neglected their creature, offers them a choice of 2 actions.

Implementation (how it works) Separated from main(), called when either the hunger or fun stats decrease to -10. Displays available actions(input), and calls appropriate method. The method always returns a boolean which is assigned to the local "survived", which is later used to run appropriate code either resetting the player's creature, or ending the game.

In the case of invalid input the method should repeat HOWEVER this should be done by the while loop in main rather than this method.

[26]:
```
//separated from main, runs when the creature goes rabid(fun or hunger stat␣
 ↪<=-10)
public static PlayerCreature creatureRabid(PlayerCreature playerCreature, int␣
 ↪magicMax)
{
    String answer=displayCreatureAttackCommands(playerCreature).toLowerCase();
    boolean survived=false;
```

```
   if (answer.equals("fight back"))
    {
        playerCreature=checkForSurvival(fightBack(playerCreature, magicMax),␣
 ↪playerCreature);
    }
    else if (answer.equals("run"))
    {
        playerCreature=checkForSurvival(runAway() , playerCreature);
    }
    else
    {
        System.out.println("Could not recognise command.");
    }

    return playerCreature;
}
```

Testing

[37]:
```
creatureRabid(createPlayerCreature("Goblin","Bob"), 50)
```

```
Goblin
Bob
0
0
0
0
false
false


Your Goblin is unhappy with your care.
>Run
>Fight back


 Run

true
```

[37]:
```
REPL.$JShell$13$PlayerCreature@bd27c16
```

[43]:
```
creatureRabid(createPlayerCreature("Goblin","Bob"), 50)
```

```
Goblin
Bob
0
0
0
```

```
0
false
false


Your Goblin is unhappy with your care.
>Run
>Fight back


 fight back

You survived the encounter but your Goblin ran away.
 Time to start again.
What species is your creature?

 Unicorn

Are you sure? You cannot change this later.(y/n)

 y

Name you creature.(This can be changed later)

 Jim

Unicorn
Jim
0
0
0
0
false
false
```

[43]:
```
REPL.$JShell$13$PlayerCreature@52a8ee74
```

[44]:
```
creatureRabid(createPlayerCreature("Goblin","Bob"), 50)
```

```
Goblin
Bob
0
0
0
0
false
false


Your Goblin is unhappy with your care.
>Run
```

```
>Fight back


 psgspgjs
Could not recognise command.
```

[44]:
```
REPL.$JShell$13$PlayerCreature@495da9f3
```

2.7.24 checkForSurvival

What it does Either ends the game or tells the player to create a new creature, depending on whether they survived or not.

Implementation (how it works) This used to be part of creatureRabid(), however was separated to prevent a bug. It would occur when entering invalid input, since survived was by default set to false, it would always result in the game ending. Now the code checking for survival only occurs when valid input is entered, and the while loop in the outside method should keep asking the player for valid input. However since there are 2 scenarios where the survival check code is run, copying it is inefficient so instead both scenarios call upon this method now.

Method checks for the value of the passed in survive variable and either ends the game by assigning gameOver, or resets the player creature by creating a new one.

[18]:
```
//separated from rest of the method so it should only run in the case of valid input
//checks value of the "sruvived" boolean passsed
//player's progress is either reset or the game is lost
public static PlayerCreature checkForSurvival(boolean survived, PlayerCreature pc)
{
    if (survived)
    {
        System.out.println("You survived the encounter but your "+getPlayerSpecies(pc)+ " ran away. \n Time to start again.");
        pc=createPlayerCreature(inputSpecies(), inputName());
    }
    else
    {
        pc=setGameOver(pc, true);
    }
    return pc;
}
```

Testing

[19]:
```
checkForSurvival(true, createPlayerCreature("Goblin","Bob"))
```

```
Goblin
Bob
0
0
0
0
false
false
You survived the encounter but your Goblin ran away.
 Time to start again.
What species is your creature?

 Goblin

Are you sure? You cannot change this later.(y/n)

 y

Name you creature.(This can be changed later)

 Bob

Goblin
Bob
0
0
0
0
false
false
```

[19]: REPL.$JShell$21$PlayerCreature@2512c35f

[20]:
```
checkForSurvival(false, createPlayerCreature("Goblin","Bob"))
```

```
Goblin
Bob
0
0
0
0
false
false
true
```

[20]: REPL.$JShell$21$PlayerCreature@63269dcf

2.7.25 regularRound

What it does Displays available actions for the player in the case of a regular round(most of the time). Each action has an associated method called.

Implementation (how it works) Separated from main(). As described it simply calls the appropriate method with the appropriate arguments. The unique event is in case of battle where different methods may be called depending on the result(represented by an integer). In the case of a win, the enemy ID is updated and the game checks if the player has beat all the enemies, in which case the loop in main() will be broken out of by updating the booleans in player record. In the case of a loss the player's creature is reset. And in the case of a draw no changes are made. Something to note is that the question asking the player to keep going now only appears in the case of a win or draw, as it did not make sense for it to appear when a loss occurs. It would also appear when all 5 enemies were defeated so now a check occurs that prevents it from being asked if the enemyID has reached its last value(5) in which case the player wins the game so it simply ends there.

[79]:
```
//regular round, separated from main()
//runs appropriate methods based on the action the player input
public static PlayerCreature regularRound(PlayerCreature playerCreature,␣
 ↪EnemyCreature[] enemies) throws IOException
{
    String answer=displayActions().toLowerCase();
    int battleResult;
    if (answer.equals("feed"))
    {
        playerCreature=feed(playerCreature);
        playerCreature=nextRound(playerCreature, false, true);
    }
    else if (answer.equals("play"))
    {
        playerCreature=play(playerCreature);
        playerCreature=nextRound(playerCreature, true, false);
    }
    else if (answer.equals("train"))
    {
        playerCreature=train(playerCreature);
        playerCreature=nextRound(playerCreature, true, true);
    }
    else if (answer.equals("check stats"))
    {
       displayStatus(playerCreature);
    }
    else if (answer.equals("change name"))
    {
        playerCreature=setName(playerCreature, inputName());
    }
    else if (answer.equals("new creature"))
    {
        System.out.println("You abandoned"+getName(playerCreature)+". You␣
 ↪monster.");
        playerCreature=createPlayerCreature(inputSpecies(), inputName());
```

```
   }
   else if (answer.equals("battle"))
   {
       answer="y";
       while (!answer.equals("n") && getCurrentEnemy(playerCreature)<5 &&␣
↪answer.equals("y"))
       {
           battleResult= battleEnemy(playerCreature, enemies,␣
↪getCurrentEnemy(playerCreature));
           if(battleResult==1)
           {
               playerCreature=setCurrentEnemy(playerCreature,1);
               if (getCurrentEnemy(playerCreature)>=enemies.length)//game is␣
↪won, all enemies defeated, breaks out the loop
               {
                   playerCreature=setGameOver(playerCreature, true);
                   playerCreature=setGameWon(playerCreature, true);
               }
               else
               {
                   answer=inputString("Would you like to battle the next enemy?
↪(y/n)");
                   playerCreature=nextRound(playerCreature, true, true);
               }


           }
           else if (battleResult==-1)//loss = reset
           {
               playerCreature=createPlayerCreature(inputSpecies(),␣
↪inputName());
               answer="n";
           }
           else//draw, can only be 0(no point in extra check)
           {
               playerCreature=nextRound(playerCreature, true, true);
               answer=inputString("Would you like to battle the next enemy?(y/
↪n)");
           }

       }
   }

   else if (answer.equals("save"))
   {
       saveGame(playerCreature);
```

```
    }
    else if (answer.equals("load"))
    {
        if(loadGame()==null)
        {
            System.out.println("Save file was empty, no creature can be␣
↪retrieved.");
        }
        else
        {
            playerCreature=loadGame();
        }
    }

    else if (answer.equals("quit"))
    {
        playerCreature=setGameOver(playerCreature, true);
    }

    else
    {
        System.out.println("Could not recognise command.");
    }
    return playerCreature;

}
```

Testing

[63]:
```
regularRound(createPlayerCreature("Goblin", "Bob"), initiateEnemies(10));
```

```
Goblin
Bob
0
0
0
0
false
false
Goblin
1
Siren
2
Minotaur
5
Giant
7
Dragon
```

```
10


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit


 feed
You fed Bob.
3
0
```

[63]:
```
REPL.$JShell$13$PlayerCreature@5ce13a9
```

[65]:
```
regularRound(createPlayerCreature("Goblin", "Bob"), initiateEnemies(10));
```

```
Goblin
Bob
0
0
0
0
false
false
Goblin
1
Siren
2
Minotaur
5
Giant
7
Dragon
10


>Feed
>Play
>Train
>Check stats
```

```
>Change name
>New creature
>Battle
>Save
>Load
>Quit


 Play

You played fetch with Bob.
3
-3
```

[65]:
```
REPL.$JShell$13$PlayerCreature@4da7b239
```

[66]:
```
regularRound(createPlayerCreature("Goblin", "Bob"), initiateEnemies(10));
```

```
Goblin
Bob
0
0
0
0
false
false
Goblin
1
Siren
2
Minotaur
5
Giant
7
Dragon
10


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit
```

```
 train

You trained Bob in the arcane.
5
-1
-2
```

[66]:
```
REPL.$JShell$13$PlayerCreature@426efcf6
```

[68]:
```
regularRound(createPlayerCreature("Goblin", "Bob"), initiateEnemies(10));
```

```
Goblin
Bob
0
0
0
0
false
false
Goblin
1
Siren
2
Minotaur
5
Giant
7
Dragon
10


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit


 check stats


Species: Goblin
Name: Bob
```

```
Hunger: 0
Fun: 0
Magic: 0
```

```
[68]: REPL.$JShell$13$PlayerCreature@7d2bea93
```

```
[69]: regularRound(createPlayerCreature("Goblin", "Bob"), initiateEnemies(10));
```

```
Goblin
Bob
0
0
0
0
false
false
Goblin
1
Siren
2
Minotaur
5
Giant
7
Dragon
10


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit


 change name

Name you creature.(This can be changed later)

 Jim

Jim
```

```
[69]: REPL.$JShell$13$PlayerCreature@6cfdeb4c
```

[70]:
```
regularRound(createPlayerCreature("Goblin", "Bob"), initiateEnemies(10));
```

```
Goblin
Bob
0
0
0
0
false
false
Goblin
1
Siren
2
Minotaur
5
Giant
7
Dragon
10


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit


 New creature

You abandonedBob. You monster.
What species is your creature?

 Fairy

Are you sure? You cannot change this later.(y/n)

 y

Name you creature.(This can be changed later)

 E

Fairy
E
```

```
0
0
0
0
false
false
```

[70]:
```
REPL.$JShell$13$PlayerCreature@3b290464
```

[80]:
```
regularRound(createPlayerCreature("Goblin", "Bob"), initiateEnemies(10));
```

```
Goblin
Bob
0
0
0
0
false
false
Goblin
1
Siren
2
Minotaur
5
Giant
7
Dragon
10


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit


 battle

Bob was defeated in combat with the Goblin
You managed to escape while Bob became a meal.
What species is your creature?
```

```
 fairy
Are you sure? You cannot change this later.(y/n)
 y
Name you creature.(This can be changed later)
 t
fairy
t
0
0
0
0
false
false
```

[80]:
```
REPL.$JShell$13$PlayerCreature@52e61eb0
```

[81]:
```
regularRound(createPlayerCreature("Goblin", "Bob"), initiateEnemies(10));
```

```
Goblin
Bob
0
0
0
0
false
false
Goblin
1
Siren
2
Minotaur
5
Giant
7
Dragon
10


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
```

```
>Save
>Load
>Quit
```

```
 save
```

[81]:
```
REPL.$JShell$13$PlayerCreature@4f099b43
```

[82]:
```
regularRound(createPlayerCreature("Goblin", "Bob"), initiateEnemies(10));
```

```
Goblin
Bob
0
0
0
0
false
false
Goblin
1
Siren
2
Minotaur
5
Giant
7
Dragon
10
```

```
>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit
```

```
 load
```

```
Goblin
Bob
0
0
```

```
0
0
false
false
0
0
0
0
Goblin
Bob
0
0
0
0
false
false
0
0
0
0
```

[82]:
```
REPL.$JShell$13$PlayerCreature@7b3122a1
```

[83]:
```
regularRound(createPlayerCreature("Goblin", "Bob"), initiateEnemies(10));
```

```
Goblin
Bob
0
0
0
0
false
false
Goblin
1
Siren
2
Minotaur
5
Giant
7
Dragon
10


>Feed
>Play
>Train
```

```
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit


 quit
```

```
true
```

[83]:
```
REPL.$JShell$13$PlayerCreature@17c565cf
```

2.7.26 printFinalMessage

What it does Prints one of 2 final messages depending on if the player has won or lost the game.

Implementation (how it works) Decides which statement to prove based on the value of the passed in gameWon argument.

[99]:
```
//runs when game has ended
//print appropriate message, based on the passed in gameWon boolean
//either a win or a loss
public static void printFinalMessage(boolean gameWon, PlayerCreature pc)
{
    //runs when game has ended
    if(getGameOver(pc))
    {
        System.out.println("YOU WON! Congratulations "+getName(pc)+" is the␣
 ↪strongest magical beast.");
    }
    else
    {
        System.out.println("You died! \n GAME OVER");
    }
    return;
}
```

Testing

[61]:
```
printFinalMessage(true,createPlayerCreature("Goblin","Bob"));
```

```
YOU WON! Congratulations Bob is the strongest magical beast.
```

[62]:
```
printFinalMessage(false,createPlayerCreature("Goblin","Bob"));
```

```
You lost!
 GAME OVER
```

2.7.27 main

What it does Calls the other methods which in turn call their own needed methods, uses while loops to repeatedly call methods, either from round to round, or simply repeats due to invalid input.

Implementation (how it works) Vastly cut down due to being separated into multiple methods. Declares a new player creature record which will be passed through the methods. It is also where the final variable for max. magic is declared and assigned. Enemies are initiated and the player creature is created initially. All these variables will be passed on to and returned from other methods.

The while loop should ensure the regular round run normally most of the time. In the case the creature goes rabid the while should ensure,even with invalid input the appropriate method is called- as no input should be able to change hunger, fun or gameOver in the player creature ADT.

[69]:
```
//calls other methods
//maintains while loops
public static void main(String[] a) throws IOException
{
    PlayerCreature playerCreature=new PlayerCreature();
    final int magicMax=10;

   EnemyCreature[] enemyCreatures= initiateEnemies(magicMax);
   playerCreature=createPlayerCreature(inputSpecies(), inputName());

    while(!getGameOver(playerCreature))
        {
            if (getHunger(playerCreature)<-10 | getFun(playerCreature) <-10)//␣
 ↪creature goes rabid
            {
                playerCreature=creatureRabid(playerCreature, magicMax);
            }

            else//regular turn
            {
                regularRound(playerCreature, enemyCreatures);
            }
        }

    printFinalMessage(getGameWon(playerCreature), playerCreature);

    return;

}
```

2.8 The complete program

This version will only compile here. To run it copy it into a file called initials.java on your local computer and compile and run it there.

[31]:
```
/* ************************************
  @author    David Valsan
  @SID       240487193
  @date      24 October 2024
  @version   3
  @level     8

    A program that alllows the user to choose and
    care for a mythical creature.
   ************************************/

import java.util.Scanner;
import java.util.Random;
import java.io.IOException;
import java.io.FileWriter;
import java.io.PrintWriter;
import java.io.BufferedReader;
import java.io.FileReader;

class PlayerCreature
    {
        String species;
        String name;
        int hunger;
        int fun;
        int magic;
        int currentEnemyID;
        boolean gameOver;
        boolean gameWon;
    }

class EnemyCreature
    {
        String species;
        int magic;
    }

public class creatureCare
    {
        //takes string from user and returns it
        //makes program easier to read
        //scanner does not have to to be declared every function that takes␣
 ↪player input
```

```
        //in this program the player only needs to input a string
        public static String inputString(String message)
        {
            Scanner scanner= new Scanner(System.in);
            System.out.println(message);
            return scanner.nextLine();
        }

        //general method
        //uses Random() to generate a random integer
        //limit is passed in as argument- generate number from 0 to (limit-1)
        public static int genRandomInt(int limit)
        {
            Random random= new Random();
            return random.nextInt(limit);
        }

        ////////////////////////////////////////////////////////

        //creates record for a new player creature
        public static PlayerCreature createPlayerCreature(String nSpecies,␣
↪String nName)
        {
            PlayerCreature nCreature= new PlayerCreature();
            nCreature.species=nSpecies;
            nCreature.name=nName;
            nCreature.hunger=0;
            nCreature.fun=0;
            nCreature.magic=0;
            nCreature.currentEnemyID=0;
            nCreature.gameOver=false;
            nCreature.gameWon=false;

            return nCreature;
        }

        //creates record for enemy creature
        public static EnemyCreature createEnemyCreature(String nSpecies, int␣
↪nMagic)
        {
            EnemyCreature nCreature= new EnemyCreature();
            nCreature.species=nSpecies;
            nCreature.magic=nMagic;

            return nCreature;
        }
```

```
        //run once, at the beginning of the program
        //initialises an array of pre-determined enemies for the player's␣
↪creature to fight
        public static EnemyCreature[] initiateEnemies(int magicMax)
        {
            final String[] enemySpecies={"Goblin", "Siren", "Minotaur",␣
↪"Giant", "Dragon"};
            final int[] enemyMagic={(int)(Math.ceil(0.1*magicMax)), (int)(Math.
↪ceil(0.2*magicMax)), (int)(Math.ceil(0.5*magicMax)), (int)(Math.ceil(0.
↪7*magicMax)), magicMax};
            EnemyCreature[] enemyCreatures=new EnemyCreature[5];

            for (int i=0; i<=enemySpecies.length-1; i++)
                {
                    enemyCreatures[i]= new EnemyCreature();
                    enemyCreatures[i]= createEnemyCreature( enemySpecies[i],␣
↪enemyMagic[i]);
                }
            return enemyCreatures;
        }

        ////////////////////////////////////////////////////////

        //getter accessor methods -player
        public static String getPlayerSpecies(PlayerCreature pc)
        {
            return pc.species;
        }
        public static String getName(PlayerCreature pc)
        {
            return pc.name;
        }
        public static int getHunger(PlayerCreature pc)
        {
            return pc.hunger;
        }
        public static int getFun(PlayerCreature pc)
        {
            return pc.fun;
        }
        public static int getPlayerMagic(PlayerCreature pc)
        {
            return pc.magic;
        }
        public static int getCurrentEnemy(PlayerCreature pc)
        {
            return pc.currentEnemyID;
```

```
        }
        public static boolean getGameOver(PlayerCreature pc)
        {
            return pc.gameOver;
        }
        public static boolean getGameWon(PlayerCreature pc)
        {
            return pc.gameWon;
        }

        //getter accessor methods -enemy
        public static String getEnemySpecies(EnemyCreature ec)
        {
            return ec.species;
        }
        public static int getEnemyMagic(EnemyCreature ec)
        {
            return ec.magic;
        }

        ////////////////////////////////////////////////////////

        //setter accessor methods (player only)
        public static PlayerCreature setName(PlayerCreature pc, String nName)
        {
            pc.name=nName;
            return pc;
        }
        public static PlayerCreature setHunger(PlayerCreature pc, int␣
↪hungerModifier)
        {
            pc.hunger+=hungerModifier;
            if(pc.hunger>10)
            {
                pc.hunger=10;
            }
            return pc;
        }
        public static PlayerCreature setFun(PlayerCreature pc, int funModifier)
        {
            pc.fun+=funModifier;
            if(pc.fun>10)
            {
                pc.fun=10;
            }
            return pc;
        }
```

```
        public static PlayerCreature setMagic(PlayerCreature pc, int␣
↪magicModifier)
        {
            pc.magic+=magicModifier;
            return pc;
        }
        public static PlayerCreature setCurrentEnemy(PlayerCreature pc, int nID)
        {
            pc.currentEnemyID+=nID;
            return pc;
        }

        public static PlayerCreature setGameOver(PlayerCreature pc, boolean nGO)
        {
            pc.gameOver=nGO;
            return pc;
        }
        public static PlayerCreature setGameWon(PlayerCreature pc, boolean nGW)
        {
            pc.gameWon=nGW;
            return pc;
        }

        ////////////////////////////////////////////////////////

        //increase hunger(feed/become less hungry)
        public static PlayerCreature feed(PlayerCreature creature)
        {
            System.out.println("You fed "+getName(creature) +".");
            return setHunger(creature, 3);
        }

         //increase fun
        public static PlayerCreature play(PlayerCreature creature)
        {
            System.out.println("You played fetch with "+getName(creature)+".");
            return setFun(creature, 3);
        }

        //increase magic
        public static PlayerCreature train(PlayerCreature creature)
        {
            System.out.println("You trained "+getName(creature)+" in the arcane.
↪");
            return setMagic(creature, 5);
        }
```

```
        //passively decreases the creature's stats every round
        public static PlayerCreature nextRound(PlayerCreature creature, boolean␣
↪isHunger, boolean isFun)
        {
            if(isHunger)
            {
                creature=setHunger(creature, -(genRandomInt(4)));
            }
            if(isFun)
            {
                creature=setFun(creature, -(genRandomInt(4)));
            }


            return creature;
        }

        ////////////////////////////////////////////////////////////////////////
↪////////////////////////////

        //allows player to input their creature's species
        //double checks with them as the species cannot be later changed
        public static String inputSpecies()
        {
            String temp="", speciesAnswer="";
            while(temp.equals("n") | !temp.equals("y"))
                {
                    temp="";
                    speciesAnswer=inputString("What species is your creature?");
                    while(!temp.equals("y") & !temp.equals("n"))
                        {
                            temp=inputString("Are you sure? You cannot change␣
↪this later.(y/n)");
                        }

                }
            return speciesAnswer;
        }

        //calls method to take input from the user
        //asks for name
        //no double-check as this can be changed
        public static String inputName()
        {
            return inputString("Name you creature.(This can be changed later)");
        }
```

```
        //displays current stats and their values
        public static void displayStatus(PlayerCreature creature)
        {
            System.out.println("\n ");
            System.out.println("Name: "+getName(creature));
            System.out.println("Hunger: "+getHunger(creature));
            System.out.println("Fun: "+getFun(creature));
            System.out.println("Magic: "+getPlayerMagic(creature));
            return;
        }

        //displays every action the player can take (input)
        //normal round
        public static String displayActions()
        {
            System.out.println("\n ");
            System.out.println(">Feed");
            System.out.println(">Play");
            System.out.println(">Train");
            System.out.println(">Check stats");
            System.out.println(">Change name");
            System.out.println(">New creature");
            System.out.println(">Battle");
            System.out.println(">Save");
            System.out.println(">Load");
            System.out.println(">Quit");
            return inputString("");
        }


        //display available commands
        public static String displayCreatureAttackCommands(PlayerCreature␣
↪creature)
        {
            System.out.println("\n ");
            System.out.println("Your "+getPlayerSpecies(creature)+" is unhappy␣
↪with your care.");
            System.out.println(">Run");
            System.out.println(">Fight back");
            return inputString("");
        }




        //allows user to fight back if attacked by their creature
        //but the stronger the creature raised the higher the chance of escaping
```

```
        public static boolean fightBack(PlayerCreature creature, int magicMax)
        {
            if(genRandomInt(magicMax+1)>getPlayerMagic(creature))
            {
                return true;
            }
            else
            {
                return false;
            }
        }

        //alternate strategy
        //50% chance of escape
        public static boolean runAway()
        {
            if(genRandomInt(2)==1)
            {
                return true;
            }
            else
            {
                return false;
            }
        }

        //write the values of the player creature record to an external file
        //each value is placed on a different row
        //gameOver and gameWon are not saved, as if either is set to true this␣
↪always means the game has ended
        public static void saveGame(PlayerCreature pc) throws IOException
        {
            PrintWriter outputStream = new PrintWriter (new␣
↪FileWriter("MythicCreatureCareSave.txt"));
            outputStream.println(getPlayerSpecies(pc));
            outputStream.println(getName(pc));
            outputStream.println(getHunger(pc));
            outputStream.println(getFun(pc));
            outputStream.println(getPlayerMagic(pc));
            outputStream.println(getCurrentEnemy(pc));
            outputStream.close();
        }

        //if data had been written to the save file, it takes each value from␣
↪each row and assigns it to the correct record field
        // these will always be in the same order
        public static PlayerCreature loadGame() throws IOException
```

```
        {
            BufferedReader inputStream = new BufferedReader (new␣
↪FileReader("MythicCreatureCareSave.txt"));
            String temp =inputStream.readLine();
            if(temp!=null)
            {
                PlayerCreature playerCreature= new PlayerCreature();
                playerCreature=createPlayerCreature(temp,inputStream.
↪readLine());
                playerCreature=setHunger(playerCreature,Integer.
↪parseInt(inputStream.readLine()));
                playerCreature=setFun(playerCreature,Integer.
↪parseInt(inputStream.readLine()));
                playerCreature=setMagic(playerCreature,Integer.
↪parseInt(inputStream.readLine()));
                playerCreature= setCurrentEnemy(playerCreature,Integer.
↪parseInt(inputStream.readLine()));
                inputStream.close();
                return playerCreature;
            }
            inputStream.close();
            return null;
        }

        ////////////////////////////////////////////////////////////////

        //runs when game has ended
        //print appropriate message, based on the passed in gameWon boolean
        //either a win or a loss
        public static void printFinalMessage(boolean gameWon, PlayerCreature pc)
        {
            //runs when game has ended
            if(gameWon)
            {
                System.out.println("YOU WON! Congratulations "+getName(pc)+" is␣
↪the strongest magical beast.");
            }
            else
            {
                System.out.println("You died! \n GAME OVER");
            }
            return;
        }

        //compares stats of the passed in player and enemy creature records
        // return 1,0 or -1 representing the result of the battle
```

```
        //1= win; 0=draw; -1=loss
        //outcome decided by their magic attributes
        public static int battleEnemy(PlayerCreature playerCreature,␣
↪EnemyCreature[] enemyCreatures, int currentEnemyID)
        {
            if(getPlayerMagic(playerCreature)>␣
↪getEnemyMagic(enemyCreatures[currentEnemyID]))
            {
                System.out.println(getName(playerCreature)+" defeated the "+␣
↪getEnemySpecies(enemyCreatures[currentEnemyID]));
                return 1;
            }
            else if (getPlayerMagic(playerCreature)<␣
↪getEnemyMagic(enemyCreatures[currentEnemyID]))
            {
                System.out.println(getName(playerCreature)+" was defeated in␣
↪combat with the "+ getEnemySpecies(enemyCreatures[currentEnemyID]));
                System.out.println("You managed to escape while␣
↪"+getName(playerCreature)+" became a meal.");
                return -1;
            }
            else
            {
                System.out.println(getName(playerCreature)+ " and the " +␣
↪getEnemySpecies(enemyCreatures[currentEnemyID]) +" were evenly matched.");
                System.out.println("The two of you managed to escape the␣
↪encounter.");
                return 0;
            }
        }

        //separated from rest of the method so it should only run in the case␣
↪of valid input
        //checks value of the "sruvived" boolean passsed
        //player's progress is either reset or the game is lost
        public static PlayerCreature checkForSurvival(boolean survived,␣
↪PlayerCreature pc)
        {
            if (survived)
            {
                System.out.println("You survived the encounter but your␣
↪"+getPlayerSpecies(pc)+ " ran away. \n Time to start again.");
                pc=createPlayerCreature(inputSpecies(), inputName());
            }
            else
            {
```

```
            pc=setGameOver(pc, true);
        }
        return pc;
    }

    //separated from main, runs when the creature goes rabid(fun or hunger␣
↪stat <=-10)
    public static PlayerCreature creatureRabid(PlayerCreature␣
↪playerCreature, int magicMax)
    {
        String answer=displayCreatureAttackCommands(playerCreature).
↪toLowerCase();
        boolean survived=false;

       if (answer.equals("fight back"))
        {
            playerCreature=checkForSurvival(fightBack(playerCreature,␣
↪magicMax), playerCreature);
        }
        else if (answer.equals("run"))
        {
            playerCreature=checkForSurvival(runAway() , playerCreature);
        }
        else
        {
            System.out.println("Could not recognise command.");
        }

        return playerCreature;
    }

    //regular round, separated from main()
    //runs appropriate methods based on the action the player input
    public static PlayerCreature regularRound(PlayerCreature␣
↪playerCreature, EnemyCreature[] enemies) throws IOException
    {
        String answer=displayActions().toLowerCase();
        int battleResult;
        if (answer.equals("feed"))
        {
            playerCreature=feed(playerCreature);
            playerCreature=nextRound(playerCreature, false, true);
        }
        else if (answer.equals("play"))
        {
            playerCreature=play(playerCreature);
            playerCreature=nextRound(playerCreature, true, false);
```

```
            }
            else if (answer.equals("train"))
            {
                playerCreature=train(playerCreature);
                playerCreature=nextRound(playerCreature, true, true);
            }
            else if (answer.equals("check stats"))
            {
               displayStatus(playerCreature);
            }
            else if (answer.equals("change name"))
            {
                playerCreature=setName(playerCreature, inputName());
            }
            else if (answer.equals("new creature"))
            {
                System.out.println("You abandoned"+getName(playerCreature)+".␣
↪You monster.");
                playerCreature=createPlayerCreature(inputSpecies(),␣
↪inputName());
            }
            else if (answer.equals("battle"))
            {
                answer="y";
                while (!answer.equals("n") && getCurrentEnemy(playerCreature)<5␣
↪&& answer.equals("y"))
                {
                    battleResult= battleEnemy(playerCreature, enemies,␣
↪getCurrentEnemy(playerCreature));
                    if(battleResult==1)
                    {
                        playerCreature=setCurrentEnemy(playerCreature,1);
                        if (getCurrentEnemy(playerCreature)>=enemies.length)//
↪game is won, all enemies defeated, breaks out the loop
                        {
                            playerCreature=setGameOver(playerCreature, true);
                            playerCreature=setGameWon(playerCreature, true);
                        }
                        else
                        {
                            answer=inputString("Would you like to battle the␣
↪next enemy?(y/n)");
                            playerCreature=nextRound(playerCreature, true,␣
↪true);
                        }
```

```
                    }
                    else if (battleResult==-1)//loss = reset
                    {
                        playerCreature=createPlayerCreature(inputSpecies(),␣
↪inputName());
                        answer="n";
                    }
                    else//draw, can only be 0(no point in extra check)
                    {
                        playerCreature=nextRound(playerCreature, true, true);
                        answer=inputString("Would you like to battle the next␣
↪enemy?(y/n)");
                    }

                }
            }

            else if (answer.equals("save"))
            {
                saveGame(playerCreature);
            }
            else if (answer.equals("load"))
            {
                if(loadGame()==null)
                {
                    System.out.println("Save file was empty, no creature can be␣
↪retrieved.");
                }
                else
                {
                    playerCreature=loadGame();
                }
            }

            else if (answer.equals("quit"))
            {
                playerCreature=setGameOver(playerCreature, true);
            }

            else
            {
                System.out.println("Could not recognise command.");
            }
            return playerCreature;

        }
```

```
       public static void main(String[] a) throws IOException
       {
           PlayerCreature playerCreature=new PlayerCreature();
           final int magicMax=50;

          EnemyCreature[] enemyCreatures= initiateEnemies(magicMax);
          playerCreature=createPlayerCreature(inputSpecies(), inputName());

           while(!getGameOver(playerCreature))
               {
                   if (getHunger(playerCreature)<-10 | getFun(playerCreature)<-10)// creature goes rabid
                   {
                       playerCreature=creatureRabid(playerCreature, magicMax);
                   }

                   else//regular turn
                   {
                       playerCreature=regularRound(playerCreature,enemyCreatures);
                   }


               }

           printFinalMessage(getGameWon(playerCreature), playerCreature);

           return;

       }
   }
```

2.8.1 Running the program

Run the following call to simulate running the complete program.

[25]:
```
creatureCare.main(null);
```

```
What species is your creature?
 unicorn
Are you sure? You cannot change this later.(y/n)
 y
Name you creature.(This can be changed later)
 Jim
```

```
>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit


 train

You trained Jim in the arcane.


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit


 train

You trained Jim in the arcane.


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit


 train

You trained Jim in the arcane.
```

```
>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit


 train

You trained Jim in the arcane.


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit


 check stats



Name: Jim
Hunger: -7
Fun: -8
Magic: 20


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
```

```
>Quit

 load
Save file was empty, no creature can be retrieved.


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit

 save


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit

 train
You trained Jim in the arcane.


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit
```

```
 train
You trained Jim in the arcane.


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit

 train
You trained Jim in the arcane.


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit

 train
You trained Jim in the arcane.


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit
```

```
 train
```

```
You trained Jim in the arcane.


Your unicorn is unhappy with your care.
>Run
>Fight back
```

```
 run
```

```
You died!
 GAME OVER
```

[29]:
```
creatureCare.main(null);
```

```
What species is your creature?
```

```
 goblin
```

```
Are you sure? You cannot change this later.(y/n)
```

```
 n
```

```
What species is your creature?
```

```
 goblin
```

```
Are you sure? You cannot change this later.(y/n)
```

```
 ljdnfldnblndlfbn
```

```
Are you sure? You cannot change this later.(y/n)
```

```
 y
```

```
Name you creature.(This can be changed later)
```

```
 bob
```

```
>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit
```

```
 check stats


Name: bob
Hunger: 0
Fun: 0
Magic: 0


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit

 load


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit

 check stats


Name: Jim
Hunger: -7
Fun: -8
Magic: 20


>Feed
>Play
```

```
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit

 play

You played fetch with Jim.


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit

 feed

You fed Jim.


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit

 check stats


Name: Jim
Hunger: -6
Fun: -8
```

```
Magic: 20


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit

 play
You played fetch with Jim.


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit

 check stats


Name: Jim
Hunger: -8
Fun: -5
Magic: 20


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
```

```
>Load
>Quit
```

```
 feed
```

```
You fed Jim.
```

```
>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit
```

```
 feed
```

```
You fed Jim.
```

```
>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit
```

```
 play
```

```
You played fetch with Jim.
```

```
>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
```

```
>Load
>Quit
```

```
 change name
```

```
Name you creature.(This can be changed later)
```

```
 Ric
```

```
>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit
```

```
 check stats
```

```
Name: Ric
Hunger: -3
Fun: -3
Magic: 20
```

```
>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit
```

```
 new creature
```

```
You abandonedRic. You monster.
What species is your creature?
```

```
 Goblin
```

```
Are you sure? You cannot change this later.(y/n)
 y
Name you creature.(This can be changed later)
 Bob


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit

 check stats


Name: Bob
Hunger: 0
Fun: 0
Magic: 0


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit

 load


>Feed
>Play
>Train
>Check stats
```

```
>Change name
>New creature
>Battle
>Save
>Load
>Quit

 check stats


Name: Jim
Hunger: -7
Fun: -8
Magic: 20


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit

 play

You played fetch with Jim.


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit

 feed

You fed Jim.
```

```
>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit

 play

You played fetch with Jim.


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit

 feed

You fed Jim.


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit

 play

You played fetch with Jim.
```

```
>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit

 feed

You fed Jim.


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit

 check stats


Name: Jim
Hunger: -3
Fun: 0
Magic: 20


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit
```

```
 feed
You fed Jim.


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit

 feed
You fed Jim.


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit

 play
You played fetch with Jim.


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit
```

```
 train
You trained Jim in the arcane.


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit

 rain
Could not recognise command.


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit

 train
You trained Jim in the arcane.


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit
```

```
 train

You trained Jim in the arcane.


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit


 train

You trained Jim in the arcane.


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit


 check stats


Name: Jim
Hunger: -6
Fun: -7
Magic: 40


>Feed
>Play
>Train
>Check stats
>Change name
```

```
>New creature
>Battle
>Save
>Load
>Quit

 save


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit

 play

You played fetch with Jim.


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit

 feed

You fed Jim.


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
```

```
>Battle
>Save
>Load
>Quit

 play

You played fetch with Jim.


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit

 feed

You fed Jim.


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit

 check stats


Name: Jim
Hunger: -2
Fun: -5
Magic: 40


>Feed
```

```
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit


 train

You trained Jim in the arcane.


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit


 train

You trained Jim in the arcane.


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit


 battle

Jim defeated the Goblin
Would you like to battle the next enemy?(y/n)

 y
```

```
Jim defeated the Siren
Would you like to battle the next enemy?(y/n)
 n


Your unicorn is unhappy with your care.
>Run
>Fight back


 run
You survived the encounter but your unicorn ran away.
 Time to start again.
What species is your creature?
 pheonix
Are you sure? You cannot change this later.(y/n)
 y
Name you creature.(This can be changed later)
 carl



>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit


 load



>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
```

```
>Load
>Quit

 check stats


Name: Jim
Hunger: -6
Fun: -7
Magic: 40


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit

 train

You trained Jim in the arcane.


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit

 train

You trained Jim in the arcane.


>Feed
>Play
>Train
```

```
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit


 play

You played fetch with Jim.


Your unicorn is unhappy with your care.
>Run
>Fight back


 fight back

You died!
 GAME OVER
```

[30]:
```
creatureCare.main(null);
```

```
What species is your creature?

 t

Are you sure? You cannot change this later.(y/n)

 y

Name you creature.(This can be changed later)

 t



>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit


 load
```

```
>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit

 check stats


Name: Jim
Hunger: -6
Fun: -7
Magic: 40


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit

 play
You played fetch with Jim.


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
```

```
>Quit


 feed

You fed Jim.


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit


 play

You played fetch with Jim.


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit


 feed

You fed Jim.


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
```

```
>Quit

 play
You played fetch with Jim.


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit

 feed
You fed Jim.


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit

 check stats


Name: Jim
Hunger: -1
Fun: -3
Magic: 40


>Feed
>Play
>Train
>Check stats
```

```
>Change name
>New creature
>Battle
>Save
>Load
>Quit


 train

You trained Jim in the arcane.


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit


 train

You trained Jim in the arcane.


>Feed
>Play
>Train
>Check stats
>Change name
>New creature
>Battle
>Save
>Load
>Quit


 train

You trained Jim in the arcane.


>Feed
>Play
>Train
>Check stats
```

```
>Change name
>New creature
>Battle
>Save
>Load
>Quit


 battle

Jim defeated the Goblin
Would you like to battle the next enemy?(y/n)

 y

Jim defeated the Siren
Would you like to battle the next enemy?(y/n)

 y

Jim defeated the Minotaur
Would you like to battle the next enemy?(y/n)

 y

Jim defeated the Giant
Would you like to battle the next enemy?(y/n)

 y

Jim defeated the Dragon
YOU WON! Congratulations Jim is the strongest magical beast.
```

END OF LITERATE DOCUMENT