MiniProjectV3-Level7

December 17, 2021

1 Mini Project

2 Level 3

2.1 Molla Fahad Kolim

2.2 7 December 2021

2.3 Version 1

2.4 Summary of the Question

Create a program that simulates a real pet that the user has to look after.

2.4.1 PetStatus()

What it does Stating that there's a new object class called PetStatus

Implementation (how it works) Create a new class of objects called PetStatus and state information that you wish you keep in that class - such as name, species, hunger and happiness levels.

[4]:
```
class PetStatus{
    //Record Class holding virtual pets details
    String PetName;
    String PetSpecies;
    Integer PetHunger;
    Integer PetHappiness;
}
```

2.4.2 CompleteProgram()

What it does Method asks user how many pets they want, then asks for the pets information eg. name, species

Implementation (how it works) Starts off by making a new object of PetStatus called Pet1, then asks how many pets the user wants. The input will then be used to initialize the array length, from which the user will be prompted to enter pet name/species and output it to user after being saved into array.

[8]:
```
public static void CompleteProgram(){
        //Main method initializing arrays and calling other methods necessary␣
↪to make program work
        Scanner scan = new Scanner(System.in);
        PetStatus Pet1 = new PetStatus();
        //Creates new instance of record called Pet1, which we use to store pet␣
↪details in and call all over the code
        IntroductionMessage();
        //Prints an introductory message telling user the point of the game.
        String[] names;
        String[] species;
        Integer[] Hunger;
        Integer[] Happiness;
        //Stating arrays that are going to be used later on
        System.out.println("How many pets would you like? ");
        final Integer numberofpets = scan.nextInt();
        //Takes user input and saves as a final Integer, meaning the␣
↪numberofpets variable cannot be changed after entered
        names = new String[numberofpets];
        species = new String[numberofpets];
        Hunger = new Integer[numberofpets];
        Happiness = new Integer[numberofpets];
        //Initializes the arrays, giving them an array size of variable␣
↪numberofpets
        for(int i = 0; i < numberofpets; i++){
            names[i] = PetName(Pet1, i);
            Hunger[i] = PetHunger(Pet1, Hunger);
            Happiness[i] = pethappiness(Pet1, Happiness);
            PetHungerState(Pet1);
            PetHappinessState(Pet1);
            //For loop - for however many pets the user wants, they will be␣
↪allowed to enter their name, initiate a hunger and happiness level.
        }
        if (numberofpets > 1){
            //Only execute method AlphabeticalSort is user has >1 pets
            AlphabeticalSort(numberofpets, names);
            //Alphabetically sort the inputted pets names and display back to␣
↪the user
        }
        PetRounds(Pet1, names, Hunger, Happiness, numberofpets);
        //Method that controls the rounds. Takes inputs
    }
```

[ ]:
```
CompleteProgram();
```

```
Welcome to the Pet Game! This is a game that immitates a real life pet that you
must look after.
```

```
The rules are simple, you first name however many pets you wish to look after.
You will then have to take care of your pet for 10 rounds, by either inputting
the keywords 'feed' or 'play'.
This will improve your pets mood, if you survive without ignoring your pets
needs for more than 2 rounds you win the game!

If you ignore your pets needs for more than 2 rounds, than you lose :( Good
Luck!


How many pets would you like?

 1

What would you like to name your 1 pet?

 Alfred

Whats the species of alfred ?

 Dragon

Happy 0th birthday to alfred the dragon

Their hunger level is low, feed them!

Your pet is happy!

Heres a list of all your pets:
Alfred
Which pet would you like to interact with?

 Alfred

Lives left 2

On a level of 1-5, your pets hunger level is currently 4
On a level of 1-5, your pets happiness level is currently 2

Round 1 would you like to feed or play with alfred?

 play

alfred happiness level is now at 3

Heres a list of all your pets:
Alfred
Which pet would you like to interact with?

 alfred

Lives left 2

On a level of 1-5, your pets hunger level is currently 1
On a level of 1-5, your pets happiness level is currently 1
```

```
Round 2 would you like to feed or play with alfred?

 feed

alfred hunger level is now at 2


Heres a list of all your pets:
Alfred
Which pet would you like to interact with?

 alfred

Lives left 1


On a level of 1-5, your pets hunger level is currently 2
On a level of 1-5, your pets happiness level is currently 3


Round 3 would you like to feed or play with alfred?

 play

alfred happiness level is now at 4


Heres a list of all your pets:
Alfred
Which pet would you like to interact with?

 alfred

Lives left 1


On a level of 1-5, your pets hunger level is currently 1
On a level of 1-5, your pets happiness level is currently 4


Round 4 would you like to feed or play with alfred?

 play

alfred happiness level is now at 5


You lost, you didnt take care of your pet properly!
```

2.4.3 getPetName()

What it does Getter, gets a private record and make it publicly available to set new names into

Implementation (how it works) Set it as a public method that takes PetStatus as return location, and return the PetName back to record

[28]:
```
public static String getPetName(PetStatus Pet1){
        return Pet1.PetName;
    } //Getters get record fields to save elements to
```

2.4.4 setPetName()

What it does Setter, used to set an element into the record field which cannot be done by any other method

Implementation (how it works) Same as getter, but instead it returns a value instead of null as a getter

[29]:
```
public static PetStatus setPetName(PetStatus Pet1, String chosenpet){
    Pet1.PetName = chosenpet.substring(0, 1).toUpperCase() + chosenpet.
↪substring(1); //Makes sure when printing out, first letter is capital
    return Pet1;
}   //Returns Pet1 with new element to object PetStatus
```

2.4.5 setPetname()

What it does Setter, used to set an element into the record field which cannot be done by any other method

Implementation (how it works) Same as getter, but instead it returns a value instead of null as a getter

[27]:
```
public static String setpetName(PetStatus Pet1, String petname){
    Pet1.PetName = petname;
    return Pet1.PetName;
}
```

2.4.6 IntroductionMessage()

What it does Prints out an introduction message telling the user how to play the game

2.4.7 Implementation (how it works)

Uses print statements to print out messages telling user how the program works

[12]:
```
public static void IntroductionMessage(){
    //Introduction method telling user what the program does and how to play.
    System.out.println("Welcome to the Pet Game! This is a game that immitates␣
↪a real life pet that you must look after.");
    System.out.println("The rules are simple, you first name however many pets␣
↪you wish to look after.");
    System.out.println("You will then have to take care of your pet for 10␣
↪rounds, by either inputting the keywords 'feed' or 'play'.");
    System.out.println("This will improve your pets mood, if you survive␣
↪without ignoring your pets needs for more than 2 rounds you win the game!");
    System.out.println("If you ignore your pets needs for more than 2 rounds,␣
↪than you lose :( Good Luck!");
    System.out.println("");
```

```
}
```

2.4.8 Testing

[2]:
```
IntroductionMessage();
```

```
Welcome to the Pet Game! This is a game that immitates a real life pet that you
must look after.
The rules are simple, you first name however many pets you wish to look after.
You will then have to take care of your pet for 10 rounds, by either inputting
the keywords 'feed' or 'play'.
This will improve your pets mood, if you survive without ignoring your pets
needs for more than 2 rounds you win the game!
If you ignore your pets needs for more than 2 rounds, than you lose :( Good
Luck!
```

2.4.9 PetName()

What it does Asks user to input the name of their pet and species

Implementation (how it works) Using Scanner library to scan for user input, from which we will save into variables and output to user

[13]:
```
public static String PetName(PetStatus Pet1, int i){
    //Method that allows the user to name their pet and species
    Scanner scan = new Scanner(System.in);
    System.out.println("What would you like to name your " + (i+1) + " pet? ");
    //Asks user what they would like to name their pet, uses 'i' to display the␣
↪pet number they are naming.
    String petname = scan.nextLine().toLowerCase();
    //Saves user input in variable petname of type string
    petname = setpetName(Pet1, petname);
    //Also saves petname into record field PetName
    System.out.println("Whats the species of " + petname + " ?");
    //Asks user for their pets species
    Pet1.PetName = petname;
    String petspecies = scan.nextLine().toLowerCase();
    //Saves user input in variable petspecies of type string
    Pet1.PetSpecies = petspecies;
    //Saves the pets species in record field PetSpecies
    System.out.println("Happy 0th birthday to " + petname + " the " +␣
↪petspecies);
    //Prints a message taking their pets name and species
    System.out.println("");
    return petname;
    //Returns petname
```

```
}
```

Testing

[20]:
```
PetStatus Pet1 = new PetStatus();
int i = 0;
PetName(Pet1, i);
```

```
What would you like to name your 1 pet?

 Alfred

Whats the species of alfred ?

 Dragon

Happy 0th birthday to alfred the dragon
```

[20]:
```
alfred
```

2.4.10 PetHunger()

What it does Randomly generates a hunger rate for the pet and saves into Pet1.PetHunger record field, also returns.

Implementation (how it works) Using Random library to generate a random number representing pets hunger level, and passes it onto a new method outputting pets hunger level as well as returning.

[14]:
```
public static Integer PetHunger(PetStatus Pet1, Integer[] Hunger){
    //Method that randomly generates a hunger value for pets.
    int Low = 1;
    int High = 5;
    //Setting boundaries of generated number, numbers include 1 2 3 4 5
    Random rand = new Random();
    //Random library initiation
    int hunger = rand.nextInt(High-Low)+Low;
    //Randomly generated number gets saved into variable hunger of type Integer
    Pet1.PetHunger = hunger;
    //Saves newly generated hunger value into record field PetHunger
    return hunger;
    //Returns hunger
}
```

Testing

[24]:
```
PetStatus Pet1 = new PetStatus();
Integer[] Hunger = new Integer[1];
```

```
PetHunger(Pet1, Hunger);
```

[24]: 3

2.4.11 PetHungerState()

What it does Takes randomly generated integer, and puts into if statement outputting whichever statement it matches

Implementation (how it works) After passing value of hunger to this method, goes into if loops seeing which one matches, then outputs a message

[24]:
```
public static void PetHungerState(PetStatus Pet1){
    //Rather than outputting a number that doesn't mean much to user, this␣
↪method takes the randomly generated number and prints out a statement instead
    //If loops that prints different messages according to whatever the␣
↪randomly generated number is
    if (Pet1.PetHunger == 5){
        System.out.println("Your pet is really hungry, feed them!");
        System.out.println("");
    }
    if (Pet1.PetHunger == 4){
        System.out.println("Their hunger level is low, feed them!");
        System.out.println("");
    }
    if (Pet1.PetHunger == 3){
        System.out.println("Their hunger level is ok, feed them soon!");
        System.out.println("");
    }
    if (Pet1.PetHunger == 2){
        System.out.println("Your pets' hunger is full");
        System.out.println("");
    }
    if (Pet1.PetHunger == 1){
        System.out.println("Your pet is bloated, let them rest.");
        System.out.println("");
    }
}
```

Testing

[31]:
```
PetStatus Pet1 = new PetStatus();
Pet1.PetHunger = 3;

PetHungerState(Pet1);
```

```
Their hunger level is ok, feed them soon!
```

2.4.12 PetHappiness()

What it does Randomly generates a happiness rate for the pet

Implementation (how it works) Using Random library to generate a random number representing pets happiness level, and passes it onto a new method outputting pets happiness level

[16]:
```
public static Integer pethappiness(PetStatus Pet1, Integer[] Happiness){
    //Method generates a random happiness value for pet
    int Low = 1;
    int High = 5;
    //Set boundaries dictating the range of the randomly generated value
    Random rand1 = new Random();
    int happiness = rand1.nextInt(High-Low)+Low;
    //Randomly generated value gets saved into variable happiness
    Pet1.PetHappiness = happiness;
    //Saves randomly generated value into record field PetHappiness
    return happiness;
    //Returns happiness
}
```

Testing

[34]:
```
PetStatus Pet1 = new PetStatus();
Integer[] Happiness = new Integer[1];

pethappiness(Pet1, Happiness);
```

[34]:
```
3
```

2.4.13 PetHappinessState()

What it does Takes randomly generated number as input, goes into if statment and prints out whatever statement it matches

Implementation (how it works) After passing randomly generated integer into method, goes into if loop checking what statement it matches and prints statement

[17]:
```
public static void PetHappinessState(PetStatus Pet1){
    //Rather than outputting a number that doesn't mean much to user, this␣
 ↪method takes the randomly generated number and prints out a statement instead
    //If loops that prints different messages according to whatever the␣
 ↪randomly generated number is
    if (Pet1.PetHappiness == 5 ){
        System.out.println("Your pet is really sad, play with them!");
        System.out.println("");
    }
    if (Pet1.PetHappiness == 4 ){
```

```
        System.out.println("Your pet is bored, play with them!");
        System.out.println("");
    }
    if (Pet1.PetHappiness == 3 ){
        System.out.println("Your pet is kinda bored, play with them if you have␣
↪time!");
        System.out.println("");
    }
    if (Pet1.PetHappiness == 2 ){
        System.out.println("Your pet is happy!");
        System.out.println("");
    }
    if (Pet1.PetHappiness == 1 ){
        System.out.println("Your pet is really happy!");
        System.out.println("");
    }
}
```

Testing

[36]:
```
PetStatus Pet1 = new PetStatus();
Pet1.PetHappiness = 3;

PetHappinessState(Pet1);
```

```
Your pet is kinda bored, play with them if you have time!
```

2.4.14 PetRounds()

What it does After getting all necessary variables passed to method, runs a for loop 10 times imitating rounds which they player must survive. Prints out hunger and happiness state of pet every round and lives left. Player then chooses their next pet to look after/action.

Implementation (how it works) After having arguments passed to it, goes into for loop for 10 rounds, adding 1 on to i every time its ran. Gets Pet1 record values and outputs them, awaits for user input and adjusts counter/round accordingly. New values are then made for hunger and happiness from which it then goes into loop again. If counter = 0 program ends else if loop i < 11 user wins program.

[33]:
```
public static void PetRounds(PetStatus Pet1, String[] names, Integer[] Hunger,␣
↪Integer[] Happiness, int numberofpets){
    //Core method for program to work, loops the rounds and outputs whether␣
↪they won or lost.
    Integer counter = 2;
    //Stating the counter will always start from 2, users can only make 2␣
↪mistakes before the program ends and displays 'you lost' message
    for(int i = 1; i < 11; i++){
```

```
        //For loops that runs a total of 10 times, meaning the user has to play␣
↪the game for 10 rounds in order to win. For every itteration i gets +1,␣
↪stopping when i < 11
        PetChooser(Pet1, i, names);
        //Calls PetChooser method, allows user to choose which pet to look after

        Scanner scan = new Scanner(System.in);
        System.out.println("Lives left " + counter);
        //Prints the status of counter
        System.out.println("");
        System.out.println("On a level of 1-5, your pets hunger level is␣
↪currently " + Pet1.PetHunger);
        System.out.println("On a level of 1-5, your pets happiness level is␣
↪currently " + Pet1.PetHappiness);
        //Prints out the hunger and happiness status of the pet
        System.out.println("");
        if(Pet1.PetHunger.equals(5)){
            //If the randomly generated number is 5, and because we count pet␣
↪status on a level from 1 to 5, we output an automated message.
            System.out.println("Round " + i + " would you like to feed or play␣
↪with " + getPetName(Pet1) + "? ");
            System.out.println("Because " + getPetName(Pet1) + "s' current␣
↪hunger level is full, you can only play with them this round. ");
            //Prints out that cannot go over 5, so +1 on other value
            Pet1.PetHappiness ++;
            //Adds 1 on to the other value and saves into record
            if(counter == 2){
                //If counter is 2, do nothing. Because we are only giving user␣
↪2 chances to fail counter cannot go over 2
                ;
            } else{
                counter = counter + 1;
                //If counter is not 2, add 1 on to the counter
            }
            System.out.println(getPetName(Pet1) + "'s happiness level is now "␣
↪+ Pet1.PetHappiness);
            //Print a message stating the current status of the pets hunger and␣
↪happiness level
            System.out.println("");
            i++;
            //Adds 1 on to the counter to say 1 round has been completed
            System.out.println("");
            int Low = 1;
            int High = 4;
            Random rand1 = new Random();
            int happiness = rand1.nextInt(High-Low)+Low;
```

```
            //Makes new range for randomly generated values, done as a␣
↪precaution in case the randomly generated value is 5. If so cannot enter the␣
↪loop again and will allow level to go over 5
            //Generates random number between 1-5
            Pet1.PetHappiness = happiness;
            //Saves happiness into record field PetHappiness
            System.out.println("Lives left " + counter);
            //Prints out the number of lives the user has left
            PetHunger(Pet1, Hunger);
            pethappiness(Pet1, Happiness);
            //Calls methods to randomly generated values with new limits
            System.out.println("On a level of 1-5, your pets hunger level is␣
↪currently " + Pet1.PetHunger);
            System.out.println("On a level of 1-5, your pets happiness level is␣
↪currently " + Pet1.PetHappiness);
            //Prints out the newly generated values as pets new status levels.
        }

        if(Pet1.PetHappiness.equals(5)){
            //This one is similar to the if loop above, but this is for␣
↪Happiness level instead of hunger
            System.out.println("Round " + i + " would you like to feed or play␣
↪with " + getPetName(Pet1) + "? ");
            System.out.println(getPetName(Pet1) + "s' current happiness level␣
↪is full, you can only feed them this round. ");
            Pet1.PetHunger ++;
            System.out.println(getPetName(Pet1) + "s' hunger level is now " +␣
↪Pet1.PetHunger);
            if(counter == 2){
                ;
            } else{
                counter = counter + 1;
            }
            System.out.println(counter);
            System.out.println("");
            i++;
            //Same concept as explained above
            System.out.println("");
            int Low = 1;
            int High = 4;
            Random rand1 = new Random();
            int happiness = rand1.nextInt(High-Low)+Low;
            //Generates random number between 1-5
            Pet1.PetHappiness = happiness;
            //Saves happiness into record pet1
            System.out.println("Lives left " + counter);
```

```
            PetHunger(Pet1, Hunger);
            pethappiness(Pet1, Happiness);
            System.out.println("On a level of 1-5, your pets hunger level is␣
↪currently " + Pet1.PetHunger);
            System.out.println("On a level of 1-5, your pets happiness level is␣
↪currently " + Pet1.PetHappiness);
        }
        System.out.println("Round " + i + " would you like to feed or play with␣
↪" + getPetName(Pet1) + "? ");
        //If randomly generated value is not 5, then output round number and␣
↪use ADT to getPetName
        String Action = scan.nextLine().toLowerCase();
        //Awaits for user input - can either be feed or play and saves into␣
↪variable Action of type String
        if (Action.equals("play")){
            //If user input is 'play', jump into this if loop
            if (Pet1.PetHunger == 1 & (Action != "feed")){
                counter = counter - 1;
            } //If user input is 'play' whilst PetHunger value is 1 (lowest␣
↪possible level) then -1 from counter
            if (Pet1.PetHappiness == 1 & (Action.equals("play"))){
                if(counter == 2){
                    ; //If counter is 2, do nothing. As we dont want counter to␣
↪>2
                } else{
                    counter = counter + 1;
                } //Else if user input is  'play' whilst PetHappiness is 1,␣
↪then +1 to counter
            }
            if (Pet1.PetHunger == 1 & Pet1.PetHappiness == 1){
                counter = counter - 1;
            } //If both randomly generated values are 1, then -1 from counter␣
↪regardless of user input
            Pet1.PetHappiness ++;
            //Adds +1 to value held in PetHappiness
            System.out.println(getPetName(Pet1) + " happiness level is now at "␣
↪+ Pet1.PetHappiness);
            //Prints out current state of pet using ADT getPetName
            PetHunger(Pet1, Hunger);
            pethappiness(Pet1, Happiness);
            //Randomly generates new values of Hunger and Happiness
            System.out.println("");
        }
        if (Action.equals("feed")){
            //Same concept as explained above, but this if loop is inversed.␣
↪Top was for happiness whilst this is for hunger
```

```
            if (Pet1.PetHappiness == 1 & (Action != "play")){
                counter = counter - 1 ;
            }
            if (Pet1.PetHunger == 1 & (Action.equals("feed"))){
                if(counter == 2){
                    ;
                } else{
                    counter = counter + 1;
                }
            }
            if (Pet1.PetHunger == 1 & Pet1.PetHappiness == 1){
                counter = counter - 1;
            }
            Pet1.PetHunger ++;
            System.out.println(getPetName(Pet1) + " hunger level is now at " +␣
↪Pet1.PetHunger);
            PetHunger(Pet1, Hunger);
            pethappiness(Pet1, Happiness);
            System.out.println("");
        }
        if (counter == 0){
            System.out.print("You lost, you didnt take care of your pet␣
↪properly!");
            System.exit(0);
        }

        if(!Action.equals("feed") && !Action.equals("play")){
            System.out.println("Action not recognised, try again.");
            i--;

        }

    }
    ProgramComplete();
    //If user gets past 10 rounds regardless of lives left in counter, prints␣
↪out a congratulation message and exits program.
}
```

2.4.15 Testing

[ ]:
```
PetStatus Pet1 = new PetStatus();
String[] names = {"jeff", "bezos"};
PetHunger(Pet1, Hunger);
pethappiness(Pet1, Happiness);
int numberofpets = 2;
```

```
PetRounds(Pet1, names, Hunger, Happiness, numberofpets);
```

```
Heres a list of all your pets:
Jeff
Bezos
Which pet would you like to interact with?
 jeff
Lives left 2

On a level of 1-5, your pets hunger level is currently 4
On a level of 1-5, your pets happiness level is currently 3

Round 1 would you like to feed or play with Jeff?
 play
Jeff happiness level is now at 4

Heres a list of all your pets:
Jeff
Bezos
Which pet would you like to interact with?
 bezos
Lives left 2

On a level of 1-5, your pets hunger level is currently 1
On a level of 1-5, your pets happiness level is currently 4

Round 2 would you like to feed or play with Bezos?
 play
Bezos happiness level is now at 5

Heres a list of all your pets:
Jeff
Bezos
Which pet would you like to interact with?
 jeff
Lives left 1

On a level of 1-5, your pets hunger level is currently 4
On a level of 1-5, your pets happiness level is currently 4

Round 3 would you like to feed or play with Jeff?
 play
```

```
Jeff happiness level is now at 5

Heres a list of all your pets:
Jeff
Bezos
Which pet would you like to interact with?

 bezos

Lives left 1

On a level of 1-5, your pets hunger level is currently 1
On a level of 1-5, your pets happiness level is currently 2

Round 4 would you like to feed or play with Bezos?

 play

Bezos happiness level is now at 3

You lost, you didnt take care of your pet properly!
```

2.4.16 PetChooser()

What it does If user has multiple pets, lets user choose with pet to look after that round, if not found then prints error message and asks again.

Implementation (how it works) Using setters and getters, it first asks for user input - which pet they wish to look after - from which its then saved into a variable. The variable is then passed to a setter where it writes over record field PetName, the PetName is then called above to display to user, if loop used in case user inputs a pet name thats not recognised, in that case the program will ask for pet name again til 1 is recognised.

[19]:
```
public static void PetChooser(PetStatus Pet1, int numberofpets, String[] names){

    System.out.println("Heres a list of all your pets: ");
    for(int i = 0; i < names.length; i++){
        System.out.println(names[i].substring(0, 1).toUpperCase() + names[i].
↪substring(1));
    }
    //Method that allows user to choose which pet they wish to look after if␣
↪theres multiple
    Scanner scan = new Scanner(System.in);
    System.out.println("Which pet would you like to interact with? ");
    //Asks which pet the user wishes to look after
    String chosenpet = scan.nextLine().toLowerCase();
    //Takes user input thats saved into variable chosenpet, and passes to␣
↪method setPetName. Sets new name for field PetName
    String tovalidate = chosenpet;
    //Transfers chosenpet to tovaliadete
```

```
    if(Arrays.toString(names).contains(tovalidate)){
        Pet1 = setPetName(Pet1, chosenpet);
    } else{
        System.out.println("Pet doesnt exist");
        PetChooser(Pet1, numberofpets, names);
    }
    //If loops to make sure user input pet exists else ask them to input it␣
 ↪again
}
```

2.4.17 Testing

[198]:
```
PetStatus Pet1 = new PetStatus();
int numberofpets = 1;
String[] names = {"jeff"};



PetChooser(Pet1, numberofpets, names);
```

```
Heres a list of all your pets:
Jeff
Which pet would you like to interact with?
```

```
 jeff
```

2.4.18 AlphabeticalSort()

What it does After user inputs pet names, alphabetically sorts the names and outputs to user

Implementation (how it works) Using bubble sort algorithm, gets user input of pet names and alphabetically sorts them. Does so by using .compateTo() method in java. If found to be greater saves value into temporary variable, swaps values and then uses temporary variable to store value in its new location

[20]:
```
public static void AlphabeticalSort(int numberofpets, String[] names){
        //Bubble sort sorting method
        String temporary;
        //Makes new temporary string to save values in when sorting
        for(int i = 0; i < numberofpets; i++)
        {
            for(int y = i + 1; y < numberofpets; y++){
                if(names[i].compareTo(names[y])> 0)
                {//Compares names of index i, to the value next to it - names␣
 ↪of index y
                    temporary = names[i];
                    names[i] = names[y];
                    names[y]= temporary;
```

```
                    //If its greater, swap names[i] into temporary variable.␣
↪Swap values over and lastly save the value in temporary in names[y]
                }
            }
        }
        System.out.println("Heres a list of all your pets in alphabetical order:
↪ ");
        for(int i = 0; i <= numberofpets - 1; i++){
            System.out.println(names[i].substring(0, 1).toUpperCase() +␣
↪names[i].substring(1));
        }//Capitalize first letter when printed
        //Prints out the sorted list for user to see.

    }
```

2.4.19 Testing

[203]:
```
int numberofpets = 3;
String[] names = {"Jeff", "Bonzo", "Cooty"};



AlphabeticalSort(numberofpets, names);
```

```
Heres a list of all your pets in alphabetical order:
Bonzo
Cooty
Jeff
```

2.4.20 ProgramComplete()

What it does After user survives 10 rounds print a congratulation message

Implementation (how it works) Use print statements to print out that the user won the game

[31]:
```
public static void ProgramComplete(){
        System.out.println("Congratulation you won! You succesfully survived 10␣
↪rounds.");
        System.out.println("The game will now close, goodbye.");
    }
```

2.4.21 Testing

[32]:
```
ProgramComplete();
```

```
Congratulation you won! You succesfully survived 10 rounds.
The game will now close, goodbye.
```

2.5 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.

[204]:
```
//Molla Fahad Kolim
//Virtual Pet Simulator

import java.util.Scanner;
import java.util.Random;
import java.util.Arrays;
//Imports libraries used in code

class PetStatus{
    //Record Class holding virtual pets details
    String PetName;
    String PetSpecies;
    Integer PetHunger;
    Integer PetHappiness;
}

class Main{
    public static void main(String[] args){
        CompleteProgram();
        System.exit(0);

    }

    public static void CompleteProgram(){
        //Main method initializing arrays and calling other methods necessary␣
↪to make program work
        Scanner scan = new Scanner(System.in);
        PetStatus Pet1 = new PetStatus();
        //Creates new instance of record called Pet1, which we use to store pet␣
↪details in and call all over the code
        IntroductionMessage();
        //Prints an introductory message telling user the point of the game.
        String[] names;
        String[] species;
        Integer[] Hunger;
        Integer[] Happiness;
        //Stating arrays that are going to be used later on
        System.out.println("How many pets would you like? ");
        final Integer numberofpets = scan.nextInt();
        //Takes user input and saves as a final Integer, meaning the␣
↪numberofpets variable cannot be changed after entered
        names = new String[numberofpets];
        species = new String[numberofpets];
```

```
        Hunger = new Integer[numberofpets];
        Happiness = new Integer[numberofpets];
        //Initializes the arrays, giving them an array size of variable␣
↪numberofpets
        for(int i = 0; i < numberofpets; i++){
            names[i] = PetName(Pet1, i);
            Hunger[i] = PetHunger(Pet1, Hunger);
            Happiness[i] = pethappiness(Pet1, Happiness);
            PetHungerState(Pet1);
            PetHappinessState(Pet1);
            //For loop - for however many pets the user wants, they will be␣
↪allowed to enter their name, initiate a hunger and happiness level.
        }
        if (numberofpets > 1){
            //Only execute method AlphabeticalSort is user has >1 pets
            AlphabeticalSort(numberofpets, names);
            //Alphabetically sort the inputted pets names and display back to␣
↪the user
        }
        PetRounds(Pet1, names, Hunger, Happiness, numberofpets);
        //Method that controls the rounds. Takes inputs
    }
    //ADT PetStatus
    //getPetName, setPetName

    public static String getPetName(PetStatus Pet1){
        return Pet1.PetName;
    } //Getters get record fields to save elements to
    //ADT, sets record field elements, takes chosenpet as argument and saves␣
↪into record field PetStatus
    //Overwrites currently held name in field PetName with newly user inputted␣
↪value and gets called throughout the program.
    public static PetStatus setPetName(PetStatus Pet1, String chosenpet){
        Pet1.PetName = chosenpet.substring(0, 1).toUpperCase() + chosenpet.
↪substring(1);
        return Pet1;
    }   //Returns Pet1 with new element to object PetStatus
    public static String setpetName(PetStatus Pet1, String petname){
        Pet1.PetName = petname;
        return Pet1.PetName;
    }
    //Introductory message for user to see, explaining program and how to play
    public static void IntroductionMessage(){
        //Introduction method telling user what the program does and how to␣
↪play.
```

```
        System.out.println("Welcome to the Pet Game! This is a game that␣
↪immitates a real life pet that you must look after.");
        System.out.println("The rules are simple, you first name however many␣
↪pets you wish to look after.");
        System.out.println("You will then have to take care of your pet for 10␣
↪rounds, by either inputting the keywords 'feed' or 'play'.");
        System.out.println("This will improve your pets mood, if you survive␣
↪without ignoring your pets needs for more than 2 rounds you win the game!");
        System.out.println("If you ignore your pets needs for more than 2␣
↪rounds, than you lose :( Good Luck!");
        System.out.println("");
    }
    //Takes arrays defined in may as arguments
    public static String PetName(PetStatus Pet1, int i){
        //Method that allows the user to name their pet and species
        Scanner scan = new Scanner(System.in);
        System.out.println("What would you like to name your " + (i+1) + " pet?␣
↪");
        //Asks user what they would like to name their pet, uses 'i' to display␣
↪the pet number they are naming.
        String petname = scan.nextLine();
        //Saves user input in variable petname of type string
        petname = setpetName(Pet1, petname).toLowerCase();
        //Also saves petname into record field PetName
        System.out.println("Whats the species of " + petname + " ?");
        //Asks user for their pets species
        Pet1.PetName = petname;
        String petspecies = scan.nextLine().toLowerCase();
        //Saves user input in variable petspecies of type string
        Pet1.PetSpecies = petspecies;
        //Saves the pets species in record field PetSpecies
        System.out.println("Happy 0th birthday to " + petname + " the " +␣
↪petspecies);
        //Prints a message taking their pets name and species
        System.out.println("");
        return petname;
        //Returns petname
    }

    public static Integer PetHunger(PetStatus Pet1, Integer[] Hunger){
        //Method that randomly generates a hunger value for pets.
        int Low = 1;
        int High = 5;
        //Setting boundaries of generated number, numbers include 1 2 3 4 5
        Random rand = new Random();
        //Random library initiation
```

```
        int hunger = rand.nextInt(High-Low)+Low;
        //Randomly generated number gets saved into variable hunger of type␣
↪Integer
        Pet1.PetHunger = hunger;
        //Saves newly generated hunger value into record field PetHunger
        return hunger;
        //Returns hunger
    }

    public static Integer pethappiness(PetStatus Pet1, Integer[] Happiness){
        //Method generates a random happiness value for pet
        int Low = 1;
        int High = 5;
        //Set boundaries dictating the range of the randomly generated value
        Random rand1 = new Random();
        int happiness = rand1.nextInt(High-Low)+Low;
        //Randomly generated value gets saved into variable happiness
        Pet1.PetHappiness = happiness;
        //Saves randomly generated value into record field PetHappiness
        return happiness;
        //Returns happiness
    }

    public static void PetHungerState(PetStatus Pet1){
        //Rather than outputting a number that doesn't mean much to user, this␣
↪method takes the randomly generated number and prints out a statement instead
        //If loops that prints different messages according to whatever the␣
↪randomly generated number is
        if (Pet1.PetHunger == 5){
            System.out.println("Your pet is really hungry, feed them!");
            System.out.println("");
        }
        if (Pet1.PetHunger == 4){
            System.out.println("Their hunger level is low, feed them!");
            System.out.println("");
        }
        if (Pet1.PetHunger == 3){
            System.out.println("Their hunger level is ok, feed them soon!");
            System.out.println("");
        }
        if (Pet1.PetHunger == 2){
            System.out.println("Your pets' hunger is full");
            System.out.println("");
        }
        if (Pet1.PetHunger == 1){
            System.out.println("Your pet is bloated, let them rest.");
            System.out.println("");
```

```
        }
    }

    public static void PetHappinessState(PetStatus Pet1){
        //Rather than outputting a number that doesn't mean much to user, this␣
↪method takes the randomly generated number and prints out a statement instead
        //If loops that prints different messages according to whatever the␣
↪randomly generated number is
        if (Pet1.PetHappiness == 5 ){
            System.out.println("Your pet is really sad, play with them!");
            System.out.println("");
        }
        if (Pet1.PetHappiness == 4 ){
            System.out.println("Your pet is bored, play with them!");
            System.out.println("");
        }
        if (Pet1.PetHappiness == 3 ){
            System.out.println("Your pet is kinda bored, play with them if you␣
↪have time!");
            System.out.println("");
        }
        if (Pet1.PetHappiness == 2 ){
            System.out.println("Your pet is happy!");
            System.out.println("");
        }
        if (Pet1.PetHappiness == 1 ){
            System.out.println("Your pet is really happy!");
            System.out.println("");
        }
    }

    public static void PetRounds(PetStatus Pet1, String[] names, Integer[]␣
↪Hunger, Integer[] Happiness, int numberofpets){
        //Core method for program to work, loops the rounds and outputs whether␣
↪they won or lost.
        Integer counter = 2;
        //Stating the counter will always start from 2, users can only make 2␣
↪mistakes before the program ends and displays 'you lost' message
        for(int i = 1; i < 11; i++){
            //For loops that runs a total of 10 times, meaning the user has to␣
↪play the game for 10 rounds in order to win. For every itteration i gets +1,␣
↪stopping when i < 11
            PetChooser(Pet1, i, names);
            //Calls PetChooser method, allows user to choose which pet to look␣
↪after
```

```
            Scanner scan = new Scanner(System.in);
            System.out.println("Lives left " + counter);
            //Prints the status of counter
            System.out.println("");
            System.out.println("On a level of 1-5, your pets hunger level is␣
↪currently " + Pet1.PetHunger);
            System.out.println("On a level of 1-5, your pets happiness level is␣
↪currently " + Pet1.PetHappiness);
            //Prints out the hunger and happiness status of the pet
            System.out.println("");
            if(Pet1.PetHunger.equals(5)){
                //If the randomly generated number is 5, and because we count␣
↪pet status on a level from 1 to 5, we output an automated message.
                System.out.println("Round " + i + " would you like to feed or␣
↪play with " + getPetName(Pet1) + "? ");
                System.out.println("Because " + getPetName(Pet1) + "s' current␣
↪hunger level is full, you can only play with them this round. ");
                //Prints out that cannot go over 5, so +1 on other value
                Pet1.PetHappiness ++;
                //Adds 1 on to the other value and saves into record
                if(counter == 2){
                    //If counter is 2, do nothing. Because we are only giving␣
↪user 2 chances to fail counter cannot go over 2
                    ;
                } else{
                    counter = counter + 1;
                    //If counter is not 2, add 1 on to the counter
                }
                System.out.println(getPetName(Pet1) + "'s happiness level is␣
↪now " + Pet1.PetHappiness);
                //Print a message stating the current status of the pets hunger␣
↪and happiness level
                System.out.println("");
                i++;
                //Adds 1 on to the counter to say 1 round has been completed
                System.out.println("");
                int Low = 1;
                int High = 4;
                Random rand1 = new Random();
                int happiness = rand1.nextInt(High-Low)+Low;
                //Makes new range for randomly generated values, done as a␣
↪precaution in case the randomly generated value is 5. If so cannot enter the␣
↪loop again and will allow level to go over 5
                //Generates random number between 1-5
                Pet1.PetHappiness = happiness;
                //Saves happiness into record field PetHappiness
```

```
                System.out.println("Lives left " + counter);
                //Prints out the number of lives the user has left
                PetHunger(Pet1, Hunger);
                pethappiness(Pet1, Happiness);
                //Calls methods to randomly generated values with new limits
                System.out.println("On a level of 1-5, your pets hunger level␣
↪is currently " + Pet1.PetHunger);
                System.out.println("On a level of 1-5, your pets happiness␣
↪level is currently " + Pet1.PetHappiness);
                //Prints out the newly generated values as pets new status␣
↪levels.
            }

            if(Pet1.PetHappiness.equals(5)){
                //This one is similar to the if loop above, but this is for␣
↪Happiness level instead of hunger
                System.out.println("Round " + i + " would you like to feed or␣
↪play with " + getPetName(Pet1) + "? ");
                System.out.println(getPetName(Pet1) + "s' current happiness␣
↪level is full, you can only feed them this round. ");
                Pet1.PetHunger ++;
                System.out.println(getPetName(Pet1) + "s' hunger level is now "␣
↪+ Pet1.PetHunger);
                if(counter == 2){
                    ;
                } else{
                    counter = counter + 1;
                }
                System.out.println(counter);
                System.out.println("");
                i++;
                //Same concept as explained above
                System.out.println("");
                int Low = 1;
                int High = 4;
                Random rand1 = new Random();
                int happiness = rand1.nextInt(High-Low)+Low;
                //Generates random number between 1-5
                Pet1.PetHappiness = happiness;
                //Saves happiness into record pet1
                System.out.println("Lives left " + counter);
                PetHunger(Pet1, Hunger);
                pethappiness(Pet1, Happiness);
                System.out.println("On a level of 1-5, your pets hunger level␣
↪is currently " + Pet1.PetHunger);
```

```
                System.out.println("On a level of 1-5, your pets happiness␣
↪level is currently " + Pet1.PetHappiness);
            }
            System.out.println("Round " + i + " would you like to feed or play␣
↪with " + getPetName(Pet1) + "? ");
            //If randomly generated value is not 5, then output round number␣
↪and use ADT to getPetName
            String Action = scan.nextLine().toLowerCase();
            //Awaits for user input - can either be feed or play and saves into␣
↪variable Action of type String
            if (Action.equals("play")){
                //If user input is 'play', jump into this if loop
                if (Pet1.PetHunger == 1 & (Action != "feed")){
                    counter = counter - 1;
                } //If user input is 'play' whilst PetHunger value is 1 (lowest␣
↪possible level) then -1 from counter
                if (Pet1.PetHappiness == 1 & (Action.equals("play"))){
                    if(counter == 2){
                        ; //If counter is 2, do nothing. As we dont want␣
↪counter to >2
                    } else{
                        counter = counter + 1;
                    } //Else if user input is  'play' whilst PetHappiness is 1,␣
↪then +1 to counter
                }
                if (Pet1.PetHunger == 1 & Pet1.PetHappiness == 1){
                    counter = counter - 1;
                } //If both randomly generated values are 1, then -1 from␣
↪counter regardless of user input
                Pet1.PetHappiness ++;
                //Adds +1 to value held in PetHappiness
                System.out.println(getPetName(Pet1) + " happiness level is now␣
↪at " + Pet1.PetHappiness);
                //Prints out current state of pet using ADT getPetName
                PetHunger(Pet1, Hunger);
                pethappiness(Pet1, Happiness);
                //Randomly generates new values of Hunger and Happiness
                System.out.println("");
            }
            if (Action.equals("feed")){
                //Same concept as explained above, but this if loop is inversed.
↪ Top was for happiness whilst this is for hunger
                if (Pet1.PetHappiness == 1 & (Action != "play")){
                    counter = counter - 1 ;
                }
                if (Pet1.PetHunger == 1 & (Action.equals("feed"))){
```

```
                    if(counter == 2){
                        ;
                    } else{
                        counter = counter + 1;
                    }
                }
                if (Pet1.PetHunger == 1 & Pet1.PetHappiness == 1){
                    counter = counter - 1;
                }
                Pet1.PetHunger ++;
                System.out.println(getPetName(Pet1) + " hunger level is now at␣
↪" + Pet1.PetHunger);
                PetHunger(Pet1, Hunger);
                pethappiness(Pet1, Happiness);
                System.out.println("");
            }
            if (counter == 0){
                System.out.print("You lost, you didnt take care of your pet␣
↪properly!");
                System.exit(0);
            }

            if(!Action.equals("feed") && !Action.equals("play")){
                System.out.println("Action not recognised, try again.");
                i--;

            }

        }
        ProgramComplete();
        //If user gets past 10 rounds regardless of lives left in counter,␣
↪prints out a congratulation message and exits program.
    }

    public static void PetChooser(PetStatus Pet1, int numberofpets, String[]␣
↪names){

        System.out.println("Heres a list of all your pets: ");
        for(int i = 0; i < names.length; i++){
            System.out.println(names[i].substring(0, 1).toUpperCase() +␣
↪names[i].substring(1));
        }
        //Method that allows user to choose which pet they wish to look after␣
↪if theres multiple
        Scanner scan = new Scanner(System.in);
        System.out.println("Which pet would you like to interact with? ");
```

```
        //Asks which pet the user wishes to look after
        String chosenpet = scan.nextLine().toLowerCase();
        //Takes user input thats saved into variable chosenpet, and passes to␣
↪method setPetName. Sets new name for field PetName
        String tovalidate = chosenpet;
        //Transfers chosenpet to tovaliadete
        if(Arrays.toString(names).contains(tovalidate)){
            Pet1 = setPetName(Pet1, chosenpet);
        } else{
            System.out.println("Pet doesnt exist");
            PetChooser(Pet1, numberofpets, names);
        }
        //If loops to make sure user input pet exists else ask them to input it␣
↪again
    }

    public static void AlphabeticalSort(int numberofpets, String[] names){
        //Bubble sort sorting method
        String temporary;
        //Makes new temporary string to save values in when sorting
        for(int i = 0; i < numberofpets; i++)
        {
            for(int y = i + 1; y < numberofpets; y++){
                if(names[i].compareTo(names[y])> 0)
                {//Compares names of index i, to the value next to it - names␣
↪of index y
                    temporary = names[i];
                    names[i] = names[y];
                    names[y]= temporary;
                    //If its greater, swap names[i] into temporary variable.␣
↪Swap values over and lastly save the value in temporary in names[y]
                }
            }
        }
        System.out.println("Heres a list of all your pets in alphabetical order:
↪ ");
        for(int i = 0; i <= numberofpets - 1; i++){
            System.out.println(names[i].substring(0, 1).toUpperCase() +␣
↪names[i].substring(1));
        }//Capitalize first letter when printed
        //Prints out the sorted list for user to see.

    }

    public static void ProgramComplete(){
```

```
        System.out.println("Congratulation you won! You succesfully survived 10␣
↪rounds.");
        System.out.println("The game will now close, goodbye.");
    }


}
```

END OF LITERATE DOCUMENT