mini-project-8

November 26, 2023

0.1 Mini project assessed exercise

0.2 Level 8

0.3 Taher Ahmed

0.4 230288917

0.5 30/12/23

0.6 1

0.7 Summary of the Question

The program outputs to the user, you need to take care of this pet and then it will ask the user please name this pet.

0.8 Justification that the program passes this level

- I can EXPLAIN HOW MY PROGRAM WORKS
- I wrote this program myself
- Is a literate program
- Includes Screen output, Keyboard input
- Defines methods doing a well-defined task and uses a sequence of method calls
- Comment gives at least author's name and student number at start of program
- All the constructs, style and features above AND
- Includes variables, assignment and expressions
- Defines and uses at least one method that returns a result
- Indentation attempted (it may be inconsistently applied)
- Comments gives at least authors name and student number what the program as a whole does.
- All variables are defined inside methods
- Includes Arrays, accessed with a loop
- Includes Loops within loops.
- Defines and uses methods including at least one that is passed and uses array arguments
- Methods individually commented about what they do and all clearly indented.
- Variable declarations are within blocks to reduce scope to a minimum.
- Consistent use of well chosen variable names that give a clear indication of their use (perhaps making comments about them redundant). Consistent use of final variables for literal constants
- Includes records defined as a special class with no methods just field definitions.
- Excellent style over comments, indentation, variable usage, final variables etc.

- Clearly decomposed into multiple small methods doing distinct jobs that take arguments / return results
- BOTH file input AND file output.
- Excellent style over comments, indentation, variable usage, final variables, etc.
- Excellent use of methods throughout
- All variables are defined in methods and have minimum scope
- Includes procedural programming style abstract data type with a clearly commented/specified set of operations, defines and uses accessor methods (all defined in the main class) to access records

0.9 The literate program development

0.10 Method name

pet_intro ## What it does Prints what the objective of the game is. ## Implementation (how it works) Prints messages using the System.out.println funciton.

[ ]:
```
public static void pet_intro () {
  System.out.println ("Welcome to the pet program");
  System.out.println ("You have to look after a pet!");
  return;
}
```

Testing

[ ]:
```
pet_intro();
```

0.11 Method name

get_pet_name ## What it does Asks the user to name the pet. ## Implementation (how it works) Prints a message using the println function, takes an input using scannar. and returns the input.

[ ]:
```
public static String get_string_input_space(String message) {
    Scanner input = new Scanner(System.in);
    String string_input = "-"; // Initializing string_input with a␣
 ↪non-alphabetic character

    while (!string_input.matches("^[a-zA-Z\\s]*$")) { // This regex allows␣
 ↪alphabetic characters and spaces
        System.out.println(message);
        string_input = input.nextLine();

        // Check if the input consists of alphabetic characters and spaces
        if (string_input.matches("^[a-zA-Z\\s]*$")) {
            // If the input is valid, it matches the regex, and you can proceed.
            System.out.println("Yes");
        } else {
            System.out.println("Enter alphabetic characters and spaces only!");
```

```
            // You don't need to consume the newline character here because the loop will continue until valid input is provided.
        }
    }
    return string_input;
}
```

0.12 Testing

[ ]:
```
get_string_input_space ("What would you like to name your pet?");
```

0.13 Method name

set_hunger ## What it does Sets the hunger. Outputs a message displaying the hunger of the pet. ## Implementation (how it works) Inisialises a variable and assigns a value between 1-5 and returns the value.

[ ]:
```
public static int set_hunger_level () {
  int hunger_level;
  Random random = new Random();
  hunger_level = random.nextInt(5) + 1; // Includes variables, assignment and expressions
  return hunger_level;

}
```

0.14 Testing

[ ]:
```
int hunger_level =set_hunger_level();
System.out.println (hunger_level);
```

0.15 Method name

get_hunger_state ## What it does Outputs the state of the pet based on the hunger level. ## Implementation (how it works) Uses if statment, based on the hunger level it assigns different strings to hunger state.

[ ]:
```
public static void get_hunger_state(String [] hunger_descriptions_array, int hunger_level) { //Defines and uses a method that take argument(s)
  // Defines and uses methods including at least one that is passed and uses array arguments

  for (int i =0; i < hunger_level; i++){ // Includes Arrays, accessed with a loop
    if (i==hunger_level-1){ //Includes Decision statements
      System.out.println("Your pet is " + hunger_descriptions_array[i]);
```

```
    }
  }
  return ;
}
```

0.16 Testing

[ ]:
```
String[] hunger_descriptions_array = {"ravenous", "famished", "starving",␣
 ↪"hungry", "full"};
get_hunger_state(hunger_descriptions_array, 3);
```

0.17 Method name

health_check ## What it does Checks if the hunger level is 2 or less and reduces the health level by one. Then the user has option to feed it, hug it, and medicine it to increase the health level. ## Implementation (how it works) Takes hunger_level and health_level as arguments. Uses if statement to check if the hunger level is 2 or less, reduces by the health level using sub. Takes an input using scanner, uses if statement to check what the input is and increases the health level accordingly.

[ ]:
```
// Method to check and update the pet's health based on hunger and user input
public static int health_check(int hunger_level, int health_level) {
  if (hunger_level <= 2) {
      health_level -= 1;
  }
  Scanner input = new Scanner(System.in);
  System.out.println("Would you like to feed it, hug it, give it medicine?");
  String pet_care = input.nextLine();
  while (!pet_care.equals("Feed it") && !pet_care.equals("Hug it") && !pet_care.
 ↪equals("Give it medicine")){
    if (hunger_level == 8) {
        System.out.println("Max health");
    } else if (pet_care.equals("Feed it")) {
        health_level -= 1;
    } else if (pet_care.equals("Give it medicine")) {
        health_level -= 2;
    } else if (pet_care.equals("Hug it")){

    }
    else {
        System.out.println("Invalid input");
    }
    pet_care = input.nextLine();
  }

  return health_level;
}
```

0.18 Testing

[ ]:
```
health_check ( 2, 8 );
```

0.19 Method name

get_pet_record_from_file ## What it does The get_pet_records_from_file function is designed to check if a file with a pet's name exists. If the file exists, it reads the pet's data from the file and creates a Pet record with those values. If the file doesn't exist, it creates a default Pet record and stores it in an array. ## Implementation (how it works) The method first constructs a File object named pet_file based on the pet_name provided, and the pet record file was stored in the format pet name . txt. It checks if the pet_file exists using pet_file.exists(). If it does, it reads the pet's data from the file using a BufferedReader. The data is expected to be in a specific order: pet's name, health level, hunger level, survival status, and the number of rounds. It then calls the create_pet_record method, passing the read data as arguments, and stores the resulting Pet object in the pet_records_array at the specified index j. If the file doesn't exist, it prints a message stating that the pet does not exist and creates a default Pet with default values using the create_pet_record.

[ ]:
```
class Pet { // Includes records defined as a special class with no methods just␣
 ↪field definitions
    String pet_name;
    int health_level;
    int hunger_level;
    boolean survived;
    int num_rounds;
}
public static Pet[] get_pet_records_from_file(Pet[] pet_records_array, String␣
 ↪pet_name, int j) {
    File pet_file = new File(pet_name + ".txt");

    if (pet_file.exists()) {
        try {
            BufferedReader file_reader = new BufferedReader(new␣
 ↪FileReader(pet_file));
            String name = file_reader.readLine();
            int health_level = Integer.parseInt(file_reader.readLine());
            int hunger_level = Integer.parseInt(file_reader.readLine());
            boolean survived = Boolean.parseBoolean(file_reader.readLine());
            int num_rounds = Integer.parseInt(file_reader.readLine());

            // Create a new Pet object and set its fields directly
            Pet my_pet = new Pet();
            my_pet.pet_name = name;
            my_pet.health_level = health_level;
            my_pet.hunger_level = hunger_level;
            my_pet.survived = survived;
```

```
            my_pet.num_rounds = num_rounds;

            pet_records_array[j] = my_pet;
            file_reader.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    return pet_records_array;
}
```

0.20 Testing

[ ]:
```
// Initialize an empty pet_records_array
Pet[] pet_records_array = new Pet[1]; // Assuming you want to store one pet␣
 ↪record.

// Initialize test data
String pet_name = "Taher";
int j = 0;

// Call the method
pet_records_array = get_pet_records_from_file(pet_records_array, pet_name, j);

// Check the result
if (pet_records_array[0] != null) {
    System.out.println("Pet Name: " + pet_records_array[0].pet_name);
    System.out.println("Health Level: " + pet_records_array[0].health_level);
    System.out.println("Hunger Level: " + pet_records_array[0].hunger_level);
    System.out.println("Survived: " + pet_records_array[0].survived);
    System.out.println("Number of Rounds: " + pet_records_array[0].num_rounds);
}
```

0.21 Method name

write_pet_record_to_file ## What it does The write_pet_record_to_file function is responsible for writing a pet's information, including its name, health level, hunger level, survival status, and the number of rounds, to a text file. The file is named after the pet's name and is used for persisting the pet's data. ## Implementation (how it works) It constructs the file name based on the pet's name and creates a PrintWriter to write to the file. It writes the pet's information to the file by printing each field's value as separate lines. Once the data is written, it closes the PrintWriter. If any errors occur during the process, it catches and handles them by printing an error message.

[ ]:
```
class Pet { // Includes records defined as a special class with no methods just␣
 ↪field definitions
  String pet_name;
```

```
  int health_level;
  int hunger_level;
  boolean survived;
  int num_rounds;
}
public static void write_pet_record_to_file(Pet pet) {

  try {
      // Construct the file name based on pet's name
      String file_name = pet.pet_name + ".txt"; // BOTH file input AND file␣
 ↪output.
      PrintWriter output_stream = new PrintWriter(new FileWriter(file_name));
      // Write the pet's information to the file
      output_stream.println(pet.pet_name);
      output_stream.println(pet.health_level);
      output_stream.println(pet.hunger_level);
      output_stream.println(pet.survived);
      output_stream.println (pet.num_rounds);
      output_stream.close(); // Close the PrintWriter when done
  }
  catch (IOException e) {
      System.err.println("Error writing the pet record to a file: " + e.
 ↪getMessage());
  }
}
```

0.22 Testing

[ ]:
```
// Create a Pet object and populate its fields
 Pet pet = new Pet();
 pet.pet_name = "Rofiq";
 pet.health_level = 85;
 pet.hunger_level = 20;
 pet.survived = true;
 pet.num_rounds = 10;

 // Call the method to write the pet's information to a file
 write_pet_record_to_file(pet);

 System.out.println("Pet record has been written to a file.");
```

0.23 Method name

main ## What it does Prints intro messages, takes an input (pet name) and finally outputs a message using the pet name ## Implementation (how it works) Call the pet_intro method, then call get_pet_name method and make a copy of the returned value and prints a message using the println function.

```
[ ]: public static void main (String[] args){ // Defines methods doing a␣
  ↪well-defined task and uses a sequence of method calls
   String pet_name;
   int health_level =8; // Some use of well chosen variable names which give␣
  ↪some information about use
   pet_intro();
   pet_name = get_pet_name ("What would you like to name your pet?");
   System.out.println ("Happy 0th Birthday " + pet_name);
     for (int i=0; i <10 ; i++){
       int hunger_level =set_hunger_level();
       if (health_level ==0) {
         System.out.println("Your pet died");
         break;
       }
       System.out.println("On a scale of 1-5, " + pet_name + "'s hunger rates "␣
  ↪+ hunger_level);
       get_hunger_state(hunger_level);
       health_level = health_check (hunger_level, health_level );
       System.out.println(health_level);
     }
     if (health_level > 0){
       System.out.println("Your pet survivied!");

     }

   return;
}
```

Testing

```
[ ]: main(null);
```

0.23.1 Running the program

Run the following call to simulate running the complete program.

```
[ ]: main(null);
```

0.24 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.

```
[ ]: /* ***************************************
   @author     TAHER AHMED
   @SID        230288917
   @date       26 September 2020
```

```
  @version   1
  @desciption The program outputs to the user, you need to take care of this␣
 ↪pet and then it will ask the user please name this pet. The pet has hunger␣
 ↪level which is a random number between 0-5. It also has health level which␣
 ↪is set to 8. If the health level is less than 3 then hte health level drops␣
 ↪then the user has to feed, hug it or give medicine to the pet. The user has␣
 ↪10 rounds, if the health level does not drop to zero then the pet survived␣
 ↪if not the pet has died.
  The user is how many players are there, and then each player is asked if they␣
 ↪are a returning player if the answer is yes then their pet name is entered␣
 ↪and the info is loaded from an external file, if the answer is no they a new␣
 ↪user is created with default value. Each user has 4 rounds, every round the␣
 ↪pet name is displayed and the pet has hunger level which is a random number␣
 ↪between 0-5. It also has health level which is set to 8. If the health level␣
 ↪is less than 3 then hte health level drops then the user has to feed, hug it␣
 ↪or give medicine to the pet. The user has 10 rounds, if the health level␣
 ↪does not drop to zero then the pet survived if not the pet has died. At the␣
 ↪end of each round the user is asked if they want to terminate the game and␣
 ↪play is latter if yes then the record info is stored in an extenral file for␣
 ↪later and they can stop playing the game and other players can continue to␣
 ↪play their game till the end.
**************************************/
import java.util.Random;
import java.util.Scanner;
import java.io.*;

class Pet { // Includes records defined as a special class with no methods just␣
 ↪field definitions
  String pet_name;
  int health_level;
  int hunger_level;
  boolean survived;
  int num_rounds;
}

class PetProgram {
  // Creates an object and set the default values to whatever is passed in the␣
 ↪arguments and returnes the object
  public static Pet create_pet_record (String pet_name, int health_level, int␣
 ↪hunger_level, boolean survived, int num_rounds){
    Pet my_pet = new Pet ();
    set_name(my_pet, pet_name);
    set_health_level(my_pet, health_level);
    set_hunger_level(my_pet, hunger_level);
    set_survived(my_pet, survived);
    set_num_rounds(my_pet, num_rounds);
    return my_pet;
```

```
}

public static void set_name(Pet pet, String pet_name) { // Includes␣
↪procedural programming style abstract data type with a clearly commented/
↪specified set of operations, defines and uses accessor methods (all defined␣
↪in the main class) to access records.
  pet.pet_name = pet_name;
}

public static void set_health_level(Pet pet, int health_level) {
  pet.health_level = health_level;
}

public static void set_hunger_level(Pet pet, int hunger_level) {
  pet.hunger_level = hunger_level;
}

public static void set_survived(Pet pet, boolean survived) {
  pet.survived = survived;
}

public static void set_num_rounds(Pet pet, int num_rounds) {
  pet.num_rounds = num_rounds;
}
// Getters
public static String get_name(Pet pet) {
  return pet.pet_name;
}

public static int get_health_level(Pet pet) {
  return pet.health_level;
}

public static int get_hunger_level(Pet pet) {
  return pet.hunger_level;
}

public static boolean get_survived(Pet pet) {
  return pet.survived;
}

public static int get_num_rounds (Pet pet){
  return pet.num_rounds;
}
// Outputs a message and takes an input, if the input does not consist of␣
↪alphabetic characters then the user is asked again to enter a value
```

```
public static String get_string_input (String message){ // Defines and uses at least one method that returns a result
  Scanner input = new Scanner(System.in);
  System.out.println(message);
  String string_input = input.nextLine();

  while (!string_input.matches("^[a-zA-Z]*$")) { // Matches method which returns true if the each value matches with one of the value in the array of alpahbetic characters
    System.out.println(message);
    // Check if the input consists of alphabetic characters
    if (!string_input.matches("^[a-zA-Z]*$")) { //Includes Decision statements
      System.out.println("Enter alphabetic characters only!");
    }

    string_input = input.nextLine();
  }

  return string_input;
}
 // Outputs a message and takes an input, if the input does not consist of alphabetic characters and space, an error message is displayed, then the user is asked again to enter a value
public static String get_string_input_space (String message){
  Scanner input = new Scanner(System.in);
  System.out.println(message);
  String string_input = input.nextLine(); // Includes Screen output, Keyboard input
  // Matches method which returns true if the each value matches with one of the value in the array of alpahbetic characters
  while (!string_input.matches("^[a-zA-Z\\s]*$")) {
    string_input = input.nextLine();
    // Check if the input consists of alphabetic characters
    if (!string_input.matches("^[a-zA-Z\\s]*$")) {
      System.out.println("Enter alphabetic characters only!");
    }

  }
  return string_input;
}
// Outputs a message, get an input, if the input is not a positive integer then an error message is displayed and the user is asked again.
public static int get_positive_integer_input(String message) {
  Scanner input = new Scanner(System.in);
  System.out.println(message);
  String str_input = input.nextLine();
```

```
    while(!str_input.matches("^-?\\d+$")){ // if the input is not an integer
      System.out.println("Invalid input. Please enter a whole number:");
      str_input = input.nextLine(); // Take input again
    }

    int integer_input = Integer.parseInt(str_input);

    while (integer_input < 1) { // If the value is negative
      System.out.println("Enter a number that is greater 0!");
      str_input = input.nextLine();
      integer_input = Integer.parseInt(str_input);
    }

    return integer_input;
  }
  // Outputs a message, and takes an input if yes or no is not entered then the␣
↪user is asked again.
  public static String get_yes_no_input (String message){
    System.out.println (message);
    Scanner input = new Scanner(System.in);
    String yes_no_input = input.next().toLowerCase();

    while  (!(yes_no_input.equals("yes") || yes_no_input.equals("no"))){
      System.out.println("Please answer with 'yes' or 'no': ");
      yes_no_input = input.next().toLowerCase();
    }

    return yes_no_input;
  }
  // Outputs the rules of the game
  public static void pet_intro () {
    System.out.println ("Welcome to the pet program"); // Includes Screen␣
↪output, Keyboard input
    System.out.println ("You have to look after a pet!");
    return;
  }
  // Uses the randome method to generate a number between 1 and 5 and sets the␣
↪hunger level to that value.
  public static void set_hunger_level_main(Pet pet) {
    Random random = new Random();
    int hunger_level = random.nextInt(5) + 1 ;
    set_hunger_level(pet, hunger_level);
    return;
  }
  // Takes array as  the argument, loop throught the array and finds␣
↪description corisponding to the hunger level and ouputs the hunger desription
```

```
public static void get_hunger_state(final String[] HUNGER_DESCRIPTION_ARRAY,␣
↪Pet pet) { // Defines and uses a method that take argument(s) // Defines and␣
↪uses methods including at least one that is passed and uses array arguments
  for (int i = 0; i < get_hunger_level(pet); i++) { // Includes Loops //␣
↪Includes Arrays, accessed with a loop

    if (i == get_hunger_level(pet) - 1) {
      System.out.println("Your pet is " + HUNGER_DESCRIPTION_ARRAY[i]);
    }

  }

}
// If the hunger level is 2 or less then the health level drops by 2, It␣
↪takes an input on how to treat the pet, based on the treatment given by the␣
↪user the health level increases
public static void health_check(Pet pet) {

  if (get_hunger_level(pet) <= 2) {
      int health_level = get_health_level(pet) - 2;
      set_health_level(pet, health_level);
  }

  Scanner input = new Scanner(System.in);
  String pet_care = get_string_input_space("Would you like to feed it, hug␣
↪it, give it medicine?");

  while (!pet_care.equals("Feed it") && !pet_care.equals("Hug it") && !
↪pet_care.equals("Give it medicine")) {
      System.out.println("Invalid input");
      pet_care = get_string_input_space("Would you like to feed it, hug it,␣
↪give it medicine?");
  }

  if (get_health_level(pet) >= 8) {
      set_health_level(pet, 8);
      System.out.println("Max health");
  }
  else if (pet_care.equals("Feed it")) {
      int health_level = get_health_level(pet) + 1;
      set_health_level(pet, health_level);
  }
  else if (pet_care.equals("Give it medicine")) {
      int health_level = get_health_level(pet) + 2;
      set_health_level(pet, health_level);
  }
```

```
    return;
  }
  // Check if a file exists with the pet name entered, if yes then reads the␣
↪value of the file and creates a record where the values are stored.
  public static Pet[] get_pet_records_from_file(Pet[] pet_records_array, String␣
↪pet_name, int j) {
    File pet_file = new File(pet_name + ".txt");

    if (pet_file.exists()) {

        try { // BOTH file input AND file output.
            BufferedReader file_reader = new BufferedReader(new␣
↪FileReader(pet_file));
            String name = file_reader.readLine();
            int health_level = Integer.parseInt(file_reader.readLine());
            int hunger_level = Integer.parseInt(file_reader.readLine());
            boolean survived = Boolean.parseBoolean(file_reader.readLine());
            int num_rounds = Integer.parseInt(file_reader.readLine());
            Pet my_pet = create_pet_record(name, health_level, hunger_level,␣
↪survived, num_rounds);
            pet_records_array[j] = my_pet;
            file_reader.close();
        }
        catch (IOException e) {
            e.printStackTrace();
        }

    }
    else {
        System.out.println("No, your pet does not exist yet.");
        Pet my_pet = create_pet_record(pet_name, 8, 0, true, 3); // Default␣
↪values
        pet_records_array[j] = my_pet;
        System.out.println("Happy 0th Birthday " +␣
↪get_name(pet_records_array[j]));
    }

    return pet_records_array;
  }
  // Write the fields of the record into a new file and the file name is pet␣
↪name.txt
  public static void write_pet_record_to_file(Pet pet) {

    try {
        // Construct the file name based on pet's name
```

```
      String file_name = get_name(pet) + ".txt"; // BOTH file input AND file␣
↪output.
      PrintWriter output_stream = new PrintWriter(new FileWriter(file_name));
      // Write the pet's information to the file
      output_stream.println(get_name(pet));
      output_stream.println(get_health_level(pet));
      output_stream.println(get_hunger_level(pet));
      output_stream.println(get_survived(pet));
      output_stream.println (get_num_rounds(pet));
      output_stream.close(); // Close the PrintWriter when done
  }
  catch (IOException e) {
      System.err.println("Error writing the pet record to a file: " + e.
↪getMessage());
  }

}

public static void main (String [] args){
  final String[] HUNGER_DESCRIPTION_ARRAY = {"ravenous", "famished",␣
↪"starving", "hungry", "full"}; // Final variables are used for literal␣
↪constants
  int num_players = get_positive_integer_input ("Enter the number of players␣
↪that will play this game.");
  Pet[] pet_records_array = new Pet[num_players];
  String returned_player;
  final int MAX_ROUNDS =4;
  pet_intro ();
  // Loop to set up pet records for each player
  for ( int j = 1; j<= num_players; j++){
    System.out.println ("Hi, player " + j);
    returned_player= get_yes_no_input("Are you a returning player? (yes/no):
↪");

    if (returned_player.equals("yes")){ // If they are a returning player
      String pet_name =get_string_input("What is your pet name?");
      pet_records_array = get_pet_records_from_file (pet_records_array,␣
↪pet_name, j-1); // get info from file
    }
    else{
      String pet_name =get_string_input("What would you like to name your pet?
↪");
      Pet my_pet = create_pet_record(pet_name, 8, 0, true, 3); // Default␣
↪values
      pet_records_array[j-1] = my_pet; // stores the record into an array
```

```
      System.out.println("Happy 0th Birthday " +␣
↪get_name(pet_records_array[j-1]));
    }

  }
  // Repeats the code for however many rounds there are
  for ( int x= 0; x<= MAX_ROUNDS; x++){
    // Repeat the code for however many players there are.
    for (int i = 0; i < pet_records_array.length; i++) {
      // Gets the record at position i
      Pet my_pet = pet_records_array[i];
      // If the position i is empty in the pet_records_array
      if (my_pet != null) {

        if (get_num_rounds(my_pet) > 0) {
          System.out.println("It is " + get_name(my_pet) + "'s turn");
          set_hunger_level_main(my_pet);
          System.out.println("On a scale of 1-5, " + get_name(my_pet) + "'s␣
↪hunger rates " + get_hunger_level(my_pet));
          get_hunger_state(HUNGER_DESCRIPTION_ARRAY, my_pet);
          health_check(my_pet);
          System.out.println("Health level: " + get_health_level(my_pet));
          // If the health_level is less than zero meaning the pet is dead.
          if (get_health_level(my_pet) <= 0) {
            System.out.println("Your pet died");
            set_survived(my_pet, false);
            set_num_rounds(my_pet, 0);
          }
          else {
            int n_rounds =get_num_rounds(my_pet) -1;
            set_num_rounds(my_pet, n_rounds);
            // If the game has eneded and the health level is greater than␣
↪zero
            if (get_health_level(my_pet) > 0 && get_num_rounds(my_pet)== 0 ) {
              System.out.println("Your pet survived!");
            }
            else {
              String terminate = get_yes_no_input("Would you like to␣
↪terminate the game? (yes/no)");
              // If they want to terminate the pet must be alive and they␣
↪should have one or more rounds left
              if (terminate.startsWith("y") && get_survived(my_pet) &&␣
↪get_num_rounds(my_pet)>= 1) {
                write_pet_record_to_file(my_pet);
                set_num_rounds(my_pet, 0);
              }
```

```
            }

          }

        }

        pet_records_array[i] = my_pet;
      }

    }

  }

  return;
 }

}
```

[2]:
```
PetProgram.main(null);
```

```
Enter the number of players that will play this game.
Welcome to the pet program
You have to look after a pet!
Hi, player 1
Are you a returning player? (yes/no):
What would you like to name your pet?
Happy 0th Birthday Taher
Hi, player 2
Are you a returning player? (yes/no):
What would you like to name your pet?
Happy 0th Birthday zidan
It is Taher's turn
On a scale of 1-5, Taher's hunger rates 4
Your pet is hungry
Would you like to feed it, hug it, give it medicine?
Max health
Health level: 8
Would you like to terminate the game? (yes/no)
It is zidan's turn
On a scale of 1-5, zidan's hunger rates 2
Your pet is famished
Would you like to feed it, hug it, give it medicine?
Health level: 7
Would you like to terminate the game? (yes/no)
It is zidan's turn
On a scale of 1-5, zidan's hunger rates 1
```

```
Your pet is ravenous
Would you like to feed it, hug it, give it medicine?
Health level: 6
Would you like to terminate the game? (yes/no)
It is zidan's turn
On a scale of 1-5, zidan's hunger rates 5
Your pet is full
Would you like to feed it, hug it, give it medicine?
Health level: 7
Your pet survived!
```

END OF LITERATE DOCUMENT