Endterm Project

Find your lovely pet

Diyar Mukhidenov

Astana IT University

IT-2004

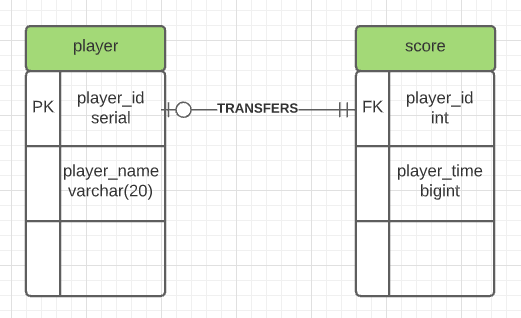
JAVA OOP

Elvira Aitmukhanbetova

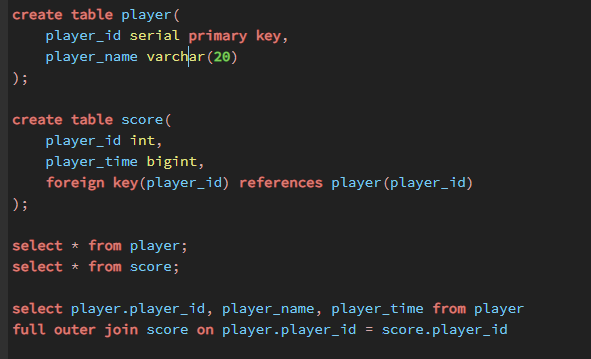
March 9, 2021

This game about user and pet. Your pet is lost, and you must find it. The dog spawns randomly and can go through the map randomly. Your initial position is x = 0 and y = 0. On every position you can call your pet, and if it is near to you will get response with “GAV GAV” which means that your dog is close. There is a score table, and you should find your pet as fast as possible in order to get first place.

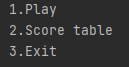
ER Table:



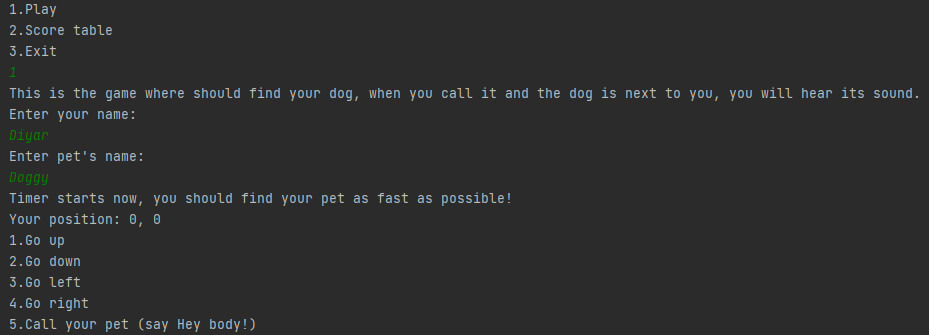
Creating tables in database:



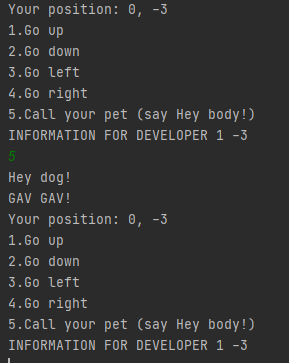
Main menu:



Play part and initializing player:



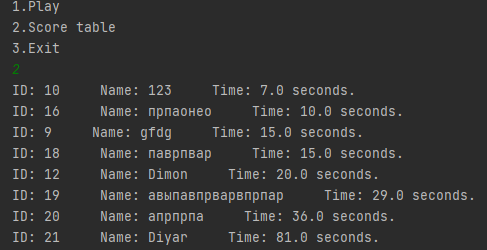
Gameplay part and calling your dog with response:



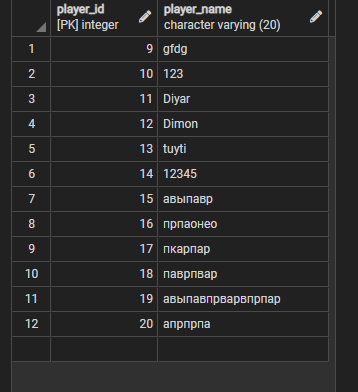
Winning the game with your time:

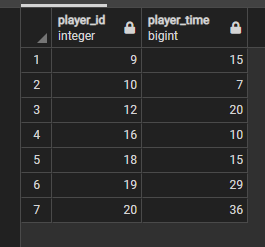


Checking score table:



Inserted data into database after the game:





The code also includes classes such as Point, Entity, Player and Pet.

The source code:

Main:

package com.company; // the package of the project  
  
import java.sql.\*; // importing sql libraries to use database  
import java.util.Scanner; // importing scanner to have an opportunity to input data  
  
public class Main { // class main  
  
 public static void menu(){ //static (without object) method which outputs the main menu of the game  
 System.*out*.println("1.Play"); // 1 to play  
 System.*out*.println("2.Score table"); // 2 to see score table  
 System.*out*.println("3.Exit"); // 3 to exit from the game  
 }  
  
 public static void main(String[] args) { // the program starts here, method main  
  
 Scanner scanner = new Scanner(System.*in*); // creating scanner object  
 boolean isStarted = false; // bool type variable which is needed to restart timer  
 long beginning = 0; // a variable for the timer  
  
 Connection connection = null; // connection for dbms  
 Statement stmt = null; // statement for dbms  
 ResultSet rs = null; // result set  
 ResultSet rss = null;// result set  
  
 try { // exception handling  
 Class.*forName*("org.postgresql.Driver"); /////////////  
 connection = DriverManager.*getConnection*("jdbc:postgresql://localhost:5432/postgres", "postgres", "12345"); ///////////  
 if (connection != null){ ////// this part of code is needed to connect to the postgre database  
 stmt = connection.createStatement(); // statement  
 // System.out.println("connected");  
 }  
 else {  
 System.*out*.println("no connection"); // if there is no connection  
 }  
 }catch (Exception e) { // if there is an exception we catch it and output  
 System.*out*.println(e); ////////  
 }  
  
 while (true) { // infinite loop  
 try{ // exception handling  
 *menu*(); // calling for method menu  
 int choice = scanner.nextInt(); // variable choice which is equal to input data  
 scanner.nextLine(); //////  
  
 if (choice == 1) { // if user wants to play then  
 System.*out*.println("This is the game where should find your dog, when you call it and the dog is next to you, you will hear its sound.");  
 System.*out*.println("Enter your name:"); // background information and inserting user's name and dog name  
 String playerName = scanner.nextLine(); // variable playerName is equal to input data  
 Player player = new Player(playerName); // creating player object with playerName name  
 System.*out*.println("Enter pet's name:"); ////  
 Pet dog = new Pet(scanner.nextLine()); // creating dog object with inserted name  
  
 PreparedStatement prst = connection.prepareStatement("insert into player(player\_name) values(?)"); // creating a prepared statement with inserting data  
 prst.setString(1,playerName); // inserting name of the player into player table  
 prst.executeUpdate(); // execute the prepared statement  
  
 System.*out*.println("Timer starts now, you should find your pet as fast as possible!"); // about timer  
 player.setPosition(0, 0); // initializing player's initial position as x=0 and y=0  
  
 int positive = 1; // int variable positive which is equal to 1  
 if ((int)(Math.*random*() \* 2) == positive) { // this part of code initialize the initial position of your dog by random  
 dog.getPosition().setX((int)(Math.*random*() \* 5)); /////////// random x from 0 to 4  
 }  
 else {  
 dog.getPosition().setX((int)(Math.*random*() \* -5)); ///////////// random x from -4 to 0  
 }  
 if ((int)(Math.*random*() \* 2) == positive) { /////////////  
 dog.getPosition().setY((int)(Math.*random*() \* 5)); ///////////// random y from 0 to 4  
 }  
 else {  
 dog.getPosition().setY((int)(Math.*random*() \* -5)); ////////////////// random y from -4 to 0  
 }  
  
 while(true){ //infinite loop  
 if (!isStarted){ // if isStarted is not false  
 isStarted = true; //then it is true  
 beginning = System.*currentTimeMillis*(); // a variable for timer  
 }  
 System.*out*.println("Your position: " + player.methodX() + ", " + player.methodY()); // output your current position  
 System.*out*.println("1.Go up"); // north way  
 System.*out*.println("2.Go down"); // south way  
 System.*out*.println("3.Go left"); // west way  
 System.*out*.println("4.Go right"); // east way  
 System.*out*.println("5.Call your pet (say Hey body!)"); // optionally calling your dog  
 //System.out.println("INFORMATION FOR DEVELOPER " + dog.getPosition().getX() + " " + dog.getPosition().getY());  
 int move = scanner.nextInt(); //variable move is equal to input data  
 scanner.nextLine(); ///  
 if (move == 1 && player.getPosition().getY() != 4){ //////////// this part of code is needed to make movements for a player  
 player.getPosition().setY(player.getPosition().getY() + 1); //going top  
 }  
 else if (move == 2 && player.getPosition().getY() != -4){ // going down  
 player.getPosition().setY(player.getPosition().getY() - 1); //////  
 }  
 else if (move == 3 && player.getPosition().getX() != -4){ // goind left  
 player.getPosition().setX(player.getPosition().getX() - 1); ///////  
 }  
 else if (move == 4 && player.getPosition().getX() != 4){ // going right  
 player.getPosition().setX(player.getPosition().getX() + 1); /////////  
 }  
 else if (move == 5){ //calling your dog  
 player.sound(); // saying HEY DOG!  
 if ((player.getPosition().getY() == dog.getPosition().getY() && (player.getPosition().getX() - dog.getPosition().getX() == 1)) || (player.getPosition().getY() == dog.getPosition().getY() && (dog.getPosition().getX() - player.getPosition().getX() == 1)) || (player.getPosition().getX() == dog.getPosition().getX() && (player.getPosition().getY() - dog.getPosition().getY() == 1)) || (player.getPosition().getX() == dog.getPosition().getX() && (dog.getPosition().getY() - player.getPosition().getY() == 1))){  
 dog.sound(); // if your dog is next to you, your dog will say gav gav  
 }  
 continue; // continue the loop  
 }  
 int player\_id = 0; // this part of code is end when you find your pet  
 rss = stmt.executeQuery("select player\_id from player where player.player\_id = player\_id"); //executing query  
 while (rss.next()){  
 player\_id = rss.getInt("player\_id"); // getting player's current id  
 }  
 if (player.getPosition().getX() == dog.getPosition().getX() && player.getPosition().getY() == dog.getPosition().getY()){ // if player and dog positions are the same  
 System.*out*.println("Your position: " + player.methodX() + ", " + player.methodY()); // output your position  
 System.*out*.println("You found your pet!"); // finding your pet  
 long time = (System.*currentTimeMillis*() - beginning)/1000; // score time  
 System.*out*.println("Your time is: " + time + " seconds"); // showing your time  
 PreparedStatement prst1 = connection.prepareStatement("insert into score(player\_id, player\_time) values(?,?)"); ///  
 prst1.setInt(1,player\_id); ////  
 prst1.setLong(2,time); //and inserting this score into table score  
 prst1.executeUpdate(); ////  
 isStarted = false; ///  
 break; // stop the loop  
 }  
 int randomMove = (int)(Math.*random*() \* 4); ///////// this part of code is needed to make random move for your dog  
 if (randomMove == 0 && dog.getPosition().getY() != 4){ // it depends on random move variable which is equal to from 0 to 3  
 dog.getPosition().setY(dog.getPosition().getY() + 1); // in this case going top  
 }  
 else if (randomMove == 1 && dog.getPosition().getY() != -4){ // in this case going down  
 dog.getPosition().setY(dog.getPosition().getY() - 1); /////  
 }  
 else if (randomMove == 2 && dog.getPosition().getX() != -4){ // in this case going left  
 dog.getPosition().setX(dog.getPosition().getX() - 1); ///////  
 }  
 else if (randomMove == 3 && dog.getPosition().getX() != 4){ // in this case going right  
 dog.getPosition().setX(dog.getPosition().getX() + 1); ///////  
 }  
 player\_id = 0; // this part of code is the end when your pet find you  
 rss = stmt.executeQuery("select player\_id from player where player.player\_id = player\_id"); //executing query  
 while (rss.next()){  
 player\_id = rss.getInt("player\_id"); // getting current player's id  
 }  
 if (player.getPosition().getX() == dog.getPosition().getX() && player.getPosition().getY() == dog.getPosition().getY()){ // if player and dog positions are the same  
 System.*out*.println("Your position: " + player.methodX() + ", " + player.methodY()); // output your position  
 System.*out*.println("You found your pet!"); // finding your pet  
 long time = (System.*currentTimeMillis*() - beginning)/1000; // score time  
 System.*out*.println("Your time is: " + time + " seconds"); // showing your time  
 PreparedStatement prst1 = connection.prepareStatement("insert into score(player\_id, player\_time) values(?,?)"); ////  
 prst1.setInt(1,player\_id); ////  
 prst1.setLong(2,time); //and inserting this score into table score  
 prst1.executeUpdate(); ////  
 isStarted = false; ///  
 break; // stop the loop  
 }  
 }  
 }  
 else if (choice == 2){ // condition, if choice is equal to 2  
 rs = stmt.executeQuery("select player.player\_id, player\_name, player\_time from player\n" +  
 "full outer join score on player.player\_id = score.player\_id order by player\_time asc"); //executing sql query  
 while (rs.next()){  
 System.*out*.println("ID: " + rs.getInt("player\_id") + " Name: " + rs.getString("player\_name") + " Time: " + rs.getFloat("player\_time") + " seconds."); // output these statements  
 }  
 }  
 else if (choice == 3){ // otherwise when it is 3  
 System.*exit*(0); // then exit the program  
 }  
 }catch (Exception e){ // if there is an exception then catch and output it  
 System.*out*.println(e); /////  
 }  
 }  
 }  
}

Point class:

package com.company; // the package of the project  
  
public class Point { // class Point which is needed to create movement to the game  
 private int x; // field x  
 private int y; // field y  
  
 public Point(){ // the constructor with no parameters which declare initial position for the object  
 x = 0; // this x is equal to zero  
 y = 0; // this y is equal to zero  
 }  
  
 public Point(int x, int y){ // the constructor with parameters x and y  
 this.x = x; // this x will be equal to argument x  
 this.y = y; // this y will be equal to argument y  
 }  
 // encapsulation part  
 public int getX() { // getter for x  
 return x;  
 }  
  
 public void setX(int x) { // setter for x  
 this.x = x;  
 }  
  
 public int getY() { // getter for y  
 return y;  
 }  
  
 public void setY(int y) { // setter for y  
 this.y = y;  
 }  
}

Entity class:

package com.company; // the package of the project  
  
public class Entity { // parent class Entity which is needed to create sublasses such as player and pet  
 private String name; // private field name (String type)  
 private Point position = new Point(); // field Point which creates an object of Point class in order to get access for subclasses to x and y coordinates  
  
 public Entity(){ // the constructor with no parameters  
 name = “entity”; // by default name is equal to “entity” value  
 }  
  
 public Entity(String name) { // the constructor with parameter String name  
 setName(name); // using setter to set name with argument name  
 }  
  
 public String getName() { //getter for name  
 return name;  
 }  
  
 public void setName(String name) { // setter for name  
 this.name = name;  
 }  
  
 public void sound(){ // method sound which will be override in subclasses  
 System.*out*.println(“some sound”); // output “some sound” by default  
 }  
  
 public Point getPosition() { //getter for position  
 return position;  
 }  
  
 public void setPosition(Point position) { //setter for position  
 this.position = position;  
 }  
 public void setPosition(int x, int y) { //setter for position with x and y arguments  
 this.position.setX(x);  
 this.position.setY(y);  
 }  
 public String methodX(){ // method which helps to get access for x position of the entity  
 return “” + position.getX();  
 }  
 public String methodY(){ // method which helps to get access for y position of the entity  
 return “” + position.getY();  
 }  
}

Player class:

package com.company; // the package of the project  
// inheritance starts here  
public class Player extends Entity { // class Player which is subclass of Entity  
 public Player(String name){ // constructor with the name which is declaired in parent class  
 super(name);  
 }  
 // polymorphism starts here  
 @Override //overriding parent's method sound  
 public void sound(){  
 System.*out*.println("Hey dog!"); // calling for your dog  
 }  
}

Pet class:

package com.company; // the package of the project  
  
public class Pet extends Entity { // class Pet which is subclass of Entity  
 public Pet(String name){ // constructor with the name which is declaired in parent class  
 super(name);  
 }  
  
 @Override //overriding parent's method sound  
 public void sound(){  
 System.*out*.println("GAV GAV!"); // it is needed when you call your dog and if it is next to you  
 }  
}