Кафедра вычислительной техники



**ЛАБОРАТОРНАЯ РАБОТА №6**

**по дисциплине:** *технология программирования*

**на тему:** *Потоки ввода-вывода.*

*Сериализация объектов в файл.*

|  |  |
| --- | --- |
| Выполнил: | Проверил: |
| Студент гр. *АВТ-808*, *АВТФ* | *ассистент каф. ВТ* |
| *Горбанев В.В.* | *Михайленко Дмитрий Анатольевич* |
| «\_\_\_» \_\_\_\_\_\_ 20\_\_г. | «\_\_\_» \_\_\_\_\_\_ 20\_\_г. |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| (подпись) | (подпись) |

# **ЦЕЛЬ И ЗАДАНИЕ**

1. Изучить особенности реализации сетевых приложений в Java.
2. Доработать программу, созданную в лабораторной работе № 5:

**Вариант 11**

Реализовать в консоли команды «Остановить генерацию ХХХ объектов» и «Продолжить генерацию ХХХ объектов». Чекбокс в интерфейсе должен вести себя так же, как если бы использовали его, а не исполняли команды (галочка появляется/исчезает).

# **Задание**

1. создать отдельное консольное приложение, которое будет играть роль TCP-сервера. Клиентом будет приложение, которое было создано в предыдущих работах;
2. TCP-сервер должен ожидать подключения клиентов и выдавать вновь подключенному клиенту список уже подключенных. Необходимо также внести изменения в интерфейс клиента, так чтобы в панели управления отображался список всех подключенных к серверу клиентов. При отсоединении клиентов или при подключении новых список должен обновляться;
3. запрограммировать специальное взаимодействие по TCP с другими клиентами через сервер по варианту;

# **Ход работы**

В ходе работы были создан сервер, а также класс клиен для взаимодействия с ним:

public class MyServer {  
  
 public static void main(String[] args) throws IOException {  
 try(ServerSocket serverSocket = new ServerSocket(8080)){  
 JFrame serverFrame = new JFrame("server");  
 serverFrame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 serverFrame.setSize(300,200);  
 serverFrame.setLocationRelativeTo(null);  
 serverFrame.setResizable(false);  
 serverFrame.setVisible(true);  
  
 while (true){  
 Socket socket = serverSocket.accept();  
 new Thread(new Server(socket)).start();  
 }  
 }  
 }  
  
}

В конструкторе клиента идет добавления пользователей на сервер:

public Client() {  
 *user* = JOptionPane.*showInputDialog*("Log in:");  
 try {  
 *socket* = new Socket("localhost", 8080);  
 ObjectOutputStream objectOutputStream = new ObjectOutputStream(*socket*.getOutputStream());  
 objectOutputStream.writeUTF(*user*);  
 objectOutputStream.writeUTF(*SET\_USER*);  
 objectOutputStream.writeObject(null);  
 objectOutputStream.flush();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
}

Также в клиенте определены методы:

public static int CountUsers()

public static void Update()

public static void Download()

public static void Swap()

public static String YouNow()

public static String NameUsers(int i)

public static void Disconnect()

CountUsers() - делает запрос на сервер и получает сколько пользователей онлайн.

Update() - загружает список существ каждого пользователя.

Download() - загружает выбранного пользователя.

Swap() – Отправляет всех котов к выбранному пользователю, и забирает всех собак у негою.

YouNow() – показывает кто сейчас вы.

NameUsers(int i) – показывает имя пользователя под этим индексом.

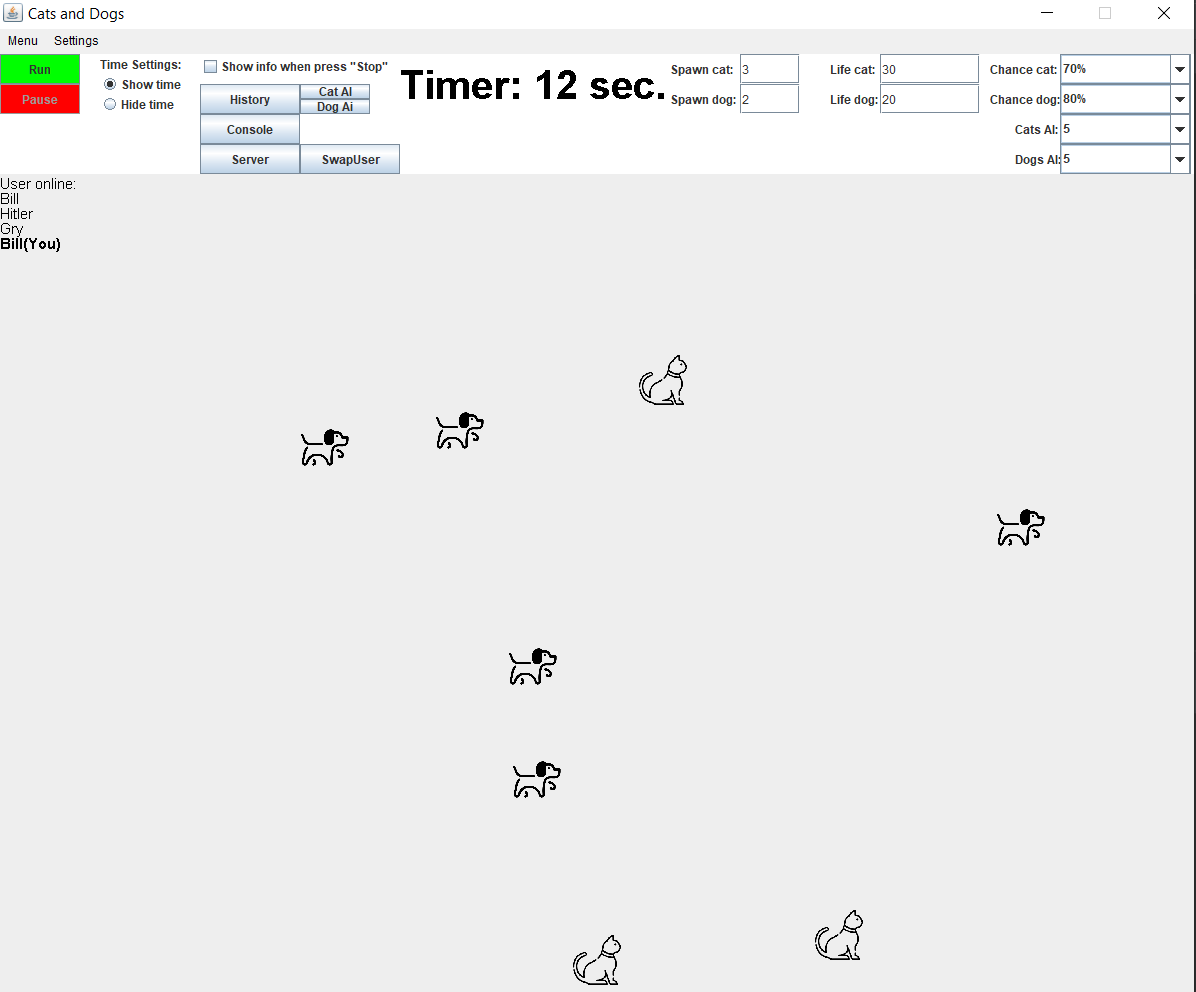
Disconnect() – отключает пользователя с сервера по клавише “E”

Сервер содержит следующие поля:

private Socket socket;  
 private static int *countUsers* = 0;  
 private static ArrayList<String> *users* = new ArrayList<>();  
 private static HashMap<String,ArrayList<Pet>> *pets* = new HashMap<>();

users – именя всех пользователей.

pets – колекция питомцев данного пользователя.



*Рис. 1. Результат выполнения метода кнопки «start».*

# Вывод:

В ходе лабораторной работы были изучить особенности реализации системы ввода-вывода в Java.

# Листинг:

GUI

package com.company;  
  
  
import com.company.BaseAI.AI.CatAI;  
import com.company.BaseAI.AI.DogAI;  
import com.company.Habitat.Habitat;  
import com.company.Habitat.Pet.Creatures.Cat;  
import com.company.Habitat.Pet.Creatures.Dog;  
import com.company.Single.Singleton;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.\*;  
import java.util.Timer;  
import java.util.TimerTask;  
  
import static java.awt.event.KeyEvent.\*;  
  
public class GUI extends JPanel {  
 public MyTimerTask timerTask = new MyTimerTask();  
  
 private int myTimer = 0;  
 Habitat window;  
 boolean endApp = false;  
 boolean timeVisible = true;  
 boolean lineVisible = true;  
 boolean useKeys = true;  
 boolean useDialog = true;  
  
 boolean dialogOn = false;  
 boolean isPressedOn = false;  
 boolean isPressedOnB = false;  
  
 boolean beginning = true;  
  
 JFrame jFrame;  
 CatAI catAI = new CatAI();  
 DogAI dogAI = new DogAI();  
  
 public JButton buttonCatAI = new JButton("Cat AI");  
 ActionListener actionListenerForCatAI = new catAIActionListener();  
 public JButton buttonDogAI = new JButton("Dog Ai");  
 ActionListener actionListenerForDogAI = new dogAIActionListener();  
 public JButton run = new JButton("Run");  
 public JButton pause = new JButton("Pause");  
 public JButton history = new JButton("History");  
 ActionListener actionListenerForHistory = new historyActionListener();  
 public JRadioButton showTime = new JRadioButton("Show time",true);  
 public JRadioButton hideTime = new JRadioButton("Hide time",false);  
 public JLabel selectTime = new JLabel("Time Settings:");  
 public JLabel selectCatsTimeSpawn = new JLabel("Spawn cat:");  
 public JLabel selectDogsTimeSpawn = new JLabel("Spawn dog:");  
 public JLabel selectCatsSpawn = new JLabel("Chance cat:");  
 public JLabel selectDogsSpawn = new JLabel("Chance dog:");  
 public JLabel selectCatsAI = new JLabel("Cats AI:");  
 public JLabel selectDogsAI = new JLabel("Dogs AI:");  
 ButtonGroup group = new ButtonGroup();  
  
 public JCheckBox screenResolution = new JCheckBox("Show info when press \"Stop\" ");  
 public JTextField spawnCats = new JTextField();  
 ActionListener actionListenerForCats = new spawnCatsActionListener();  
 public JTextField spawnDogs = new JTextField();  
 ActionListener actionListenerForDogs = new spawnDogsActionListener();  
  
 public JLabel selectLifeOfCats = new JLabel("Life cat:");  
 public JLabel selectLifeOfDogs = new JLabel("Life dog:");  
 public JTextField lifeCats = new JTextField();  
 ActionListener actionListenerLifeCats = new lifeCatsActionListener();  
 public JTextField lifeDogs = new JTextField();  
 ActionListener actionListenerLifeDogs = new lifeDogsActionListener();  
  
  
 public JComboBox catsComboBox = new JComboBox();  
 ActionListener catsComboBoxActionListener = new catsComboBoxActionListener();  
  
 public JComboBox dogsComboBox = new JComboBox();  
 ActionListener dogsComboBoxActionListener = new dogsComboBoxActionListener();  
  
 public JComboBox priorityCatAI = new JComboBox();  
 ActionListener priorityAIActionListenerForCats = new priorityCatAIActionListener();  
  
 public JComboBox priorityDogAI = new JComboBox();  
 ActionListener priorityAIActionListenerForDogs = new priorityDogAIActionListener();  
  
 public GUI(Habitat window, JFrame jFrame){  
 this.window = window;  
 this.jFrame = jFrame;  
  
  
 lifeCats.addActionListener(actionListenerLifeCats);  
 lifeDogs.addActionListener(actionListenerLifeDogs);  
 lifeCats.setText(String.*valueOf*(window.getCatsTimeOfLife()));  
 lifeDogs.setText(String.*valueOf*(window.getDogsTimeOfLife()));  
 spawnCats.setToolTipText("Select time spawn for cat");  
 spawnDogs.setToolTipText("Select time spawn for dog");  
 catsComboBox.setToolTipText("Select chance spawn for cat");  
 dogsComboBox.setToolTipText("Select chance spawn for dog");  
 priorityCatAI.setToolTipText("Select cats priority AI");  
 priorityDogAI.setToolTipText("Select cats priority AI");  
  
 catsComboBox.addItem("0%");  
 catsComboBox.addItem("10%");  
 catsComboBox.addItem("20%");  
 catsComboBox.addItem("30%");  
 catsComboBox.addItem("40%");  
 catsComboBox.addItem("50%");  
 catsComboBox.addItem("60%");  
 catsComboBox.addItem("70%");  
 catsComboBox.addItem("80%");  
 catsComboBox.addItem("90%");  
 catsComboBox.addItem("100%");  
 catsComboBox.setSelectedIndex(7);  
 catsComboBox.setBackground(Color.*white*);  
 catsComboBox.setPreferredSize(new Dimension(130,30));  
  
 dogsComboBox.addItem("0%");  
 dogsComboBox.addItem("10%");  
 dogsComboBox.addItem("20%");  
 dogsComboBox.addItem("30%");  
 dogsComboBox.addItem("40%");  
 dogsComboBox.addItem("50%");  
 dogsComboBox.addItem("60%");  
 dogsComboBox.addItem("70%");  
 dogsComboBox.addItem("80%");  
 dogsComboBox.addItem("90%");  
 dogsComboBox.addItem("100%");  
 dogsComboBox.setSelectedIndex(4);  
 dogsComboBox.setBackground(Color.*white*);  
 dogsComboBox.setPreferredSize(new Dimension(130,30));  
  
 priorityCatAI.addItem("1");  
 priorityCatAI.addItem("2");  
 priorityCatAI.addItem("3");  
 priorityCatAI.addItem("4");  
 priorityCatAI.addItem("5");  
 priorityCatAI.addItem("6");  
 priorityCatAI.addItem("7");  
 priorityCatAI.addItem("8");  
 priorityCatAI.addItem("9");  
 priorityCatAI.addItem("10");  
 priorityCatAI.setSelectedIndex(4);  
 priorityCatAI.setBackground(Color.*white*);  
 priorityCatAI.setPreferredSize(new Dimension(130,30));  
  
 priorityDogAI.addItem("1");  
 priorityDogAI.addItem("2");  
 priorityDogAI.addItem("3");  
 priorityDogAI.addItem("4");  
 priorityDogAI.addItem("5");  
 priorityDogAI.addItem("6");  
 priorityDogAI.addItem("7");  
 priorityDogAI.addItem("8");  
 priorityDogAI.addItem("9");  
 priorityDogAI.addItem("10");  
 priorityDogAI.setSelectedIndex(4);  
 priorityDogAI.setBackground(Color.*white*);  
 priorityDogAI.setPreferredSize(new Dimension(130,30));  
  
 spawnCats.setText(String.*valueOf*(window.getN1()));  
 spawnCats.addActionListener(actionListenerForCats);  
 spawnDogs.setText(String.*valueOf*(window.getN2()));  
 spawnDogs.addActionListener(actionListenerForDogs);  
 catsComboBox.addActionListener(catsComboBoxActionListener);  
 dogsComboBox.addActionListener(dogsComboBoxActionListener);  
 history.addActionListener(actionListenerForHistory);  
 buttonCatAI.addActionListener(actionListenerForCatAI);  
 buttonDogAI.addActionListener(actionListenerForDogAI);  
 priorityCatAI.addActionListener(priorityAIActionListenerForCats);  
 priorityDogAI.addActionListener(priorityAIActionListenerForDogs);  
  
 add(buttonCatAI);  
 add(buttonDogAI);  
 add(catsComboBox);  
 add(dogsComboBox);  
 add(priorityCatAI);  
 add(priorityDogAI);  
 group.add(showTime);  
 group.add(hideTime);  
 showTime.setBackground(Color.*white*);  
 hideTime.setBackground(Color.*white*);  
 add(screenResolution);  
 add(showTime);  
 add(hideTime);  
 add(selectTime);  
 add(spawnCats);  
 add(spawnDogs);  
 add(selectCatsTimeSpawn);  
 add(selectCatsSpawn);  
 add(selectDogsTimeSpawn);  
 add(selectDogsSpawn);  
 add(selectCatsAI);  
 add(selectDogsAI);  
 add(history);  
 add(lifeCats);  
 add(lifeDogs);  
 add(selectLifeOfCats);  
 add(selectLifeOfDogs);  
  
  
 add(run);  
 add(pause);  
 run.setEnabled(false);  
 pause.setEnabled(false);  
 run.setBackground(Color.*RED*);  
 pause.setBackground(Color.*RED*);  
 screenResolution.setBackground(Color.*white*);  
  
 }  
  
  
 public void paintComponent(Graphics g){  
 super.paintComponent(g);  
  
 if(beginning){  
 selectLifeOfCats.setVisible(false);  
 selectLifeOfDogs.setVisible(false);  
 buttonCatAI.setVisible(false);  
 buttonDogAI.setVisible(false);  
 lifeCats.setVisible(false);  
 lifeDogs.setVisible(false);  
 history.setVisible(false);  
 selectCatsTimeSpawn.setVisible(false);  
 selectDogsTimeSpawn.setVisible(false);  
 selectCatsSpawn.setVisible(false);  
 selectDogsSpawn.setVisible(false);  
 selectCatsAI.setVisible(false);  
 selectDogsAI.setVisible(false);  
 catsComboBox.setVisible(false);  
 dogsComboBox.setVisible(false);  
 priorityCatAI.setVisible(false);  
 priorityDogAI.setVisible(false);  
 spawnCats.setVisible(false);  
 spawnDogs.setVisible(false);  
 selectTime.setVisible(false);  
 showTime.setVisible(false);  
 hideTime.setVisible(false);  
 run.setVisible(false);  
 pause.setVisible(false);  
 screenResolution.setVisible(false);  
 lineVisible = false;  
 timeVisible = false;  
 g.setColor(Color.*BLACK*);  
 Font myFont = new Font("Times Roman", Font.*BOLD*, 100);  
 g.setFont(myFont);  
 g.drawString("Press \"B\" to start", 150,window.getSizeY()/2);  
 }  
  
 history.setBounds(200,30,100,30);  
 selectCatsTimeSpawn.setBounds(670,0,200,30);  
 selectDogsTimeSpawn.setBounds(670,30,200,30);  
 selectCatsSpawn.setBounds(990,0,130,30);  
 selectDogsSpawn.setBounds(990,30,140,30);  
 selectCatsAI.setBounds(1015,60,50,30);  
 selectDogsAI.setBounds(1015,90,50,30);  
 lifeCats.setBounds(880,0,100,30);  
 lifeDogs.setBounds(880,30,100,30);  
 selectLifeOfCats.setBounds(830,0,100,30);  
 selectLifeOfDogs.setBounds(830,30,100,30);  
 catsComboBox.setBounds(1060,0,150,30);  
 dogsComboBox.setBounds(1060,30,150,30);  
 priorityCatAI.setBounds(1060,60,150,30);  
 priorityDogAI.setBounds(1060,90,150,30);  
  
 spawnCats.setBounds(740,0,60,30);  
 spawnDogs.setBounds(740,30,60,30);  
  
 selectTime.setBounds(100,0,90,20);  
 showTime.setBounds(100,20,90,20);  
 hideTime.setBounds(100,40,90,20);  
 run.setBounds(0,0,80,30);  
 pause.setBounds(0,30,80,30);  
 screenResolution.setLocation(200,0);  
 buttonCatAI.setBounds(300,30,70,15);  
 buttonDogAI.setBounds(300,45,70,15);  
  
 if(lineVisible) {  
 g.setColor(Color.*white*);  
 g.fillRect(0, 0, 10000, 120);  
 }  
 if(timeVisible) {  
 g.setColor(Color.*BLACK*);  
 Font myFont = new Font("Times Roman", Font.*BOLD*, 40);  
 g.setFont(myFont);  
 g.drawString("Timer: " + myTimer + " sec.", 400, 45);  
 }  
 if(endApp){  
 JFrame endFrame = new JFrame("Результаты");  
 endFrame.setAlwaysOnTop(true);  
 endFrame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 endFrame.setSize(400,400);  
 endFrame.setLocationRelativeTo(null);  
 endFrame.setResizable(false);  
  
 JPanel endPanel = new JPanel(){  
 public void paint(Graphics g){  
 super.paint(g);  
  
 g.setColor(Color.*BLACK*);  
 Font myFont=new Font("Arial",Font.*BOLD*,20);  
 g.setFont(myFont);  
 g.drawString("Время работы: "+myTimer ,100,100);  
 g.setColor(Color.*GREEN*);  
 myFont=new Font("Times new Roman",Font.*ITALIC*,20);  
 g.setFont(myFont);  
 g.drawString("Количество кошек: "+window.cats ,100,140);  
 g.setColor(Color.*ORANGE*);  
 myFont=new Font("Calibri",Font.*PLAIN*,20);  
 g.setFont(myFont);  
 g.drawString("Количество собак: "+window.dogs ,100,180);  
 g.setColor(Color.*RED*);  
 myFont=new Font("Georgia",Font.*BOLD* + Font.*ITALIC*,20);  
 g.setFont(myFont);  
 int quantityPet = window.dogs + window.cats;  
 g.drawString("Всего питомцев: "+quantityPet ,100,220);  
 }  
 };  
 endFrame.add(endPanel);  
  
  
  
 endFrame.setVisible(true);  
 endApp = false;  
 }  
  
 for (int i = 0; i < window.quantityPet; i++) {  
 if (Dog.class.isAssignableFrom(Singleton.*getInstance*().Get(i).getClass())) {  
 Image imageDog = new ImageIcon("img/dog.png").getImage();  
 g.drawImage(imageDog, Singleton.*getInstance*().Get(i).getX(), Singleton.*getInstance*().Get(i).getY(), 50, 50, null);  
 } else {  
 Image imageCat = new ImageIcon("img/cat.png").getImage();  
 g.drawImage(imageCat, Singleton.*getInstance*().Get(i).getX(), Singleton.*getInstance*().Get(i).getY(), 50, 50, null);  
 }  
  
 }  
  
  
  
 addKeyListener(new KeyAdapter() {  
 @Override  
 public void keyTyped(KeyEvent e) { }  
  
  
 @Override  
 public void keyPressed(KeyEvent e) {  
 switch (e.getKeyCode()){  
 case *VK\_B*:  
 if(!useKeys)break;  
 if(!isPressedOnB) {  
 buttonCatAI.setVisible(true);  
 buttonDogAI.setVisible(true);  
 selectLifeOfCats.setVisible(true);  
 selectLifeOfDogs.setVisible(true);  
 lifeCats.setVisible(true);  
 lifeDogs.setVisible(true);  
 history.setVisible(true);  
 spawnCats.setVisible(true);  
 spawnCats.setEnabled(false);  
 spawnDogs.setVisible(true);  
 spawnDogs.setEnabled(false);  
 run.setBackground(Color.*RED*);  
 pause.setBackground(Color.*GREEN*);  
 timerTask.StartWork();  
 lifeCats.setEnabled(false);  
 lifeDogs.setEnabled(false);  
 run.setEnabled(false);  
 pause.setEnabled(true);  
 isPressedOnB = true;  
 selectCatsTimeSpawn.setVisible(true);  
 selectDogsTimeSpawn.setVisible(true);  
 selectCatsSpawn.setVisible(true);  
 selectDogsSpawn.setVisible(true);  
 selectCatsAI.setVisible(true);  
 selectDogsAI.setVisible(true);  
 catsComboBox.setVisible(true);  
 catsComboBox.setEnabled(false);  
 dogsComboBox.setVisible(true);  
 dogsComboBox.setEnabled(false);  
 priorityCatAI.setVisible(true);  
 priorityCatAI.setEnabled(false);  
 priorityDogAI.setVisible(true);  
 priorityDogAI.setEnabled(false);  
 selectTime.setVisible(true);  
 showTime.setVisible(true);  
 hideTime.setVisible(true);  
 run.setVisible(true);  
 pause.setVisible(true);  
 screenResolution.setVisible(true);  
 lineVisible = true;  
 timeVisible = true;  
 beginning = false;  
  
 repaint();  
 }  
 break;  
 case *VK\_T*:  
 if(!useKeys)break;  
 if(!isPressedOn) {  
 timeVisible = !timeVisible;  
 isPressedOn = true;  
  
 if(timeVisible){  
 showTime.setSelected(true);  
 hideTime.setSelected(false);  
 }else{  
 showTime.setSelected(false);  
 hideTime.setSelected(true);  
 }  
 repaint();  
 }  
 break;  
 case *VK\_E*:  
 timerTask.StopWork();  
 if(!useKeys)break;  
 useKeys = false;  
  
  
 timeVisible = false;  
 endApp = true;  
 lineVisible = false;  
 selectLifeOfCats.setVisible(false);  
 selectLifeOfDogs.setVisible(false);  
 lifeCats.setVisible(false);  
 lifeDogs.setVisible(false);  
 history.setVisible(false);  
 selectTime.setVisible(false);  
 showTime.setVisible(false);  
 hideTime.setVisible(false);  
 run.setVisible(false);  
 pause.setVisible(false);  
 screenResolution.setVisible(false);  
 selectCatsTimeSpawn.setVisible(false);  
 selectDogsTimeSpawn.setVisible(false);  
 selectCatsSpawn.setVisible(false);  
 selectDogsSpawn.setVisible(false);  
 selectCatsAI.setVisible(false);  
 selectDogsAI.setVisible(false);  
 spawnCats.setVisible(false);  
 spawnDogs.setVisible(false);  
 catsComboBox.setVisible(false);  
 dogsComboBox.setVisible(false);  
 priorityCatAI.setVisible(false);  
 priorityDogAI.setVisible(false);  
 buttonCatAI.setVisible(false);  
 buttonDogAI.setVisible(false);  
 window.allClear();  
  
 repaint();  
 break;  
 }  
  
 }  
  
 @Override  
 public void keyReleased(KeyEvent e) {  
 if (e.getKeyCode() == *VK\_T*) {  
 if (!useKeys) return;  
 isPressedOn = false;  
  
 }  
  
  
  
 }  
 });requestFocusInWindow();  
  
 if(dialogOn){  
 infDialog info = new infDialog(jFrame,window,myTimer);  
 dialogOn = false;  
 timerTask.StartWork();  
 run.setEnabled(false);  
 pause.setEnabled(true);  
 run.setBackground(Color.*RED*);  
 pause.setBackground(Color.*GREEN*);  
 repaint();  
 }  
  
 showTime.addItemListener(e -> {  
 timeVisible = true;  
 repaint();  
 });  
 hideTime.addItemListener(e -> {  
 timeVisible = false;  
 repaint();  
 });  
  
  
  
 screenResolution.addActionListener(e -> {  
 if(!isPressedOn) {  
 useDialog = !useDialog;  
 repaint();  
 isPressedOn = true;  
 }  
 });  
 isPressedOn = false;  
  
  
 run.addActionListener(e -> {  
 if(!isPressedOn) {  
 if (catAI.isFlag()) {  
 catAI.startAI();  
 }  
 if (dogAI.isFlag()) {  
 dogAI.startAI();  
 }  
 timerTask.StartWork();  
 run.setEnabled(false);  
 pause.setEnabled(true);  
 run.setBackground(Color.*RED*);  
 pause.setBackground(Color.*GREEN*);  
 spawnCats.setEnabled(false);  
 spawnDogs.setEnabled(false);  
 isPressedOn = true;  
 catsComboBox.setEnabled(false);  
 dogsComboBox.setEnabled(false);  
 priorityCatAI.setEnabled(false);  
 priorityDogAI.setEnabled(false);  
 lifeCats.setEnabled(false);  
 lifeDogs.setEnabled(false);  
 repaint();  
 }  
 });isPressedOn = false;  
 pause.addActionListener(e -> {  
 if (!catAI.isFlag()) {  
 catAI.stopAI();  
 }  
 if (!dogAI.isFlag()) {  
 dogAI.stopAI();  
 }  
 timerTask.WaitWork();  
 run.setEnabled(true);  
 pause.setEnabled(false);  
 run.setBackground(Color.*GREEN*);  
 pause.setBackground(Color.*RED*);  
 catsComboBox.setEnabled(true);  
 dogsComboBox.setEnabled(true);  
 priorityCatAI.setEnabled(true);  
 priorityDogAI.setEnabled(true);  
 spawnCats.setEnabled(true);  
 spawnDogs.setEnabled(true);  
 lifeCats.setEnabled(true);  
 lifeDogs.setEnabled(true);  
 if (!useDialog)  
 dialogOn = true;  
 isPressedOnB = false;  
 repaint();  
 requestFocusInWindow();  
  
 });  
  
  
  
  
 }  
 private class catsComboBoxActionListener implements ActionListener {  
 int num;  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 num = catsComboBox.getSelectedIndex();  
 switch (num) {  
 case 0:  
 window.setP1(0);  
 break;  
 case 1:  
 window.setP1(10);  
 break;  
 case 2:  
 window.setP1(20);  
 break;  
 case 3:  
 window.setP1(30);  
 break;  
 case 4:  
 window.setP1(40);  
 break;  
 case 5:  
 window.setP1(50);  
 break;  
 case 6:  
 window.setP1(60);  
 break;  
 case 7:  
 window.setP1(70);  
 break;  
 case 8:  
 window.setP1(80);  
 break;  
 case 9:  
 window.setP1(90);  
 break;  
 case 10:  
 window.setP1(100);  
 break;  
  
 }  
 requestFocusInWindow();  
 }  
 }  
 private class dogsComboBoxActionListener implements ActionListener {  
 int num;  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 num = dogsComboBox.getSelectedIndex();  
 switch (num) {  
 case 0:  
 window.setP2(0);  
 break;  
 case 1:  
 window.setP2(10);  
 break;  
 case 2:  
 window.setP2(20);  
 break;  
 case 3:  
 window.setP2(30);  
 break;  
 case 4:  
 window.setP2(40);  
 break;  
 case 5:  
 window.setP2(50);  
 break;  
 case 6:  
 window.setP2(60);  
 break;  
 case 7:  
 window.setP2(70);  
 break;  
 case 8:  
 window.setP2(80);  
 break;  
 case 9:  
 window.setP2(90);  
 break;  
 case 10:  
 window.setP2(100);  
 break;  
  
 }  
 requestFocusInWindow();  
 }  
 }  
 private class spawnCatsActionListener implements ActionListener {  
 String number;  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 number = spawnCats.getText();  
 try{  
 window.setN1(Integer.*parseInt*(number));  
 }catch (Throwable number){  
 JOptionPane.*showMessageDialog*(null, "Ввод некорректного значения!!!","Error",JOptionPane.*ERROR\_MESSAGE*);  
 }  
 }  
 }  
 private class lifeCatsActionListener implements ActionListener {  
 String number;  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 number = lifeCats.getText();  
 try{  
 window.setCatsTimeOfLife(Integer.*parseInt*(number));  
 }catch (Throwable number){  
 JOptionPane.*showMessageDialog*(null, "Ввод некорректного значения!!!","Error",JOptionPane.*ERROR\_MESSAGE*);  
 }  
 }  
 }  
 private class lifeDogsActionListener implements ActionListener {  
 String number;  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 number = lifeDogs.getText();  
 try{  
 window.setDogsTimeOfLife(Integer.*parseInt*(number));  
 }catch (Throwable number){  
 JOptionPane.*showMessageDialog*(null, "Ввод некорректного значения!!!","Error",JOptionPane.*ERROR\_MESSAGE*);  
 }  
 }  
 }  
 private class spawnDogsActionListener implements ActionListener {  
 String number;  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 number = spawnDogs.getText();  
 try{  
 window.setN2(Integer.*parseInt*(number));  
 }catch (Throwable number){  
 JOptionPane.*showMessageDialog*(null, "Ввод некорректного значения!!!","Error",JOptionPane.*ERROR\_MESSAGE*);  
 }  
 }  
 }  
  
 private class historyActionListener implements ActionListener {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 JTextArea display = new JTextArea(21,33);  
 display.setEditable(false);  
  
  
 for (int i = 0; i<window.quantityPet;i++){  
  
 if(Singleton.*getInstance*().Get(i) instanceof Cat){  
 display.append("Номер: "+i+"\n");  
 display.append("Петомец: Кошка \n");  
  
 }else {  
 display.append("Номер: "+i+"\n");  
 display.append("Петомец: Собака\n");  
 }  
 display.append("Время рождения: "+Singleton.*getInstance*().Get(i).getTimeOfBirth()+"\n");  
 display.append("Id питомца: "+Singleton.*getInstance*().Get(i).getID()+"\n");  
 }  
 JFrame infFrame = new JFrame("Питомцы");  
 infFrame.setSize(400,400);  
 infFrame.setLocationRelativeTo(null);  
 infFrame.setResizable(false);  
 JPanel middlePanel=new JPanel();  
 JScrollPane scroll = new JScrollPane(display);  
 scroll.setVerticalScrollBarPolicy(ScrollPaneConstants.*VERTICAL\_SCROLLBAR\_ALWAYS*);  
 middlePanel.add(scroll);  
 infFrame.add(middlePanel);  
 infFrame.setVisible(true);  
 }  
 }  
 private class catAIActionListener implements ActionListener {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if (catAI.isFlag()) {  
 catAI.startAI();  
 } else {  
 catAI.stopAI();  
 }  
  
 }  
 }  
 private class dogAIActionListener implements ActionListener {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 if (dogAI.isFlag()) {  
 dogAI.startAI();  
 } else {  
 dogAI.stopAI();  
 }  
 }  
 }  
 public class MyTimerTask {  
 public Timer timer = new Timer(true);  
  
 public void StartWork() {  
 TimerTask timerTask = new TimerTask() {  
 @Override  
 public void run() {  
  
 if(myTimer != 0)  
 window.update(myTimer);  
  
  
 myTimer++;  
  
 repaint();  
 }  
 };  
  
 timer.scheduleAtFixedRate(timerTask, 0, 1000);  
 }  
  
 public void StopWork() {  
 timer.cancel();  
 }  
 public void WaitWork() {  
 timer.cancel();  
 timer = new Timer();  
 }  
 }  
 private class priorityCatAIActionListener implements ActionListener {  
 int num;  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 num = priorityCatAI.getSelectedIndex();  
 switch (num) {  
 case 0:  
 catAI.setPriority(1);  
 break;  
 case 1:  
 catAI.setPriority(2);  
 break;  
 case 2:  
 catAI.setPriority(3);  
 break;  
 case 3:  
 catAI.setPriority(4);  
 break;  
 case 4:  
 catAI.setPriority(5);  
 break;  
 case 5:  
 catAI.setPriority(6);  
 break;  
 case 6:  
 catAI.setPriority(7);  
 break;  
 case 7:  
 catAI.setPriority(8);  
 break;  
 case 8:  
 catAI.setPriority(9);  
 break;  
 case 9:  
 catAI.setPriority(10);  
 break;  
 }  
 requestFocusInWindow();  
 }  
  
 }  
 private class priorityDogAIActionListener implements ActionListener {  
 int num;  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 num = priorityDogAI.getSelectedIndex();  
 switch (num) {  
 case 0:  
 dogAI.setPriority(1);  
 break;  
 case 1:  
 dogAI.setPriority(2);  
 break;  
 case 2:  
 dogAI.setPriority(3);  
 break;  
 case 3:  
 dogAI.setPriority(4);  
 break;  
 case 4:  
 catAI.setPriority(5);  
 break;  
 case 5:  
 dogAI.setPriority(6);  
 break;  
 case 6:  
 dogAI.setPriority(7);  
 break;  
 case 7:  
 dogAI.setPriority(8);  
 break;  
 case 8:  
 dogAI.setPriority(9);  
 break;  
 case 9:  
 dogAI.setPriority(10);  
 break;  
 }  
 requestFocusInWindow();  
 }  
  
 }  
}

Main

package com.company;  
  
  
import com.company.Habitat.Habitat;  
import com.company.Single.Singleton;  
  
public class Main{  
  
  
 public static void main(String[] args) {  
 Habitat myHabitat = new Habitat(1200,1000);  
 Singleton.*getInstance*().setHabitat(myHabitat);  
 MyJFrame MyJFrame = new MyJFrame(myHabitat);  
  
  
  
 }  
}

MyFrame

package com.company;  
  
import com.company.Habitat.Habitat;  
  
import javax.swing.\*;  
  
  
public class MyJFrame extends JFrame {  
  
 MyJFrame(Habitat window){  
  
 JFrame jFrame = new JFrame("Cats and Dogs");  
 jFrame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 jFrame.setResizable(false);  
  
 GUI myGUI = new GUI(window,jFrame);  
 jFrame.add(myGUI);  
 MyMenuBar myMenuBar = new MyMenuBar(myGUI);  
 jFrame.setMenuBar(myMenuBar);  
  
  
  
  
  
  
 jFrame.setVisible(true);  
 jFrame.setSize(window.getSizeX(),window.getSizeY());  
 jFrame.setLocationRelativeTo(null);  
 }  
}

MyMenuBar

package com.company;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
  
public class MyMenuBar extends MenuBar {  
 MyMenuBar(GUI myGUI){  
 MenuBar newMenu = new MenuBar();  
 Menu menu = new Menu("Menu");  
 add(menu);  
 MenuItem runTimer = new MenuItem("Run");  
 MenuItem stopTimer = new MenuItem("Stop");  
 MenuItem endApp = new MenuItem("Shut down");  
 menu.add(runTimer);  
 menu.add(stopTimer);  
 menu.add(new MenuItem("-"));  
 menu.add(endApp);  
  
  
 Menu settings = new Menu("Settings");  
 add(settings);  
 MenuItem hideOrShowTime = new MenuItem("Timer On/Off");  
 MenuItem showInfoWhenPressStop = new MenuItem("Show info when press stop");  
 Menu selectSpawnChanceCats = new Menu("Spawn chance cats");  
 MenuItem cats0 = new MenuItem("0%");  
 MenuItem cats1 = new MenuItem("10%");  
 MenuItem cats2 = new MenuItem("20%");  
 MenuItem cats3 = new MenuItem("30%");  
 MenuItem cats4 = new MenuItem("40%");  
 MenuItem cats5 = new MenuItem("50%");  
 MenuItem cats6 = new MenuItem("60%");  
 MenuItem cats7 = new MenuItem("70%");  
 MenuItem cats8 = new MenuItem("80%");  
 MenuItem cats9 = new MenuItem("90%");  
 MenuItem cats10 = new MenuItem("100%");  
 selectSpawnChanceCats.add(cats0);  
 selectSpawnChanceCats.add(cats1);  
 selectSpawnChanceCats.add(cats2);  
 selectSpawnChanceCats.add(cats3);  
 selectSpawnChanceCats.add(cats4);  
 selectSpawnChanceCats.add(cats5);  
 selectSpawnChanceCats.add(cats6);  
 selectSpawnChanceCats.add(cats7);  
 selectSpawnChanceCats.add(cats8);  
 selectSpawnChanceCats.add(cats9);  
 selectSpawnChanceCats.add(cats10);  
 Menu selectSpawnChanceDogs = new Menu("Spawn chance dogs");  
 MenuItem dogs0 = new MenuItem("0%");  
 MenuItem dogs1 = new MenuItem("10%");  
 MenuItem dogs2 = new MenuItem("20%");  
 MenuItem dogs3 = new MenuItem("30%");  
 MenuItem dogs4 = new MenuItem("40%");  
 MenuItem dogs5 = new MenuItem("50%");  
 MenuItem dogs6 = new MenuItem("60%");  
 MenuItem dogs7 = new MenuItem("70%");  
 MenuItem dogs8 = new MenuItem("80%");  
 MenuItem dogs9 = new MenuItem("90%");  
 MenuItem dogs10 = new MenuItem("100%");  
 selectSpawnChanceDogs.add(dogs0);  
 selectSpawnChanceDogs.add(dogs1);  
 selectSpawnChanceDogs.add(dogs2);  
 selectSpawnChanceDogs.add(dogs3);  
 selectSpawnChanceDogs.add(dogs4);  
 selectSpawnChanceDogs.add(dogs5);  
 selectSpawnChanceDogs.add(dogs6);  
 selectSpawnChanceDogs.add(dogs7);  
 selectSpawnChanceDogs.add(dogs8);  
 selectSpawnChanceDogs.add(dogs9);  
 selectSpawnChanceDogs.add(dogs10);  
 Menu selectSpawnChancePets = new Menu("Spawn chance");  
 selectSpawnChancePets.add(selectSpawnChanceCats);  
 selectSpawnChancePets.add(selectSpawnChanceDogs);  
  
 MenuItem changeSpawnCats = new MenuItem("Change spawn time cats");  
 MenuItem changeSpawnDogs = new MenuItem("Change spawn time dogs");  
 Menu spawnTime = new Menu("Change spawn time");  
 spawnTime.add(changeSpawnCats);  
 spawnTime.add(changeSpawnDogs);  
 MenuItem lifeTimeCats = new MenuItem("Select life time of cats");  
 MenuItem lifeTimeDogs = new MenuItem("Select life time of dogs");  
 Menu lifeTime = new Menu("Select life time");  
 lifeTime.add(lifeTimeCats);  
 lifeTime.add(lifeTimeDogs);  
  
  
  
 settings.add(hideOrShowTime);  
 settings.add(showInfoWhenPressStop);  
 settings.add(new MenuItem("-"));  
 settings.add(selectSpawnChancePets);  
 settings.add(new MenuItem("-"));  
 settings.add(spawnTime);  
 settings.add(new MenuItem("-"));  
 settings.add(lifeTime);  
  
  
  
 runTimer.addActionListener(e -> myGUI.run.doClick());  
 stopTimer.addActionListener(e -> myGUI.pause.doClick());  
 hideOrShowTime.addActionListener(e -> {  
 myGUI.timeVisible = !myGUI.timeVisible;  
 if(myGUI.timeVisible){  
 myGUI.showTime.setSelected(true);  
 myGUI.hideTime.setSelected(false);  
 }else{  
 myGUI.showTime.setSelected(false);  
 myGUI.hideTime.setSelected(true);  
 }  
  
 });  
 showInfoWhenPressStop.addActionListener(e -> {  
 myGUI.useDialog = !myGUI.useDialog;  
 myGUI.screenResolution.setSelected(!myGUI.useDialog);  
 });  
 endApp.addActionListener(e -> {  
 myGUI.timerTask.StopWork();  
 myGUI.useKeys = false;  
  
 myGUI.timeVisible = false;  
 myGUI.endApp = true;  
 myGUI.lineVisible = false;  
 myGUI.selectTime.setVisible(false);  
 myGUI.showTime.setVisible(false);  
 myGUI.hideTime.setVisible(false);  
 myGUI.run.setVisible(false);  
 myGUI.pause.setVisible(false);  
 myGUI.screenResolution.setVisible(false);  
 myGUI.selectCatsTimeSpawn.setVisible(false);  
 myGUI.selectDogsTimeSpawn.setVisible(false);  
 myGUI.selectCatsSpawn.setVisible(false);  
 myGUI.selectDogsSpawn.setVisible(false);  
 myGUI.spawnCats.setVisible(false);  
 myGUI.spawnDogs.setVisible(false);  
 myGUI.catsComboBox.setVisible(false);  
 myGUI.dogsComboBox.setVisible(false);  
 myGUI.window.allClear();  
  
 myGUI.repaint();  
 });  
 cats0.addActionListener(e -> {  
 myGUI.window.setP1(0);  
 myGUI.catsComboBox.setSelectedIndex(0);  
 });  
 cats1.addActionListener(e -> {  
 myGUI.window.setP1(10);  
 myGUI.catsComboBox.setSelectedIndex(1);  
 });  
 cats2.addActionListener(e -> {  
 myGUI.window.setP1(20);  
 myGUI.catsComboBox.setSelectedIndex(2);  
 });  
 cats3.addActionListener(e -> {  
 myGUI.window.setP1(30);  
 myGUI.catsComboBox.setSelectedIndex(3);  
 });  
 cats4.addActionListener(e -> {  
 myGUI.window.setP1(40);  
 myGUI.catsComboBox.setSelectedIndex(4);  
 });  
 cats5.addActionListener(e -> {  
 myGUI.window.setP1(50);  
 myGUI.catsComboBox.setSelectedIndex(5);  
 });  
 cats6.addActionListener(e -> {  
 myGUI.window.setP1(60);  
 myGUI.catsComboBox.setSelectedIndex(6);  
 });  
 cats7.addActionListener(e -> {  
 myGUI.window.setP1(70);  
 myGUI.catsComboBox.setSelectedIndex(7);  
 });  
 cats8.addActionListener(e -> {  
 myGUI.window.setP1(80);  
 myGUI.catsComboBox.setSelectedIndex(8);  
 });  
 cats9.addActionListener(e -> {  
 myGUI.window.setP1(90);  
 myGUI.catsComboBox.setSelectedIndex(9);  
 });  
 cats10.addActionListener(e -> {  
 myGUI.window.setP1(100);  
 myGUI.catsComboBox.setSelectedIndex(10);  
 });  
 dogs0.addActionListener(e -> {  
 myGUI.window.setP2(0);  
 myGUI.dogsComboBox.setSelectedIndex(0);  
 });  
 dogs1.addActionListener(e -> {  
 myGUI.window.setP2(10);  
 myGUI.dogsComboBox.setSelectedIndex(1);  
 });  
 dogs2.addActionListener(e -> {  
 myGUI.window.setP2(20);  
 myGUI.dogsComboBox.setSelectedIndex(2);  
 });  
 dogs3.addActionListener(e -> {  
 myGUI.window.setP2(30);  
 myGUI.dogsComboBox.setSelectedIndex(3);  
 });  
 dogs4.addActionListener(e -> {  
 myGUI.window.setP2(40);  
 myGUI.dogsComboBox.setSelectedIndex(4);  
 });  
 dogs5.addActionListener(e -> {  
 myGUI.window.setP2(50);  
 myGUI.dogsComboBox.setSelectedIndex(5);  
 });  
 dogs6.addActionListener(e -> {  
 myGUI.window.setP2(60);  
 myGUI.dogsComboBox.setSelectedIndex(6);  
 });  
 dogs7.addActionListener(e -> {  
 myGUI.window.setP2(70);  
 myGUI.dogsComboBox.setSelectedIndex(7);  
 });  
 dogs8.addActionListener(e -> {  
 myGUI.window.setP2(80);  
 myGUI.dogsComboBox.setSelectedIndex(8);  
 });  
 dogs9.addActionListener(e -> {  
 myGUI.window.setP2(90);  
 myGUI.dogsComboBox.setSelectedIndex(9);  
 });  
 dogs10.addActionListener(e -> {  
 myGUI.window.setP2(100);  
 myGUI.dogsComboBox.setSelectedIndex(10);  
 });  
 changeSpawnCats.addActionListener(e -> {  
 String result = JOptionPane.showInputDialog(  
 null,  
 "Введите период рождения.","Рождение котов",JOptionPane.INFORMATION\_MESSAGE);  
 try{  
 myGUI.window.setN1(Integer.parseInt(result));  
 myGUI.spawnCats.setText(String.valueOf(myGUI.window.getN1()));  
 }catch (Throwable number){  
 JOptionPane.showMessageDialog(null, "Ввод некорректного значения!!!","Error",JOptionPane.ERROR\_MESSAGE);  
 }  
 });  
 changeSpawnDogs.addActionListener(e -> {  
 String result = JOptionPane.showInputDialog(  
 null,  
 "Введите период рождения.","Рождение собак",JOptionPane.INFORMATION\_MESSAGE);  
 try{  
 myGUI.window.setN2(Integer.parseInt(result));  
 myGUI.spawnDogs.setText(String.valueOf(myGUI.window.getN2()));  
 }catch (Throwable number){  
 JOptionPane.showMessageDialog(null, "Ввод некорректного значения!!!","Error",JOptionPane.ERROR\_MESSAGE);  
 }  
 });  
  
 lifeTimeCats.addActionListener(e -> {  
 String result = JOptionPane.*showInputDialog*(  
 null,  
 "Введите время жизни.","Жизнь котов",JOptionPane.*INFORMATION\_MESSAGE*);  
 try{  
 myGUI.window.setCatsTimeOfLife(Integer.*parseInt*(result));  
 myGUI.lifeCats.setText(String.*valueOf*(myGUI.window.getCatsTimeOfLife()));  
 }catch (Throwable number){  
 JOptionPane.*showMessageDialog*(null, "Ввод некорректного значения!!!","Error",JOptionPane.*ERROR\_MESSAGE*);  
 }  
 });  
 lifeTimeDogs.addActionListener(e -> {  
 String result = JOptionPane.*showInputDialog*(  
 null,  
 "Введите время жизни.","Жизнь котов",JOptionPane.*INFORMATION\_MESSAGE*);  
 try{  
 myGUI.window.setDogsTimeOfLife(Integer.*parseInt*(result));  
 myGUI.lifeDogs.setText(String.*valueOf*(myGUI.window.getDogsTimeOfLife()));  
 }catch (Throwable number){  
 JOptionPane.*showMessageDialog*(null, "Ввод некорректного значения!!!","Error",JOptionPane.*ERROR\_MESSAGE*);  
 }  
 });  
 }  
}

infDialog

package com.company;  
  
import com.company.Habitat.Habitat;  
  
import javax.swing.\*;  
import java.awt.\*;  
  
public class infDialog extends JDialog {  
 private JButton okButton;  
 private JButton closeButton;  
  
 private JPanel info;  
 private JTextArea textTime;  
 private JPanel panel1;  
 public infDialog(JFrame owner, Habitat window, int time){  
 super(owner,"Information",true);  
 setResizable(false);  
  
 JPanel panelText = new JPanel();  
 int quantityPet = window.dogs + window.cats;  
 textTime.setText("Время работы: " + time +"\nКоличество кошек: " + window.cats + "\nКоличество собак: " + window.dogs + "\nВсего питомцев: " + quantityPet);  
 textTime.setEnabled(false);  
  
  
  
 panelText.add(textTime);  
  
 JPanel panel = new JPanel();  
 panel.setLayout(new FlowLayout());  
  
 //add(new MyComponent(thisWindow,time));  
  
  
 okButton.addActionListener(e -> {  
 owner.dispose();  
 System.*exit*(0);  
 });  
  
 closeButton.addActionListener(e -> dispose());  
  
  
  
 panel.add(okButton);  
 panel.add(closeButton);  
 add(panelText,BorderLayout.*NORTH*);  
 add(panel, BorderLayout.*SOUTH*);  
 setSize(400,400);  
 setLocationRelativeTo(null);  
 setVisible(true);  
 }  
}

Habitat

package com.company.Habitat;  
  
  
import com.company.Habitat.Pet.Creatures.Cat;  
import com.company.Habitat.Pet.Creatures.Dog;  
import com.company.Single.Singleton;  
  
  
import java.util.Random;  
  
  
public class Habitat {  
 private int sizeX;  
 private int sizeY;  
 private int N1 = 5, P1 = 70;  
 private int N2 = 3, P2 = 40;  
  
 private int petsID;  
  
 private int catsTimeOfLife = 3;  
 private int dogsTimeOfLife = 2;  
  
 public int quantityPet = 0;  
 public int allSpawn = 0;  
 public int cats,dogs;  
  
  
 public Habitat(int x, int y){  
 sizeX = x;  
 sizeY = y;  
 }  
 public int getSizeX() { return sizeX; }  
 public void setSizeX(int sizeX) { this.sizeX = sizeX; }  
 public int getSizeY() { return sizeY; }  
 public void setSizeY(int sizeY) { this.sizeY = sizeY; }  
  
 public void update(int time) {  
  
 if ((new Random().nextInt(100) < P1) && (time % N1 == 0)) {  
  
 petsID = (new Random().nextInt(65550));  
 Cat myCat = new Cat(petsID,time);  
 myCat.setX(new Random().nextInt(sizeX - 100));  
 if(myCat.getX()<50)myCat.setY(myCat.getX() + 100);  
 myCat.setY(new Random().nextInt(sizeY - 100));  
 if(myCat.getY()<50)myCat.setY(myCat.getY() + 100);  
  
 Singleton.*getInstance*().SetID(petsID);  
 Singleton.*getInstance*().SetHashMap(allSpawn,myCat.getTimeOfBirth());  
 Singleton.*getInstance*().Add(myCat);  
 quantityPet++;  
 allSpawn++;  
 cats++;  
 }  
 if ((new Random().nextInt(100) < P2) && (time % N2 == 0)) {  
 petsID = (new Random().nextInt(65550));  
 Dog myDog = new Dog(petsID,time);  
 myDog.setX(new Random().nextInt(sizeX - 100));  
 if(myDog.getX()<50)myDog.setX(myDog.getX() + 100);  
 myDog.setY(new Random().nextInt(sizeY - 100));  
 if(myDog.getY()<50)myDog.setY(myDog.getY() + 100);  
  
 Singleton.*getInstance*().SetID(petsID);  
 Singleton.*getInstance*().SetHashMap(allSpawn,myDog.getTimeOfBirth());  
 Singleton.*getInstance*().Add(myDog);  
 allSpawn++;  
 quantityPet++;  
 dogs++;  
 }  
  
 TimeToDie(time);  
 }  
  
 private void TimeToDie(int time){  
 for (int i = 0; i < quantityPet; i++) {  
 if (Dog.class.isAssignableFrom(Singleton.*getInstance*().Get(i).getClass())) {  
 if(Singleton.*getInstance*().Get(i).getTimeOfBirth() + dogsTimeOfLife < time){  
 Singleton.*getInstance*().Delete(i);  
 quantityPet--;  
  
 }  
  
 } else {  
 if(Singleton.*getInstance*().Get(i).getTimeOfBirth() + catsTimeOfLife < time){  
 Singleton.*getInstance*().Delete(i);  
 quantityPet--;  
  
 }  
 }  
  
 }  
  
  
 }  
  
  
 public void allClear(){  
 Singleton.*getInstance*().Clear();  
 quantityPet = 0;  
 }  
  
  
 public void setP1(int p1) {  
 P1 = p1;  
 }  
  
 public void setP2(int p2) {  
 P2 = p2;  
 }  
  
 public int getN1() {  
 return N1;  
 }  
  
 public int getN2() {  
 return N2;  
 }  
  
 public void setN1(int n1) {  
 N1 = n1;  
 }  
  
 public void setN2(int n2) {  
 N2 = n2;  
 }  
  
 public void setCatsTimeOfLife(int catsTimeOfLife) {  
 this.catsTimeOfLife = catsTimeOfLife;  
 }  
  
 public void setDogsTimeOfLife(int dogsTimeOfLife) {  
 this.dogsTimeOfLife = dogsTimeOfLife;  
 }  
  
 public int getCatsTimeOfLife() {  
 return catsTimeOfLife;  
 }  
  
 public int getDogsTimeOfLife() {  
 return dogsTimeOfLife;  
 }  
}

Pet

package com.company.Habitat.Pet;  
  
public abstract class Pet implements IBehaviour {  
 private int x,y;  
 private int id;  
 private int timeOfBirth;  
 public Pet(int id, int birth){  
 this.id = id;  
 timeOfBirth = birth;  
 }  
  
  
 public void setX(int x) { this.x = x; }  
 public void setY(int y) { this.y = y; }  
 public int getX() { return x; }  
 public int getY() { return y; }  
 public int getID(){ return id; }  
 public int getTimeOfBirth() { return timeOfBirth; }  
}

IBehaviour

package com.company.Habitat.Pet;  
  
public interface IBehaviour {  
 int getX();  
 int getY();  
 void setX(int x);  
 void setY(int y);  
}

Dog

package com.company.Habitat.Pet.Creatures;  
  
import com.company.Habitat.Pet.Pet;  
import com.company.Single.Singleton;  
  
public class Dog extends Pet {  
  
 public Dog(int id, int birth) {  
 super(id,birth);  
 }  
 private int speed = 4;  
 private int stay;  
 private int go = 2000;  
 private int targetX;  
 private int targetY;  
 @Override  
 public void move() {  
 for(int i = 0; i< Singleton.*getInstance*().array.size(); i++){  
 if(Singleton.*getInstance*().Get(i) instanceof Cat){  
 stay = Singleton.*getInstance*().Get(i).getX() + getY() +Singleton.*getInstance*().Get(i).getY() + getY();  
 if(stay < go ){  
 targetX = Singleton.*getInstance*().Get(i).getX();  
 targetY = Singleton.*getInstance*().Get(i).getY();  
 }  
 }  
 }  
 if(getX() > targetX){  
 setX(getX()-speed);  
 }  
 if(getX() < targetX){  
 setX(getX()+speed);  
 }  
 if(getY() > targetY){  
 setY(getY()-speed);  
 }  
 if(getY() < targetY){  
 setY(getY()+speed);  
 }  
  
 }  
}

Cat

package com.company.Habitat.Pet.Creatures;  
  
import com.company.Habitat.Pet.Pet;  
import com.company.Single.Singleton;  
  
public class Cat extends Pet {  
 private int speed = 10;  
  
  
  
 private boolean isRight = false;  
 private boolean isBottom = false;  
 private boolean isLeft = false;  
 private boolean isTop = false;  
 public Cat(int id, int birth) {  
 super(id,birth);  
 }  
 @Override  
 public void move() {  
 if (!isRight && !isBottom && !isLeft && !isTop) {  
 if (getX() <= Singleton.*getInstance*().getWindow().getSizeX() - 80) {  
 setX(getX() + speed);  
 } else isRight = true;  
 }  
 if (isRight && !isBottom && !isLeft && !isTop) {  
 if (getY() <= Singleton.*getInstance*().getWindow().getSizeY() - 120) {  
 setY(getY() + speed);  
 } else isBottom = true;  
 }  
 if (isRight && isBottom && !isLeft && !isTop) {  
 if (getX() >= 10) {  
 setX(getX() - speed);  
 } else isLeft = true;  
 }  
 if (isRight && isBottom && isLeft && !isTop) {  
 if (getY() >= 130) {  
 setY(getY() - speed);  
 } else isTop = true;  
 }  
 if (isRight && isBottom && isLeft && isTop) {  
 isRight = false;  
 isBottom = false;  
 isLeft = false;  
 isTop = false;  
 }  
  
 }  
}

Singleton

package com.company.Single;  
  
import com.company.Habitat.Habitat;  
import com.company.Habitat.Pet.Pet;  
  
import java.util.ArrayList;  
import java.util.HashMap;  
import java.util.TreeSet;  
  
public class Singleton implements mySingle {  
 private static final Singleton *singleton* = new Singleton();  
 Habitat window;  
 public ArrayList<Pet> array = new ArrayList<>();  
 TreeSet<Integer> petsID = new TreeSet<>();  
 HashMap<Integer,Integer> petsLife = new HashMap<>();  
  
 private Singleton(){  
  
 }  
 public void setHabitat(Habitat habitat){  
 window = habitat;  
 }  
  
 public Habitat getWindow() {  
 return window;  
 }  
  
 public void SetID(int id){  
 petsID.add(id);  
 }  
 public void SetHashMap(int num,int lifeTime){  
 petsLife.put(num,lifeTime);  
 }  
  
 public void Add(Pet pet){  
 array.add(pet);  
 petsID.add(pet.getID());  
 }  
  
 public Pet Get(int number){  
 return array.get(number);  
  
 }  
  
 public void Clear(){  
 array.clear();  
 }  
 public void Delete(int i){  
 array.remove(i);  
 petsID.remove(i);  
 petsLife.remove(i);  
 }  
 public static Singleton getInstance(){  
 return *singleton*;  
 }  
}

mySingleton

package com.company.Single;  
  
import com.company.Habitat.Pet.Pet;  
  
public interface mySingle {  
 void Add(Pet pet);  
 Pet Get(int number);  
 void Clear();  
  
}

BaseAI

package com.company.BaseAI;  
  
  
public abstract class BaseAI implements Runnable {  
 protected int priority = 1;  
 protected boolean flag;  
 protected Thread t;  
 public void run() {}  
 public void stopAI() {}  
 public synchronized void startAI() {}  
 public void setPriority(int priority) {}  
}

CatAI

package com.company.BaseAI.AI;  
  
import com.company.BaseAI.BaseAI;  
import com.company.Habitat.Pet.Creatures.Cat;  
import com.company.Single.Singleton;  
  
  
public class CatAI extends BaseAI {  
 public CatAI() {  
 t=new Thread(this);  
 flag= false;  
 t.start();  
 }  
  
 public void run() {  
  
 while (true) {  
 try {  
 synchronized (this) {  
 while (flag) {  
 wait();  
 }  
 }  
 for (int i = 0; i < Singleton.*getInstance*().array.size(); i++) {  
 if (Singleton.*getInstance*().Get(i) instanceof Cat) {  
 Singleton.*getInstance*().Get(i).move();  
 }  
 }  
  
 Thread.*sleep*(100);  
 } catch (InterruptedException ie) {  
 }  
 }  
 }  
 public void stopAI() {  
 flag = true;  
 }  
 public synchronized void startAI() {  
 flag = false;  
 notify();  
 }  
 public boolean isFlag(){  
 return flag;  
 }  
 public void setPriority(int priority){  
 this.priority = priority;  
 }  
}

DogAI

package com.company.BaseAI.AI;  
  
import com.company.BaseAI.BaseAI;  
import com.company.Habitat.Pet.Creatures.Dog;  
import com.company.Single.Singleton;  
  
public class DogAI extends BaseAI {  
 public DogAI() {  
 t=new Thread(this);  
 flag= false;  
 t.start();  
 }  
  
 public void run() {  
  
 while (true) {  
 try {  
 synchronized (this) {  
 while (flag) {  
 wait();  
 }  
 }  
 for (int i = 0; i < Singleton.*getInstance*().array.size(); i++) {  
 if (Singleton.*getInstance*().Get(i) instanceof Dog) {  
 Singleton.*getInstance*().Get(i).move();  
 }  
 }  
  
 Thread.*sleep*(100);  
 } catch (InterruptedException ie) {  
 }  
 }  
 }  
 public void stopAI() {  
 flag = true;  
 }  
 public synchronized void startAI() {  
 flag = false;  
 notify();  
 }  
 public boolean isFlag(){  
 return flag;  
 }  
 public void setPriority(int priority){  
 this.priority = priority;  
 }  
}