**Homework Durnev Alex**

**Project: Tamagochi**

**Topic: Serialization, Deserialization, Thread, Unit test**

**Задание.**

Добавить в игру «Тамагочи» из предыдущего ДЗ синхронизацию состояния питомца в файле через механизм сериализации.

Для этого необходимо написать службу синхронизации игры с ассоциированным файлом, которая будет запускаться в отдельном фоновом потоке.

Последовательность работы:

1. В основном потоке создается объект службы синхронизации. Через него читается предыдущее состояние (десериализация), сохраненное в специальном файле. Если файла нет – то игра начинается сначала, создается пустой файл.
2. Запускается поток синхронизации, который с заданной периодичностью выполняет сохранение состояния животного (тамагочи) в файле используя сериализацию Java.
3. Перед выходом из игры основной поток вызывает прерывание interrupt для службы синхронизации и ожидает join завершения последней синхронизации состояния данной службы

**Дополнительно по желанию можно реализовать следующие компоненты:**

1. Цикл изменения параметров питомца в отдельном потоке/потоках – голод, здоровье и т.д. изменять в отдельных потоках (здесь необходимо позаботиться о синхронизации и предотвратить гонки данных).

2. Можно помимо данных питомца выполнять сериализацию времени выхода из игры, а при запуске игры считывать это время и изменять параметры питомца в соответствии с прошедшим временем. Например, если пользователь очень долго не заходил в игру – то питомец может умереть.

**Предыдущее задание (**теперь можно делать сразу вместе с этим):

Реализовать игру-симуляцию «Тамагочи» (<https://ru.wikipedia.org/wiki/%D0%A2%D0%B0%D0%BC%D0%B0%D0%B3%D0%BE%D1%87%D0%B8>) на языке Java. Для этого сделать:

* Класс своего питомца, с которым можно выполнять игровые действия
* Класс самой игры, внутри которой есть питомцы, CLI (без псевдографики, простенький)
* При необходимости доп классы.

Реализовать сериализацию состояния игры (класс питомца и, при необходимости, доп классы), чтобы после закрытия программы и повторного запуска игра продолжалась. Использовать стандартную сериализацию Java.

Source code

Class Tamagochi

package com.company;  
  
import java.io.Serializable;  
  
// Class "Tamagotchi"  
public class Tamagochi implements Serializable {  
  
 // Fields  
 private int age; // возраст  
 private int health; // здоровье  
 private int satiety; // сытость  
 private int clean; // чистота  
 private int bonus; // награда за действия  
  
 // empty constructor  
 public Tamagochi() {  
 this.age = 0;  
 this.health = 100;  
 this.satiety = 100;  
 this.clean = 100;  
 this.bonus = 0;  
 }  
  
 // constructor with parameters  
 public Tamagochi (int age, int health, int satiety, int clean, int bonus) {  
 this.age = age;  
 this.health = health;  
 this.satiety = satiety;  
 this.clean = clean;  
 this.bonus = bonus;  
 }  
  
 // Getters  
 public int getAge() {  
 return age;  
 }  
 public int getHealth() {  
 return health;  
 }  
 public int getSatiety() {  
 return satiety;  
 }  
 public int getClean() {  
 return clean;  
 }  
 public int getBonus() {  
 return bonus;  
 }  
  
 // Setters  
 public void setAge(int age) {  
 this.age = age;  
 }  
 public void setHealth(int health) {  
 this.health = health;  
 }  
 public void setSatiety(int satiety) {  
 this.satiety = satiety;  
 }  
 public void setClean(int clean) {  
 this.clean = clean;  
 }  
 public void setBonus(int bonus) {  
 this.bonus = bonus;  
 }  
  
 // Method to play the game  
 public void play() {  
 if (isAlive()) {  
 health -= 10;  
 clean -= 10;  
 satiety -= 10;  
 bonus += 1;  
 } else  
 System.*out*.println("Tamagotchi died");  
// age++;  
 }  
  
 // Method goes to the bed  
 public void goToSleep() {  
 if(isAlive()) {  
 health = 100;  
 clean -= 10;  
 satiety = 100;  
 bonus += 1;  
 age++;  
 } else  
 System.*out*.println("Tamagotchi died");  
 }  
  
 // Method to feed Tamagotchi  
 public void feed(){  
 if(isAlive()) {  
 health = 100;  
 clean -= 10;  
 satiety = 100;  
 bonus += 1;  
 }else  
 System.*out*.println("Tamagotchi died");  
 }  
  
 // Method to wash Tamagotchi  
 public void takeAShower() {  
 if(isAlive()) {  
 health -= 10;  
 clean = 100;  
 satiety -= 10;  
 bonus += 1;  
 } else  
 System.*out*.println("Tamagotchi died");  
 }  
  
 // Method to check alive my tamagochi or no  
 public boolean isAlive() {  
 return health >= 10 || satiety >= 10;  
 }  
  
 @Override  
 public String toString () {  
 return "Tamagotchi's health " + health + "% , he is satiety on " + satiety  
 + "% , he clean on " + clean + "% , his age = " + age + " bonus = " + bonus;  
 }  
  
} // Class "Tamagotchi" - end

Class TamagochiSerialization

package com.company;  
  
import java.io.\*;  
  
public class TamagochiSerialization {  
  
 private final String fileName;  
  
 public TamagochiSerialization(String fileName) {  
 this.fileName = fileName;  
 }  
  
 // Method to serialization game  
 public static void serialize(Tamagochi tam) {  
 try (FileOutputStream fos = new FileOutputStream("Tamagoch.txt");  
 ObjectOutputStream oos = new ObjectOutputStream(fos)) {  
 oos.writeObject(tam);  
// System.out.println("Data is synchronized");  
 } catch (IOException ex) {  
 System.*out*.println("Exception " + ex);  
 }  
 }  
  
 // Method to deserialization game  
 public static Tamagochi deserialization() {  
 try(FileInputStream fis = new FileInputStream("Tamagoch.txt");  
 ObjectInputStream ois = new ObjectInputStream(fis)) {  
 return (Tamagochi) ois.readObject();  
 } catch (IOException ex) {  
 System.*out*.println("Exception " + ex);  
 } catch (ClassNotFoundException ex) {  
 System.*out*.println("Deserialization exception: " + ex);  
 }  
 return null;  
 }  
}

Class TamagochiSave

package com.company;  
  
public class TamagochiSave extends Thread {  
  
 // Fields  
 private final String fileName; // the name of the file associated with this synchronizer  
 private Tamagochi tamagotchi; // tamagotchi field  
 private final int timeInterval; // time interval to wait syncing  
  
 public TamagochiSave (Tamagochi tamagotchi, String fileName, int timeInterval) {  
 this.tamagotchi = tamagotchi;  
 this.fileName = fileName;  
 this.timeInterval = timeInterval;  
 }  
  
 // Getter  
 public Tamagochi getTamagotchi() {  
 return tamagotchi;  
 }  
  
 // Method to launch sync  
 public void run() {  
 // The main cycle background task  
 while (true) {  
 // 1. Write information in file  
 TamagochiSerialization.*serialize*(tamagotchi);  
 //System.out.println("Data is synchronized");  
 try {  
 Thread.*sleep*(timeInterval);  
 } catch (InterruptedException ex) {  
 System.*out*.println("An error occurred during sleep: " + ex.getMessage());  
 interrupt();  
 }  
 if(isInterrupted()) {  
 TamagochiSerialization.*serialize*(tamagotchi);  
 System.*out*.println("Synchronizer stopped.");  
 break;  
 }  
 }  
 }  
}

Main

package com.company;  
  
import java.io.File;  
import java.util.Scanner;  
  
public class Main {  
  
 public static Tamagochi checkExists(String fileName) {  
 File file = new File(fileName);  
 if (!file.exists()) {  
 return new Tamagochi ();  
 }  
 return TamagochiSerialization.*deserialization*();  
 }  
  
 public static void main(String[] args) throws InterruptedException {  
  
 String select;  
 Scanner in = new Scanner(System.*in*);  
// Tamagochi tam = new Tamagochi();  
  
 Tamagochi tam = TamagochiSerialization.*deserialization*();  
  
 TamagochiSave tamagochiSave = new TamagochiSave  
 (*checkExists*("Tamagoch.txt"), "Tamagoch.txt", 3000);  
  
 //Tamagochi tam = TamagochiSynchronizer.deserialization();  
  
  
 // Create new object "tamagotchi"  
// TamagochiSynchronizer tamagochi =  
// new TamagochiSynchronizer(tam,"Tamagochi.txt", 3000);  
  
 // additional thread by daemon  
 tamagochiSave.setDaemon(true);  
  
 // Create tamagotchi  
 tam = tamagochiSave.getTamagotchi();  
  
 // launch synchronizer in a separate thread  
 tamagochiSave.start();  
  
 // Create CLI  
 label:  
 while(true) {  
 System.*out*.println("1 - Create new Pet ");  
 System.*out*.println("2 - Show your pet");  
 System.*out*.println("3 - Play with your pet");  
 System.*out*.println("4 - Feed your pet");  
 System.*out*.println("5 - Wash your pet");  
 System.*out*.println("6 - Go to the bed");  
 System.*out*.println("0 - Exit");  
 select = in.nextLine();  
 switch(select) {  
 case "1":  
 tam = new Tamagochi();  
 System.*out*.println(tam);  
 break;  
 case "2":  
 System.*out*.println(tam);  
 break;  
 case "3":  
 tam.play();  
 System.*out*.println(tam);  
 break;  
 case "4":  
 tam.feed();  
 System.*out*.println(tam);  
 break;  
 case "5":  
 tam.takeAShower();  
 System.*out*.println(tam);  
 break;  
 case "6":  
 tam.goToSleep();  
 System.*out*.println(tam);  
 case "0":  
 System.*out*.println("Goodbye");  
 break label;  
 } // switch  
 } // while  
  
 in.close();// close scanner  
  
 tamagochiSave.interrupt(); // interrupting a separate synchronization thread  
  
 tamagochiSave.join(); // waiting for the self-completion of the thread  
  
 System.*out*.println("Sub thread stopped, all data saved"); // thread has ended, the data is saved  
 } // main  
  
} // Main

Test

Create tamagochi, play with him, eat, take a shower etc.

Изображение выглядит как текст

Автоматически созданное описание Изображение выглядит как текст

Автоматически созданное описание Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание Изображение выглядит как текст

Автоматически созданное описание Изображение выглядит как текст

Автоматически созданное описание

Exit from game and launch the game again and we see not new tamagachi, but we see tamagochi which we played previously game

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание Изображение выглядит как текст

Автоматически созданное описание

**UNIT TEST**

Class TamagochiTest

package com.company;  
  
import org.junit.jupiter.api.BeforeEach;  
import org.junit.jupiter.api.Test;  
  
import static org.junit.jupiter.api.Assertions.\*;  
  
class TamagochiTest {  
  
 static Tamagochi *tam* = null;  
  
 @BeforeEach  
 public void createTam() {  
 *tam* = new Tamagochi(0, 100, 100, 100, 0);  
 }  
  
 @Test  
 void play() {  
 *tam*.play();  
 *assertEquals*(0, *tam*.getAge());  
 *assertEquals*(90, *tam*.getHealth());  
 *assertEquals*(90, *tam*.getSatiety());  
 *assertEquals*(90, *tam*.getClean());  
 *assertEquals*(1, *tam*.getBonus());  
 }  
  
 @Test  
 void goToSleep() {  
 *tam*.goToSleep();  
 *assertEquals*(1, *tam*.getAge());  
 *assertEquals*(100, *tam*.getHealth());  
 *assertEquals*(100, *tam*.getSatiety());  
 *assertEquals*(90, *tam*.getClean());  
 *assertEquals*(1, *tam*.getBonus());  
 }  
  
 @Test  
 void feed() {  
 *tam*.feed();  
 *assertEquals*(100, *tam*.getHealth()); // health  
 *assertEquals*(90, *tam*.getClean()); // clean  
 *assertEquals*(100, *tam*.getSatiety()); // satiete  
 *assertEquals*(1, *tam*.getBonus()); // bonus  
 }  
  
 @Test  
 void takeAShower() {  
 *tam*.takeAShower();  
 *assertEquals*(90, *tam*.getHealth());  
 *assertEquals*(100, *tam*.getClean());  
 *assertEquals*(90, *tam*.getSatiety());  
 *assertEquals*(1, *tam*.getBonus());  
 }  
  
 @Test  
 void isAlive() {  
 *tam*.isAlive();  
 *assertEquals*(10 , 10);  
 *assertEquals*(10, 10);  
 }  
}

TEST

Изображение выглядит как текст

Автоматически созданное описание

Class TamagochiSerializationTest

package com.company;  
  
import org.junit.jupiter.api.BeforeAll;  
import org.junit.jupiter.api.BeforeEach;  
import org.junit.jupiter.api.Test;  
  
import java.io.File;  
import java.io.IOException;  
  
import static org.junit.jupiter.api.Assertions.\*;  
  
class TamagochiSerializationTest {  
  
 static Tamagochi *tamagochi* = null; // the tamagochi to test  
 static TamagochiSerialization *tam* = null; // sterilizer for testing  
 static String *fileName* = null; // associated file  
  
 @BeforeAll  
 public static void setUpClass() {  
 *tamagochi* = new Tamagochi(0, 100, 100, 100, 0);  
 *fileName* = "tam Test.txt";  
 }  
  
 @BeforeEach  
 public void setUp() {  
 *tam* = new TamagochiSerialization(*fileName*);  
 }  
  
 public void tearDown() {  
 File file = new File(*fileName*);  
 if (file.exists()) {  
 file.delete();  
 }  
 }  
  
 @Test  
 void serialize() {  
 // 1 check the efficiency  
 TamagochiSerialization.*serialize*(*tamagochi*);  
 // 2. check the correctness of serialization through deserialization  
 Tamagochi deserialized = null;  
 // doing deserialization  
 deserialized = TamagochiSerialization.*deserialization*();  
 *fail*();  
  
 *assertNotNull*(deserialized); // не null  
 *assertEquals*(*tamagochi*.getAge(), deserialized.getAge());  
 *assertEquals*(*tamagochi*.getHealth(), deserialized.getHealth());  
 *assertEquals*(*tamagochi*.getSatiety(), deserialized.getSatiety());  
 *assertEquals*(*tamagochi*.getClean(), deserialized.getClean());  
 *assertEquals*(*tamagochi*.getBonus(), deserialized.getBonus());  
 }  
  
 @Test  
 void deserialization() {  
 Tamagochi tamagochi = TamagochiSerialization.*deserialization*();  
  
 }  
}

TEST

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание