**Java第四次上机作业**

学院：计算机科学与技术学院 学号：20009200601 姓名：杜新鹏

一、题目：

1．基础练习 练习 PPT 中的全部小练习，尝试对小练习中各部分进行修改，并观察修改后 的执行效果，可不写入实验报告。 2．使用多线程和装饰者模式再次模拟宠物商店 你是宇宙中，最有名的一家 Cyberpunk 宠物商店的老板，而你的雇员全是你 所创造的机器人，他们是——商店总管李、宠物看护者、宠物 Tony 哥、宠物饲 喂猿、宠物服装师、宠物驯化王……，你的顾客来着宇宙的每一个角落。 显然你的宠物店规模和业务量都很大，为了探索更合理的运营方式，你计划 通过软件模拟你的商店，在将来（现在不用做）通过分析商店日志找到优化方案。 商店总管李：负责与客户交流，帮助客户选择相应的服务，并安排其他机器 人执行相应项目。听从你的指令，汇报宠物店在某段时间发生了什么。 宠物看护者：负责宠物的寄养，宠物主人将宠物交给商店，寄养，设定寄养 时间，到时通知认领（与上次不同，这次要实现真正的计时，并且实现真正到时 提醒，可以用程序中的一秒模拟现实世界中的一小时（或十分钟））。 宠物 Tony 哥：负责给宠物的毛发洗、剪、吹、拉、烫、染、焗，这里运用第 2 / 3 页 装饰者模式，实现对宠物毛发的装饰。 宠物饲喂猿：负责给宠物喂食，当宠物饿的时候（过了一段时间就会饿）。 宠物服装师：负责给宠物穿各种各样的服饰，颜色可以是红、橙、黄、绿、 青、蓝、紫（也不一定都要），种类可以是帽、鞋、裤、褂、裙，材质可以是棉、 麻、丝、绒、革；同 Tony 哥一样，负责对宠物衣着装饰，使用装饰者模式。 为了商店收益最大化，每一类的机器人你都创造了很多很多个（总是够用）， 但总管李只有一个，当总管李服务了顾客选择了相应的消费项目，相应的项目立 马转到后台执行，这样总管李又可以马上服务新的客人。每一项项目都有标准的 执行流程，因此每个同类项目（例如，所有宠物染色的时间，戴帽的时间）执行 的时间可以看作是相同的，不同类项目执行时间不同。为了清楚商店的运行情况， 你要求每个机器人把自己的每一步操作写到商店日志里，当然这也是私下进行 的。 你总是很忙，忙着发明新的机器人，没有时间看商店日志，因此，你可以要 求总管李汇报商店在某一段时间发生了哪些事。 要求/提示如下：

(1). 实现真正的计时并提醒，可以通过创建一个线程，在线程中计时，到 时提醒；

(2). 总管李只有一个，可以尝试使用单件模式（不做强制要求，也可以只 实例化一个），总管李其实有点类似以往作业里的 Test 类，他提供一系列的操作 选择，并与用户沟通制定个性化的服务；总管李服务的结果应当是生成一个用户 需求的服务单（可以用字符串描述），相应的其他雇员按照服务单在后台执行;

(3). 顾客消费项目在后台执行也应当用到线程，这样，选完项目，沟通好 之后，开启相应的线程，就可以继续服务下一个顾客；

(4). 在后台执行的线程由于使用屏幕输出，会打断总管李的工作，因此， 应当将每一步的操作写到商店日志中——一个文本文档中，每一条操作记录格式 可以是： 时间 雇员 宠物 操作 宠物描述；

(5). 汇报商店发生了哪些事可以读取文本文件，然后输出到屏幕；

(6). 应当为每一步装饰操作（剪毛、染色、戴帽……）设定一定的时间;

(7). 不同线程在往文本文档写入数据的时候要考虑“同时写”造成的互斥访问问题，如何解决线程的互斥问题；

(8). 程序中需要实现的类较多，可以尝试画画简化 UML 图，梳理梳理关系；

(9). 可以自定义异常类，并在合适的位置使用异常处理机制。

二、问题分析：

题目中需要用到线程来解决，同时我感觉这是比较合适的解决方法，题目中要求真正实现时间管理，可以使用线程中的sleep（）方法来实现对线程一定间隔时间的暂停，从而达到时间管理的目的。而当当前线程不足以处理当下的事件时，创造出一个新的线程来保证时间的继续进行是很有必要的。由于总管李只有一个，因此要求我们将总管李的能力设计的足够全面，具有较好的人机交互能力。另外应当在涉及文件操作的时候设置异常类也是很有必要的，它可以避免我们的程序在运行时卡死。

另外使用文件锁可以使进程独立运行不受干扰，保证在多线程运行时不出错。

三、代码实现：

package 第四次上机作业;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.io.RandomAccessFile;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Calendar;

import java.util.Scanner;

public class PetStore {

static Scanner input = new Scanner(System.in);

public String name;

Manager Lee;

ArrayList<Pet> allPet = new ArrayList<Pet>();

PetStore(){

}

public PetStore(String name){

this.name = name;

this.Lee = Manager.getManager();

this.Lee.setEmployer(this);

try {

RandomAccessFile petStoreLog= new RandomAccessFile( this.name +".txt","rwd");

long sTLength;

sTLength = petStoreLog.length();

petStoreLog.seek(sTLength);

Calendar calendar = Calendar.getInstance();

SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

byte[] use = new String(formatter.format(calendar.getTime())+" "+"宠物商店"+this.name+"创立！！！"+"\n").getBytes("utf-8");

petStoreLog.write(use);

petStoreLog.close();

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

public void serve(Pet pet) {

allPet.add(pet);

pet.storeName = this.name;

Lee.serve(pet);

}

public void manage() {

Lee.report();//总管Lee向“我”汇报商店发生的事件

}

}

package 第四次上机作业;

public class PetFeed extends Thread{

public boolean statu=true;

private LockOutPutTxtFile logTxt;

Pet pet;

PetFeed(){

}

public boolean setBusy() {

this.statu = false;

return this.statu;

}

public boolean setFree() {

this.statu = true;

return this.statu;

}

void feed(Pet pet) {

this.pet = pet;

}

public void run() {

this.setBusy();

logTxt = new LockOutPutTxtFile(pet.storeName+".txt");

logTxt.write(" "+this+" "+this.pet.name+" 处于被喂养状态\n");

while(pet.HP <= pet.MaxHP \* 0.9) {

// 设置每 0.1 秒增加 1 HP

pet.HP++;

try {

logTxt.write(" "+this+" "+this.pet.name+" 处于被喂养状态，HP恢复1\n");

Thread.currentThread().sleep(100);//休眠100ms即0.1秒，使得喂养时Hp恢复有时间递增效应

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

logTxt.write(" "+this+" "+this.pet.name+" 喂养结束\n");

this.setFree();//喂养结束，释放该线程

}

}

package 第四次上机作业;

import java.util.ArrayList;

public class PetCare implements Runnable {

//定义一个布尔型变量，用于决定线程是否结束

public boolean statu=true;

public int time;

public Pet pet;

ArrayList<PetFeed> allPetFeed = new ArrayList<PetFeed>();

private LockOutPutTxtFile logTxt;

public void setPet(Pet pet) {

this.pet = pet;

}

public boolean setBusy() {

this.statu = false;

return this.statu;

}

public boolean setFree() {

this.statu = true;

return this.statu;

}

@Override

public void run() {

// TODO Auto-generated method stub

this.setBusy();

// 进行看护

// 写入商店日志文件 开始进行对 this.pet 的看护，

// 写入数据格式 时间 雇员 宠物 操作 宠物描述

logTxt = new LockOutPutTxtFile(pet.storeName+".txt");

logTxt.write(" "+this+" "+this.pet.name+" 处于被看护状态\n");

while(this.time != 0) {

time--;

try {

//每隔1秒钟，宠物的HP就会减一

Thread.sleep(1000);

this.pet.HP--;

if(this.pet.HP <= this.pet.MaxHP \* 0.3) {

// 写入商店日志文件 this.pet 饥饿，准备进行投喂

// 写入数据格式 时间 雇员 宠物 操作 宠物描述

logTxt.write(" "+this+" "+this.pet.name+" 在看护中，处于饥饿状态，准备投喂食物 \n");

this.feedPet();

}

} catch (InterruptedException e) {

e.printStackTrace();

}

}

// 看护到时，提醒领走宠物

// 在屏幕输出： xx 宠物看护时间到时，提醒领走

// 写入商店日志： xx 宠物看护时间到时，提醒领走

logTxt.write(" "+this+" "+this.pet.name+" 看护结束，不要忘记提醒主人领回宠物哦！\n");

System.out.print(" "+this+" "+this.pet.name+" 看护结束，请领走宠物\n");

//结束线程

this.setFree();

}

private void feedPet() {

// 首先查找空闲的PetFeed

boolean flag = false;

PetFeed choosePF = null;

for(PetFeed pf: allPetFeed) {

if(pf.statu) {

flag = true;

choosePF = pf;

break;

}

}

if(flag) {

}

else {

// 现存的PetFeed都处于忙状态，则新建一个，并保存一下

choosePF = new PetFeed();

allPetFeed.add(choosePF);

}

// 设置喂养的宠物

choosePF.feed(pet);

// 创建线程 进行宠物喂养

Thread petCareT = new Thread(choosePF);

petCareT.start();

}

}

package 第四次上机作业;

import java.util.ArrayList;

import java.util.Scanner;

public class Manager {

static Scanner input = new Scanner(System.in);

private volatile static Manager manager;

private PetStore employer;

ArrayList<PetCare> allPetCare = new ArrayList<PetCare>();

ArrayList<HairDecoratorPattern> allHairDecPet = new ArrayList<HairDecoratorPattern>();

private Manager (){}

public static Manager getManager() {

if (manager == null) {

synchronized (Manager.class) {

if (manager == null) {

manager = new Manager();

//如果没有manager，则创建一个实例对象

}

}

}

return manager;

}

public PetStore getEmployer() {

return employer;

}

public void setEmployer(PetStore employer) {

this.employer = employer;

}

public void serve(Pet pet) {

System.out.print("\n您好！欢迎光临！\n请选择您需要的服务：\n");

int choose;

do{

System.out.print("\n操作菜单：\n1.寄养宠物\n2.给宠物洗澡\n3.给宠物修剪毛发\n4.给宠物毛发染色\n5.给宠物戴个头盔\n6.给宠物穿鞋\n7.退出\n请选择操作：");

choose = input.nextInt();

if(choose == 1) {

this.parkPet(pet);

}

else if(choose == 2 || choose == 3 || choose == 4){

this.PetHairDecorator(pet, choose);

}

else if(choose == 5 || choose == 6) {

this.PetClothesDecorator(pet, choose);

}

else {

break;

}

}while(choose != 7);

System.out.print("\n您已退出该商店！！！\n");

}

private void PetClothesDecorator(Pet pet, int choose) {

// TODO Auto-generated method stub

}

private void PetHairDecorator(Pet pet, int point) {

// TODO Auto-generated method stub

HairDecoratorPattern chooseHDP = chooseHDP();

chooseHDP.setPet(pet);

chooseHDP.setPetHairDecoratorPoint(point);

// 创建线程 毛发装饰

Thread petCleanT = new Thread(chooseHDP);

petCleanT.start();

}

private HairDecoratorPattern chooseHDP() {

boolean flag = false;

HairDecoratorPattern chooseHDP = null;

for(HairDecoratorPattern hdp: allHairDecPet) {

if(hdp.statu) {

flag = true;

chooseHDP = hdp;

break;

}

}

if(flag) {

}

else {

chooseHDP = new HairDecoratorPattern();

allHairDecPet.add(chooseHDP);

}

return chooseHDP;

}

private void parkPet(Pet pet) {

System.out.println("请输入您寄养的时长（输入整数）：");

int time = input.nextInt();

boolean flag = false;

PetCare choosePC = null;

for(PetCare pc: allPetCare) {

if(pc.statu) {

flag = true;

choosePC = pc;

break;

}

}

if(flag) {

}

else {

choosePC = new PetCare();

allPetCare.add(choosePC);

}

choosePC.pet = pet;

choosePC.time = time;

// 创建线程 进行宠物寄养

Thread petCareT = new Thread(choosePC);

petCareT.start();

}

public void report() {

LockOutPutTxtFile logTxt = new LockOutPutTxtFile(this.employer.name+".txt");

System.out.print(logTxt.read(0, 0, 0));

}

}

package 第四次上机作业

public abstract class Pet {

public String description ;

public String name ;

public String storeName;

public int MaxHP;

public int HP;

public Pet(){

}

public Pet(String name, int HP){

this.name = name;

this.HP = HP;

this.MaxHP = HP;

}

public Pet(Pet pet) {

this.name = pet.name;

this.description = pet.description;

this.storeName = pet.storeName;

this.HP = pet.HP;

this.MaxHP = pet.MaxHP;

}

public String getDescription() {

return this.description;

}

public String decoratorDes() {

return null;

}

}

package 第四次上机作业;

public class Dragon extends Pet {

Dragon() {

}

public Dragon(String name,int HP){

super(name,HP);

this.description = this.name+"(龙)";

}

public String getDescription() {

return this.description;

}

}

package 第四次上机作业;

public class Dog extends Pet {

Dog() {

}

public Dog(String name, int HP){

super(name,HP);

this.description = this.name+"是狗狗";

}

public String getDescription() {

return this.description;

}

}

package 第四次上机作业;

public class Cat extends Pet {

Cat() {

}

public Cat(String name, int HP){

super(name,HP);

this.description = this.name+"(è)";

}

public String getDescription() {

return this.description;

}

}

package 第四次上机作业;

public class Bird extends Pet {

Bird() {

}

public Bird(String name,int HP){

super(name,HP);

this.description = this.name+"(鸟)";

}

public String getDescription() {

return this.description;

}

}

package 第四次上机作业;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.io.RandomAccessFile;

import java.nio.ByteBuffer;

import java.nio.channels.FileChannel;

import java.nio.channels.FileLock;

import java.text.SimpleDateFormat;

import java.util.Calendar;

public class LockOutPutTxtFile {

private String filename;

private RandomAccessFile raf;

private FileChannel fc;

FileLock fl;

public LockOutPutTxtFile(String filename){

this.filename = filename;

}

public long write(String text) {

// 使用文件锁 FileLock.lock()进行阻塞式读写文件

try {

raf = new RandomAccessFile(this.filename, "rw");

fc = raf.getChannel();

fl = fc.lock();

if(fl != null) {

long fSize = fc.size();

fc.position(fSize);

Calendar calendar = Calendar.getInstance();

SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

String Log = formatter.format(calendar.getTime())+" "+text;

fc.write(ByteBuffer.wrap(Log.getBytes("utf-8")));

fl.release();

long lateFSize = fc.size();

fc.close();

raf.close();

return lateFSize - fSize;

}

} catch (FileNotFoundException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return -1;

}

public String read(int year, int month, int day) {

// 使用文件锁 FileLock.lock() 进行阻塞式读取文件

String logText = null;

try {

raf = new RandomAccessFile(this.filename, "rw");

fc = raf.getChannel();

fl = fc.lock();

if(fl != null) {

long rafLength = raf.length();

byte[] b = new byte[(int) rafLength];

raf.read(b);

logText = new String(b, "utf-8");

fl.release();

fc.close();

raf.close();

}

} catch (FileNotFoundException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return logText;

}

}

package 第四次上机作业;

import StudentHomeWork.Fourth.Pets.Pet;

public class RedHair extends PetHairDecorator{

public RedHair(Pet pet) {

super(pet);

this.description = this.description + "毛发染红 ";

}

public String decoratorDes() {

return "染毛发";

}

public String getDescription() {

return this.description;

}

}

package 第四次上机作业;

public abstract class PetHairDecorator extends Pet {

protected Pet decoratedHairPet;

public PetHairDecorator() {

}

public PetHairDecorator(Pet pet) {

super(pet);

this.decoratedHairPet = pet;

this.description = this.description + "， 进行了以下美发操作：";

}

public String decoratorDes() {

return null;

}

public String getDescription() {

return this.description;

}

}

package 第四次上机作业;

public class HairDecoratorPattern implements Runnable {

public boolean statu=true;

private LockOutPutTxtFile logTxt;

private Pet pet;

private int petHairDecoratorPoint;

public int getPetHairDecoratorPoint() {

return petHairDecoratorPoint;

}

public void setPetHairDecoratorPoint(int petHairDecoratorPoint) {

this.petHairDecoratorPoint = petHairDecoratorPoint;

}

public Pet getPet() {

return pet;

}

public void setPet(Pet pet) {

this.pet = pet;

}

public boolean setBusy() {

this.statu = false;

return this.statu;

}

public boolean setFree() {

this.statu = true;

return this.statu;

}

private void decorator() {

switch(this.petHairDecoratorPoint) {

case 2:

this.pet = new HairClean(this.pet);

break;

case 3:

this.pet = new HairCut(this.pet);

break;

case 4:

this.pet = new RedHair(this.pet);

break;

}

}

@Override

public void run() {

this.setBusy();

logTxt = new LockOutPutTxtFile(this.pet.storeName+".txt");

this.decorator();

logTxt.write(" "+this+" "+this.pet.name+" "+this.pet.decoratorDes()+" "+this.pet.description+"\n");

try {

Thread.currentThread().sleep(5000);

} catch (InterruptedException e) {

e.printStackTrace();

}

logTxt.write(" "+this+" "+this.pet.name+" "+this.pet.decoratorDes()+"完毕\n");

this.setFree();

}

}

package 第四次上机作业;

public class HairCut extends PetHairDecorator{

public HairCut(Pet pet) {

super(pet);

this.description = this.description + "修剪毛发 ";

}

public String decoratorDes() {

return "剪毛发";

}

public String getDescription() {

return this.description;

}

}

package 第四次上机作业;

public class HairClean extends PetHairDecorator {

public HairClean() {

}

public HairClean(Pet pet) {

super(pet);

this.description = this.description + "清洗毛发 ";

}

public String decoratorDes() {

return "洗澡";

}

public String getDescription() {

return this.description;

}

}

package 第四次上机作业;

public class GoldenHair extends PetHairDecorator{

public GoldenHair(Pet pet) {

super(pet);

this.decoratedHairPet.description.concat("毛发染金 ");

}

public String getDescription() {

return this.decoratedHairPet.description;

}

}

package 第四次上机作业;

public class PetShoes extends PetClothesDecorator{

public PetShoes(Pet pet) {

super(pet);

this.decoratedHairPet.description.concat("穿鞋 ");

}

public String getDescription() {

return this.decoratedHairPet.description;

}

}

package 第四次上机作业;

public abstract class PetClothesDecorator extends Pet {

protected Pet decoratedHairPet;

public PetClothesDecorator(Pet pet) {

this.decoratedHairPet = pet;

decoratedHairPet.description.concat("， 进行了以下穿衣操作：");

}

public String getDescription() {

return this.decoratedHairPet.description;

}

}

package 第四次上机作业;

public class PetCap extends PetClothesDecorator{

public PetCap(Pet pet) {

super(pet);

this.decoratedHairPet.description.concat("戴帽 ");

}

public String getDescription() {

return this.decoratedHairPet.description;

}

}

package 第四次上机作业;

public class GolenClothes extends PetClothesDecorator{

public GolenClothes(Pet pet) {

super(pet);

this.decoratedHairPet.description.concat("金色的 ");

}

public String getDescription() {

return this.decoratedHairPet.description;

}

}

package 第四次上机作业;

public class CottonClothes extends PetClothesDecorator{

public CottonClothes(Pet pet) {

super(pet);

this.decoratedHairPet.description.concat("棉的材质 ");

}

public String getDescription() {

return this.decoratedHairPet.description;

}

}

package 第四次上机作业;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.io.RandomAccessFile;

import java.util.ArrayList;

import java.util.Scanner;

public final class Test {

static Scanner input = new Scanner(System.in);

ArrayList<Pet> allPet = new ArrayList<Pet>();

ArrayList<PetStore> allPetStore = new ArrayList<PetStore>();

public static void main(String[] args) {

Test test = new Test();

int choose;

do{

System.out.print("\n操作菜单：\n1.创建宠物\n2.创建宠物商店\n3.逛宠物商店\n4.打理宠物商店\n5.退出\n请选择操作：");

choose = input.nextInt();

switch(choose) {

case 1:

test.createPet();

break;

case 2:

test.createPetStore();

break;

case 3:

test.shoppingInPetStore();

break;

case 4:

test.managePetStore();

break;

case 5:

System.out.print("\n您已退出！\n");

default:

}

}while(choose != 5);

}

private void createPet() {

int choose;

System.out.print("\n创建宠物：\n1.创建一只狗\n2.创建一只猫\n3.创建一只鸟\n4.创建一条龙\n5.取消\n请选择操作：");

choose = input.nextInt();

System.out.print("请输入宠物名\n");

String name = input.next();

System.out.print("请输入HP\n");

int HP = input.nextInt();

switch(choose) {

case 1:

Dog newDog = new Dog(name, HP);

allPet.add(newDog);

break;

case 2:

Cat newCat = new Cat(name, HP);

allPet.add(newCat);

break;

case 3:

Bird newBird = new Bird(name, HP);

allPet.add(newBird);

break;

case 4:

Dragon newDragon = new Dragon(name, HP);

allPet.add(newDragon);

break;

case 5:

System.out.print("\n您已返回上一层菜单！\n");

default:

}

}

private void createPetStore() {

System.out.println("请输入宠物店名：");

String name = input.next();

PetStore newPetStore = new PetStore(name);

this.allPetStore.add(newPetStore);

try {

RandomAccessFile petStoreLog= new RandomAccessFile( name +".txt","rwd");

long pSLLength = petStoreLog.length();

byte[] b = new byte[(int) pSLLength];

petStoreLog.read(b);

String pSLText = new String(b, "utf-8");

System.out.println(pSLText);

petStoreLog.close();

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

private void shoppingInPetStore() {

System.out.println("逛宠物商店");

PetStore pS = this.allPetStore.get(this.choosePetStore());

pS.serve(this.allPet.get(this.choosePet()));

}

private void managePetStore() {

System.out.println("管理物商店");

PetStore pS = this.allPetStore.get(this.choosePetStore());

pS.manage();

}

private int choosePetStore() {

int indexPS = 1;

for(PetStore pS: this.allPetStore) {

System.out.println(indexPS+"."+pS.name);

indexPS++;

}

System.out.println("请输入相应序号选择宠物商店：");

return input.nextInt() - 1;

}

private int choosePet() {

int indexP = 1;

for(Pet p: this.allPet) {

System.out.println(indexP+"."+p.name);

indexP++;

}

System.out.println("请输入相应序号选择宠物：");

return input.nextInt() - 1;

}

}

四、运行结果：

Debug没过。。。。。。

五、遇到的问题：

总体来说，这次遇到的问题特别多，尤其是线程的使用方面，虽然线程的原理很好理解，但是真正较好地运用线程解决问题却没那么简单（个人而言）。所以这次作业的话，我个人感觉最大的难点还是线程。

六、总结：

通过这次作业，好好练习了一下线程的使用，线程的两种创建方法也都在代码中有所体现，学习到了一些run（）方法的设计方式，另外，对文件锁的使用熟练了一些，更重要的是，对类的操作更加熟练了。