**第七章作业**

（共100分）

一、单选题，并说明理由（对的为什么对，错的为什么错，每个选项都要评论）（共10分，每题5分；其中选对2分，解释3分）

1. 要实现线程，可以从以下（ B ）类继承。

A. Runnable B. Thread C. Throwable D. Stream

是实现该接口 这是异常类 通常指的是Java流式处理（Stream API）

2、 关于线程设计，下列描述正确的是（ ）

A、线程对象必须实现Runnable接口 也可以继承 Thread 类

B、启动一个线程直接调用线程对象的run()方法 应该使用 start() 方法

C、Java提供对多线程同步提供语言级的支持

D、一个线程可以包含多个进程 说反了

二、程序阅读题，写出程序的输出结果，并进行解释（每题40分，共 40 分，其中运行结果20分，解释20分）

1、

public class MulTr1 implements Runnable {

public void run() {

synchronized (this) {

try {

for (int i = 0; i < 3; i++) {

System.out.println(i);

Thread.sleep(100);

if (i == 1) {

throw new InterruptedException("打断了");

}

}

}

catch (InterruptedException e) {

System.out.println(e.getMessage());

}

finally {

System.out.println("in finally");

}

}

}

public static void main(String[] args) {

MulTr1 m1 = new MulTr1();

Thread t1 = new Thread(m1);

Thread t2 = new Thread(m1);

t1.start();

t2.start();

}

}

0

1

打断了

in finally

0

1

打断了

in finally

解释:

在 run 方法中，有一个 synchronized (this) 块，它锁定了当前对象（this），这意味着在同一时刻只有一个线程可以执行此块中的代码。

在 for 循环中，它打印从0到2的数字，并在每次迭代之后休眠100毫秒，以模拟一些计算或操作。

在 catch 块中，它捕获了 InterruptedException 异常，并打印异常的消息，这是在异常对象

在 main 方法中，创建了两个线程 t1 和 t2，它们都使用相同的 MulTr1 实例 m1 作为运行目标。

t1先获得同步锁,进入for循环,打印0,1后抛出InterruptedException("打断了"),被catch捕获,打印”打断了”,执行finally块打印in finally,然后t1释放同步锁,t2获得同步锁开始执行,与t1执行过程相同.

三、阅读和编写程序题（50分，共1题）

1、阅读下面的程序，写出运行结果（如果不止一种，写出一种即可）；并按要求改写程序。（其中阅读程序部分20分，改写程序部分30分）。提示：本程序模拟两个Dota小兵（Creep）攻击一个玩家的场景。

class Player {

private int hp = 100; //体力

public int getHP() {

return hp;

}

public void setHP(int hp) {

this.hp = hp;

}

}

public class Creep implements Runnable {

private Player player = new Player();

@Override

public void run() {

synchronized (this) {

for (int i = 0; i < 3; i++) {

System.out.println(Thread.currentThread().getName() + " attack...");

this.attack(20);

System.out.println(Thread.currentThread().getName() + ": 当前player的hp值= " + player.getHP());

if (player.getHP() <= 0) {

System.out.println(Thread.currentThread().getName() + ": player is dead.");

break;

}

try {

Thread.sleep(100);

}

catch (InterruptedException e) {

}

}

System.out.println(Thread.currentThread().getName() + " end.");

}

}

public void attack(int y) { //攻击掉血

player.setHP(player.getHP() - y);

}

public static void main(String[] args) {

Creep r = new Creep();

Thread ca = new Thread(r, "Creep-A");

Thread cb = new Thread(r, "\t\t\tCreep-B");

ca.start();

cb.start();

}

}

1. 写出运行结果，并进行解释，如果不止一种结果，写出一种即可（这部分20分，其中结果10分，解释10分）

Creep-A: 当前player的hp值= 80

Creep-A attack...

Creep-A: 当前player的hp值= 60

Creep-A attack...

Creep-A: 当前player的hp值= 40

Creep-A end.

Creep-B attack...

Creep-B: 当前player的hp值= 20

Creep-B attack...

Creep-B: 当前player的hp值= 0

Creep-B: player is dead.

Creep-B end.

解释:

在 Creep 的 run 方法中，它使用 synchronized 关键字锁定了当前对象（this），以确保多个怪物不会同时攻击同一个玩家。

在 run 方法中，每个怪物执行一个循循环，每次循环中，它打印当前线程的名称，然后调用 attack 方法对玩家造成伤害。然后，它打印当前玩家的 hp 值，并检查玩家的 hp 是否小于等于0，如果是，表示玩家死亡，就会结束战斗。

在每次攻击后，怪物会休眠100毫秒，以模拟攻击的时间间隔。

在 main 方法中，创建了一个 Creep 对象，并分别创建了两个线程 ca 和 cb 来运行这两个怪物。然后启动这两个线程。一开始ca获得同步锁,攻击三次,每次血量减少20,并打印目前血量,最后end,然后释放同步锁,cb获得同步锁开始执行,攻击两次后血量变为0,打印dead,执行break跳出循环,最后打印end.

(2)改写程序。上述程序采用的是同步块的方式，现改为采用同步方法的方式。要求达到和上述程序同样的输出结果（不考虑结果随机出现）。要求（这部分30分）：

（a）移除Creep类里的attack方法；移除run方法中的同步块。

（b）在Player类里新建一个同步方法：synchronized void beAttacked(int y)，模拟creep三次攻击的行为，并安照同样的内容输出。如果玩家血降到0后，该线程提前终止。

（c）main方法同之前的设置，创建两个线程。

class Player {

    private int hp = 100; //体力

    public int getHP() {

      return hp;

    }

    public void setHP(int hp) {

      this.hp = hp;

    }

    synchronized void beAttacked(int y){

        for (int i = 0; i < 3; i++) {

            System.out.println(Thread.currentThread().getName() + " attack...");

            this.setHP(this.getHP() - y);

            System.out.println(Thread.currentThread().getName() + ": 当前player的hp值= " + this.getHP());

            if (this.getHP() <= 0) {

              System.out.println(Thread.currentThread().getName() + ": player is dead.");

              break;

            }

          try {

            Thread.sleep(100);

          }

          catch (InterruptedException e) {

          }

        }

        System.out.println(Thread.currentThread().getName() + " end.");

  }

}

 class Creep implements Runnable {

    private Player player = new Player();

    @Override

    public void run() {

        player.beAttacked(20);

    }

    public static void main(String[] args) {

      Creep r = new Creep();

      Thread ca = new Thread(r, "Creep-A");

      Thread cb = new Thread(r, "\t\t\tCreep-B");

      ca.start();

      cb.start();

    }

  }

---------------------------------------------

提交方式：

1. 上传到[坚果云链接](https://send2me.cn/UhZVqQpC/Swut7qAZCQb8yw)中（或扫描下面的二维码）：
2. 文件名严格按照：学号-姓名-第几次作业.docx(或pdf)的方式命名。
3. 每次作业在截止日期之前，若发现提交内容有误，可以再提交一次。但每次作业最多只能提交两次。第二次提交的文件后面加上“（2）”的字样。
4. 作业提交形式：单一的word或pdf文档。不要上传Eclipse、VScode、IDEA等项目文件，不要压缩打包，不要以文件夹形式上传。

4. 截止时间：10月13日晚23:59