# 遗物

## 作用时机

（黄色部分是不认识，或者比较难用）

**public** **void** onPlayCard(AbstractCard c, AbstractMonster m) {}

**public** **void** onObtainCard(AbstractCard c) {}

**public** **void** onGainGold() {}

**public** **void** onLoseGold() {}

**public** **void** onEquip() {}

**public** **void** onUnequip() {}

**public** **void** atPreBattle() {}

**public** **void** atBattleStart() {}

**public** **void** atBattleStartPreDraw() {}

**public** **void** atTurnStart() {}

**public** **void** onPlayerEndTurn() {}

**public** **void** onBloodied() {}

**public** **void** onNotBloodied() {}

**public** **void** onManualDiscard() {}

**public** **void** onUseCard(AbstractCard targetCard, UseCardAction useCardAction) {}

**public** **void** onVictory() {}

**public** **void** onMonsterDeath(AbstractMonster m) {}

**public** **int** onPlayerGainBlock(**int** blockAmount)

{

**return** blockAmount;

}

**public** **int** onPlayerHeal(**int** healAmount)

{

**return** healAmount;

}

**public** **void** onMeditate() {}

**public** **void** onEnergyRecharge() {}

**public** **void** onRest() {}

**public** **void** onRitual() {}

**public** **void** onEnterRestRoom() {}

**public** **void** onRefreshHand() {}

**public** **void** onShuffle() {}

**public** **void** onSmith() {}

**public** **void** onAttack(DamageInfo info, **int** damageAmount, AbstractCreature target) {}

**public** **int** onAttacked(DamageInfo info, **int** damageAmount)

{

**return** damageAmount;

}

**public** **void** onExhaust(AbstractCard card) {}

**public** **void** onTrigger() {}

**public** **void** onTrigger(AbstractCreature target) {}

**public** **boolean** checkTrigger()

{

**return** **false**;

}

**public** **void** onEnterRoom(AbstractRoom room) {}

**public** **void** onCardDraw(AbstractCard drawnCard) {}

**public** **void** onChestOpen(**boolean** bossChest) {}

**public** **void** onDrawOrDiscard() {}

**public** **void** onMasterDeckChange() {}

**public** **boolean** canPlay(AbstractCard card)

{

**return** **true**;

}

**public** **void** onUsePotion() {}

**public** **void** onLoseHp(**int** damageAmount) {}

## 描述

**public** String getUpdatedDescription()

{

**return** **this**.DESCRIPTIONS[0];

}

基础的遗物描述就一段话**this**.DESCRIPTIONS[0]

需要更新描述的：

**this**.description = (**this**.DESCRIPTIONS[0] + **this**.DESCRIPTIONS[1] + **this**.card1.name + **this**.DESCRIPTIONS[2]);

**this**.tips.clear();

**this**.tips.add(**new** PowerTip(**this**.name, **this**.description));

initializeTips();

第一句改遗物描述

第二句清除原有描述和遗物名字

第三句加上现有遗物名字和描述即加上描述-第一句

第四句不知道，一起用吧

需要用**import** com.megacrit.cardcrawl.helpers.PowerTip;

## 遗物闪烁

AbstractDungeon.actionManager.addToBottom(**new** RelicAboveCreatureAction(AbstractDungeon.player, **this**));

让遗物在角色这边闪一下AbstractDungeon.player可被替换

**this** 指的是这个遗物可以让其他遗物闪（不过没什么必要，一般都是**this**）

需要用

flash();

**this**.flash();

小遗物图标闪一下（两句都行）

## 计数板

**this**.counter = 0;这个用来显示计数比如钢笔尖阳光花，显示出在遗物图标上

# 卡牌

c.name 返回卡牌的名称

c.name == "打击" 返回 **ture** or **false**

c.name.contains("打击")

c.type 返回种类

c.type == AbstractCard.CardType.SKILL 返回 **ture** or **false**

ATTACK, SKILL, POWER, STATUS, CURSE;

c.rarity 返回稀有度

c.rarity == AbstractCard.CardRarity.BASIC 返回 **ture** or **false**

BASIC, SPECIAL, COMMON, UNCOMMON, RARE, CURSE;

c.target 返回卡牌目标

c.target == AbstractCard.CardTarget.NONE

ENEMY, ALL\_ENEMY, SELF, NONE, SELF\_AND\_ENEMY, ALL;

c.color 返回卡牌颜色

c.color == AbstractCard.CardColor.COLORLESS

RED, GREEN, BLUE, COLORLESS, CURSE;

c.cost

c.costForTurn

c.energyOnUse

c.exhaust = **true**; 将c消耗

c.upgrade(); 将c升级

c.canUpgrade() 返回 **ture** or **false**

c.rawDescription +="消失";

c.initializeDescription();

这两句不知道是不是一起的，用来改卡牌的描述

AbstractDungeon.effectList.add(**new** ShowCardAndObtainEffect(**new** Bite(), Settings.WIDTH / 2.0F, Settings.HEIGHT / 2.0F));

牌组添加一张撕咬，需要以下两个

**import** com.megacrit.cardcrawl.vfx.cardManip.ShowCardAndObtainEffect;

**import** com.megacrit.cardcrawl.core.Settings;

# Power（能力）

这个的制作比遗物难一点建议从遗物开始做，下面可以跳过作用时机

## 作用时机

**public** **float** atDamageGive(**float** damage, DamageInfo.DamageType type)

{

**return** damage;

}

**public** **float** atDamageFinalGive(**float** damage, DamageInfo.DamageType type)

{

**return** damage;

}

**public** **float** atDamageFinalReceive(**float** damage, DamageInfo.DamageType type)

{

**return** damage;

}

**public** **float** atDamageReceive(**float** damage, DamageInfo.DamageType damageType)

{

**return** damage;

}

**public** **void** atStartOfTurn() {}

**public** **void** atStartOfTurnPostDraw() {}

**public** **void** atEndOfTurn(**boolean** isPlayer) {}

**public** **void** atEndOfRound() {}

**public** **void** onDamageAllEnemies(**int**[] damage) {}

**public** **int** onHeal(**int** healAmount)

{

**return** healAmount;

}

**public** **int** onAttacked(DamageInfo info, **int** damageAmount)

{

**return** damageAmount;

}

**public** **void** onAttack(DamageInfo info, **int** damageAmount, AbstractCreature target) {}

**public** **void** onPlayCard(AbstractCard card, AbstractMonster m) {}

**public** **void** onUseCard(AbstractCard card, UseCardAction action) {}

**public** **void** onAfterUseCard(AbstractCard card, UseCardAction action) {}

**public** **void** onSpecificTrigger() {}

**public** **void** onDeath() {}

**public** **void** atEnergyGain() {}

**public** **void** onExhaust(AbstractCard card) {}

**public** **int** modifyBlock(**int** blockAmount)

{

**return** blockAmount;

}

**public** **void** onGainedBlock(**int** blockAmount) {}

**public** **int** onPlayerGainedBlock(**int** blockAmount)

{

**return** blockAmount;

}

**public** **void** onRemove() {}

**public** **void** onEnergyRecharge() {}

**public** **void** onDrawOrDiscard() {}

**public** **void** onAfterCardPlayed(AbstractCard usedCard) {}

**public** **void** onInitialApplication() {}

**public** **void** onApplyPower(AbstractPower power, AbstractCreature target, AbstractCreature source) {}

**public** **void** onLoseHp(**int** damageAmount) {}

}

## 基础

p.type == AbstractPower.PowerType.BUFF

p.type == AbstractPower.PowerType.DEBUFF

p.amount p的层数如果没有p；返回**null**

p.ID

p.ID.equals("Weakened"**)**

p.ID.equals("Vulnerable")

**if** (m.hasPower("Vulnerable")) {

**int** n = m.getPower("Vulnerable").amount;

}

如果m有易伤，定义局部变量整数n=易伤的层数，这个语句很有用

**int** i = p.amount;

## 给予power

用以下语句来给予power

AbstractDungeon.actionManager.addToTop(**new** ApplyPowerAction(AbstractDungeon.player, AbstractDungeon.player, **new** XXXPower(AbstractDungeon.player, **N**), **N**));

addToTop 或 addToBottom

第一个：AbstractDungeon.player 被给予者

第二个：AbstractDungeon.player 给予者

第三个：AbstractDungeon.player 给予者

XXXPower：power的名称

N:给予的层数，一般来说前后两个N一样

还可以这样

AbstractDungeon.player.addPower(**new** StrengthPower(AbstractDungeon.player, 1));

AbstractDungeon.player.addPower(**new** DexterityPower(AbstractDungeon.player, 1));

## 解除power

AbstractDungeon.actionManager.addToBottom(**new** RemoveSpecificPowerAction(m, m, "shang"));

解除指定buff——shang

AbstractDungeon.actionManager.addToBottom(**new** RemoveDebuffsAction(AbstractDungeon.player));

解除所有debuff

也可以这样，下面就要你看懂，其实也很容易看懂

**for** (AbstractPower p : AbstractDungeon.player.powers) {

**if** (p.type == AbstractPower.PowerType.DEBUFF) {

AbstractDungeon.actionManager.addToTop(**new** RemoveSpecificPowerAction(AbstractDungeon.player, AbstractDungeon.player, p.ID));

}

}

AbstractDungeon.actionManager.addToBottom(**new** ReducePowerAction(m, m, "yinchang", 1));

减少吟唱power1层

# 定义变量

## 全局定义变量

**private** **boolean** a = **false**;

**private** **static** **final** **int** b = 25;

**private** **int** c;

**private** AbstractCard card;

**private** AbstractPower power;

**private** AbstractPlayer p;

基本上就是定义卡牌和整数和正误

全局定义一定要用到**this**.XXX

比如**this**.fsn = 0;**this**.card1 = **new** Wound();

用**this**.XXX后，全局都能使用

## 局部定义变量

定义仅存在一个{}里，仅仅只能在{}里。

**boolean** a = **false**;

**int** c;

AbstractCard card;

AbstractPower power;

AbstractPlayer p;

**int** i = MathUtils.random(1, 10);

# For语句

**for** (AbstractMonster m : AbstractDungeon.getCurrRoom().monsters.monsters){}

对于地图上所有怪物（不管死没死或者半死不活）都做一下运行{}里的

**for** (AbstractCard c : AbstractDungeon.player.drawPile.group) {}

对于角色的摸牌区的每张牌都做一下运行{}里的

下面还有各种区

masterDeck 牌组（这是固定的，不随战斗中增加牌等变化）

drawPile 摸牌区

hand 手牌区

discardPile 弃牌区

exhaustPile 消耗区

**for** (AbstractPower p : AbstractDungeon.player.powers) {}

对于角色的每个power都做一下{}里的

# 地图

AbstractDungeon.getCurrRoom() **instanceof** MonsterRoomElite 当前地图是否符合精英怪房

AbstractDungeon.getCurrRoom() **instanceof** MonsterRoomBoss 当前地图是否符合首领怪房

AbstractDungeon.getCurrRoom().phase == AbstractRoom.RoomPhase.COMBAT 当前地图是否在战斗中

# 怪物

m.isDead 返回 **ture** or **false**

m.isDying 返回 **ture** or **false**

m.hasPower("zhuoshao") 返回 **ture** or **false**

# 职业

AbstractDungeon.player.masterHandSize 抽牌数（参见背包）

AbstractDungeon.player.energy.energyMaster 能量数（参见添水）

AbstractDungeon.player.hand.size() 返回当前手牌数

此处hand依然可以换

masterDeck 牌组（这是固定的，不随战斗中增加牌等变化）

drawPile 摸牌区

hand 手牌区

discardPile 弃牌区

exhaustPile 消耗区

# 其他

m.currentHealth m的当前生命

m.maxHealth m的生命上限

MathUtils.round(m.currentHealth \* 0.03F);

m的当前生命的3% 一般小数转化整数要用到MathUtils.round

AbstractDungeon.player.increaseMaxHp(5, **true**);

主角增加5生命上限，**true**表示会产生效应（显示生命上限+5），**false**则没有

m.decreaseMaxHealth(5)

m减少生命上限

# 动作

AbstractDungeon.actionManager.addToTop(**new** GainEnergyAction(1));

主角获得1能量

AbstractDungeon.actionManager.addToBottom(**new** DrawCardAction(AbstractDungeon.player, 1));

主角摸1张牌

AbstractDungeon.actionManager.addToBottom(**new** DiscardAction(AbstractDungeon.player, AbstractDungeon.player, 1, **false**));

弃**1**张牌，**false** 指是否随机弃牌

AbstractDungeon.player.heal(1);

主角回复1生命

AbstractDungeon.actionManager.addToTop(**new** GainBlockAction(AbstractDungeon.player, AbstractDungeon.player, 1));

主角得到1格挡

# 伤害

## 种类

d.type == DamageInfo.DamageType.NORMAL

damage.type == DamageInfo.DamageType.THORNS

Damage.type == DamageInfo.DamageType.HP\_LOSS

## 固定伤害

AbstractDungeon.actionManager.addToBottom(**new** DamageAllEnemiesAction(**null**,

DamageInfo.createDamageMatrix(3, **true**), DamageInfo.DamageType.THORNS, AbstractGameAction.AttackEffect.BLUNT\_LIGHT));

水银沙漏，开心刀的全体伤害（固定伤害）

AbstractGameAction.AttackEffect.BLUNT\_LIGHT中的BLUNT\_LIGHT可被替换：

BLUNT\_LIGHT, BLUNT\_HEAVY, SLASH\_DIAGONAL, SMASH, SLASH\_HEAVY, SLASH\_HORIZONTAL, SLASH\_VERTICAL, NONE, FIRE, POISON, SHIELD;

AbstractDungeon.actionManager.addToBottom(**new** DamageRandomEnemyAction(**new** DamageInfo(AbstractDungeon.player, 3, DamageInfo.DamageType.THORNS), AbstractGameAction.AttackEffect.FIRE));

铜拔的随机伤害（固定伤害）

充能佛珠的伤害

AbstractDungeon.actionManager.addToTop(**new** DamageAction(m, **new** DamageInfo(AbstractDungeon.player, (m.maxHealth - m.currentHealth) / 10, DamageInfo.DamageType.THORNS), AbstractGameAction.AttackEffect.BLUNT\_LIGHT));

AbstractDungeon.actionManager

.addToBottom(**new** LoseHPAction(AbstractDungeon.player, AbstractDungeon.player, 5));

}

主角失去5点体力

AbstractDungeon.actionManager.addToTop(**new** LoseHPAction(m, m, 6));

m失去6点体力

## 受力量等加成伤害

# 程序包

缺少什么包就添加以下（全部加入也行）

**import** com.megacrit.cardcrawl.cards.DamageInfo;

**import** com.megacrit.cardcrawl.cards.DamageInfo.DamageType;

**import** com.megacrit.cardcrawl.actions.common.\*;

**import** com.megacrit.cardcrawl.powers.\*;

**import** com.megacrit.cardcrawl.actions.unique.\*;

# 药水 奖励界面

AbstractDungeon.getCurrRoom().addPotionToRewards(PotionHelper.getRandomPotion(AbstractDungeon.miscRng));

上面那个地方可被替换

addRelicToRewards(AbstractRelic relic)

addPotionToRewards(AbstractPotion potion)

addCardToRewards()

AbstractRelic relic，AbstractPotion potion需要被替换成遗物或药水id

如星系仪

**public** **void** onEquip() //当你装备时//

{

**for** (**int** i = 0; i < 4; i++) { //做个循环5次，奖励界面最多5个，可以更多，不过显示起来不好看//

AbstractDungeon.getCurrRoom().addCardToRewards(); //在奖励界面添加卡牌奖励（三选一的那个）//

}

AbstractDungeon.combatRewardScreen.open(**this**.DESCRIPTIONS[1]); //奖励界面上方出现文字**this**.DESCRIPTIONS[1]//

AbstractDungeon.getCurrRoom().rewardPopOutTimer = 0.0F; //这个不清楚//

}

# 金钱

AbstractDungeon.getCurrRoom().addGoldToRewards(30**)**;

在奖励界面出现30块钱

AbstractDungeon.player.gainGold(**50**);

直接获得50块

AbstractDungeon.effectList.add(**new** RainingGoldEffect(**50**));

添加钱的效应，其实就是动画

# 注意

m c p 等不是一成不变

但是 AbstractDungeon.player 就是指角色，不会变

**private** AbstractPlayer m;这样子**this**.m也是角色了

# 构造函数（难）

第一个是构造更新描述的函数

**public** **void** updateDescription()

{

**this**.description = (**this**.DESCRIPTIONS[0] + num + **this**.DESCRIPTIONS[1] + num \* (num + 1) / 2 + **this**.DESCRIPTIONS[2]);

**this**.tips.clear();

**this**.tips.add(**new** PowerTip(**this**.name, **this**.description));

initializeTips();

}

构造好后，用updateDescription();即可调用更新函数

第二个是计算power的amount（层数）

**private** **int** GetPowerCount(AbstractCreature m, String powerId)

{

AbstractPower power = m.getPower(powerId);

**return** power != **null** ? power.amount : 0; //如果没有这个power原本是返回层数null，现在是0//

}

以后用GetPowerCount(AbstractDungeon.player, "yinchang")

返回吟唱power的层数，没有就返回0

第三个是。。。

# 实例

## 遗物

这里就做一个吟唱之玲

**package** yiwuMod.relics; //这里是打包在yiwuMod文件夹里的relics文件夹里//

**import** basemod.abstracts.CustomRelic;

**import** com.badlogic.gdx.Files;

**import** com.badlogic.gdx.Gdx;

**import** com.badlogic.gdx.graphics.Texture;

**import** com.megacrit.cardcrawl.actions.GameActionManager;

**import** com.megacrit.cardcrawl.actions.animations.VFXAction;

**import** com.megacrit.cardcrawl.actions.common.ApplyPowerAction;

**import** com.megacrit.cardcrawl.characters.AbstractPlayer;

**import** com.megacrit.cardcrawl.core.EnergyManager;

**import** com.megacrit.cardcrawl.core.Settings;

**import** com.megacrit.cardcrawl.dungeons.AbstractDungeon;

**import** com.megacrit.cardcrawl.helpers.Hitbox;

**import** com.megacrit.cardcrawl.powers.yinchangPower;

**import** com.megacrit.cardcrawl.relics.AbstractRelic;

**import** com.megacrit.cardcrawl.relics.AbstractRelic.LandingSound;

**import** com.megacrit.cardcrawl.relics.AbstractRelic.RelicTier;

**import** com.megacrit.cardcrawl.vfx.combat.ShockWaveEffect;

**import** com.megacrit.cardcrawl.vfx.combat.ShockWaveEffect.ShockWaveType;

//上面是导入各种包//

**public** **class** yinchang //这里是制作yichang.class文件注意一定要与文件名一样//

**extends** CustomRelic //这里固定格式，mod拓展遗物//

{

**public** **static** **final** String ID = "yinchang"; //遗物id，必须有//

**private** **static** **final** String IMG = "img/relics/yinchang.png"; //遗物图片，必须有//

**private** **static** **final** String OUTLINE = "img/relics/outline/yinchang.png"; //遗物轮廓图，必须有//

**private** **boolean** firstTurn = **true**; //这里声明全局正误变量 firstTurn，好像后面没用到。。。。//

**private** **int** zu1 = 1; //这里定义第一个全局整数变量zu1=1//

**private** **int** zu2 = 10; //这里定义第二个全局整数变量zu2=10//

**public** yinchang() //开始yinchang的代码了，必须有//

{

**super**("yinchang", **new** Texture(Gdx.files.internal("img/relics/yinchang.png")), **new** Texture(Gdx.files.internal("img/relics/outline/yinchang.png")), AbstractRelic.RelicTier.BOSS, AbstractRelic.LandingSound.HEAVY);

} //一些参数，必需有//

**public** **void** atPreBattle() //触发时机：在准备战斗开始时//

{

**this**.zu1 = 1; //初始zu1 = 1//

**this**.zu2 = (3 \* AbstractDungeon.player.energy.energyMaster);

} //zu3 = 3倍能量，角色初始每回合3能量，这时就是9，以后会增加//

**public** **void** atTurnStart() //在回合开始时//

{

**if** (!AbstractDungeon.player.hasPower("yinchang")) //判断：如果主角没有吟唱power//

{

AbstractDungeon.actionManager.addToBottom(**new** ApplyPowerAction(AbstractDungeon.player, AbstractDungeon.player, **new** yinchangPower(AbstractDungeon.player, **this**.zu1, **this**.zu2, **false**), **this**.zu1, **true**));

//给予主角吟唱power **this**.zu1是几回合，**this**.zu2是伤害，刚开始是1回合后对所有敌人造成9点伤害//

AbstractDungeon.actionManager.addToBottom(**new** VFXAction(AbstractDungeon.player, **new** ShockWaveEffect(AbstractDungeon.player.hb.cX, AbstractDungeon.player.hb.cY, Settings.RED\_TEXT\_COLOR, ShockWaveEffect.ShockWaveType.ADDITIVE), 0.5F));

//这个是个动画，复制尖啸来的//

**this**.zu1 += 1;

**this**.zu2 = (**this**.zu1 \* (**this**.zu1 + 1) / 2 \* 3 \* AbstractDungeon.player.energy.energyMaster);

//给予吟唱power后，回合增加1，伤害增加？？//

}

}

**public** String getUpdatedDescription()

{

**return** **this**.DESCRIPTIONS[0];

}

//遗物的描述，这里遗物描述是在josn文件里面，也可以在class这里直接写名称和描述//

**public** AbstractRelic makeCopy()

{

**return** **new** yinchang();

}

//结束语返回新的吟唱遗物，这里必须有//

}

## Power

这里继续吟唱power

**package** com.megacrit.cardcrawl.powers;

//打包power在那个文件夹里//

**import** com.megacrit.cardcrawl.actions.AbstractGameAction.AttackEffect;

**import** com.megacrit.cardcrawl.actions.GameActionManager;

**import** com.megacrit.cardcrawl.actions.common.DamageAction;

**import** com.megacrit.cardcrawl.actions.common.ReducePowerAction;

**import** com.megacrit.cardcrawl.actions.common.RemoveSpecificPowerAction;

**import** com.megacrit.cardcrawl.cards.DamageInfo;

**import** com.megacrit.cardcrawl.cards.DamageInfo.DamageType;

**import** com.megacrit.cardcrawl.core.AbstractCreature;

**import** com.megacrit.cardcrawl.core.CardCrawlGame;

**import** com.megacrit.cardcrawl.dungeons.AbstractDungeon;

**import** com.megacrit.cardcrawl.helpers.ImageMaster;

**import** com.megacrit.cardcrawl.localization.LocalizedStrings;

**import** com.megacrit.cardcrawl.localization.PowerStrings;

**import** com.megacrit.cardcrawl.monsters.AbstractMonster;

**import** com.megacrit.cardcrawl.monsters.MonsterGroup;

**import** com.megacrit.cardcrawl.rooms.AbstractRoom;

**import** com.megacrit.cardcrawl.rooms.AbstractRoom.RoomPhase;

//上面就不说了//

**public** **class** yinchangPower //同样做个yinchangPower.class//

**extends** AbstractPower //拓展power//

{

**public** **static** **final** String POWER\_ID = "yinchang";

**private** **static** **final** PowerStrings powerStrings = CardCrawlGame.languagePack.getPowerStrings("yinchang");

**public** **static** **final** String NAME = powerStrings.NAME;

**public** **static** **final** String[] DESCRIPTIONS = powerStrings.DESCRIPTIONS;

//上面四个必要有，注意这里power的描述是在json文件里//

**private** **boolean** justApplied = **false**;

**private** **static** **final** **int** EFFECTIVENESS\_STRING = 25;

**private** **int** ycDamgeAmount;

//上面定义全局变量//

**public** yinchangPower(AbstractCreature paramAbstractCreature, **int** paramInt1, **int** paramInt2, **boolean** paramBoolean)

{

**this**.name = NAME;

**this**.ID = "yinchang";

**this**.owner = paramAbstractCreature;

**this**.amount = paramInt1;

**this**.ycDamgeAmount = paramInt2;

updateDescription();

**this**.img = ImageMaster.loadImage("images/powers/32/knowledge.png");

**if** (paramBoolean) {

**this**.justApplied = **true**;

}

**this**.isTurnBased = **true**;

**this**.priority = 99;

}

**public** **void** atEndOfRound()

{

**if** (**this**.justApplied)

{

**this**.justApplied = **false**;

**return**;

}

}

**public** **void** atStartOfTurn() //回合开始时//

{

**if** (**this**.amount == 1) //如果吟唱层数等于1//

{

**if** (AbstractDungeon.getCurrRoom().phase == AbstractRoom.RoomPhase.COMBAT) {

**for** (AbstractMonster localAbstractMonster : AbstractDungeon.getCurrRoom().monsters.monsters)

{

AbstractDungeon.actionManager.addToTop(**new** DamageAction(localAbstractMonster, **new** DamageInfo(**this**.owner, **this**.ycDamgeAmount, DamageInfo.DamageType.THORNS), AbstractGameAction.AttackEffect.BLUNT\_LIGHT));

AbstractDungeon.actionManager.addToBottom(**new** RemoveSpecificPowerAction(**this**.owner, **this**.owner, "yinchang"));

//if战斗中，for对每个怪物造成伤害，remove吟唱power//

}

}

}

**else** {

AbstractDungeon.actionManager.addToBottom(**new** ReducePowerAction(**this**.owner, **this**.owner, "yinchang", 1));

}

//否则吟唱层数减1//

}

**public** **void** updateDescription() //用来更新描述，这里是构造一个函数updateDescription(); //

{

**this**.description = (DESCRIPTIONS[0] + **this**.amount + DESCRIPTIONS[1] + **this**.ycDamgeAmount + DESCRIPTIONS[2]);

}

}