回合制游戏编码过程

实现Utils.getValue

```java
public double getValue(double[] values) {
    // TO DO
    Random random = new Random();
    double val = random.nextDouble();

    // 10,100
    int min = 10;
    int max = 100;
    double res = val * (100 - 10 + 1) + 10;
    return Math.floor(res);
}
```

Utils.isDo

```java
/**
 * 判断技能是否释放
 *
 * @param x : 技能触发的概率
 * @return
 */
public boolean isDo(double x) {
    // 生成数组
    int[] arr = new int[10];
    int len = (int) (x * 10);
    for (int i = 0; i < len; i++) {
        arr[i] = 1;
    }
    Random random = new Random();
    int index = random.nextInt(10);
    return arr[index] == 1;
}
```

Fighting.baseAttack

```java
 double baseAttack(int sed) {
        Utils utils = new Utils();
        // a->b
        Player a = null;
        Player b = null;
        // 判定攻击者和防御者
        if (sed == 0) {
            a = playerA;
            b = playerB;
        } else {
            a = playerB;
            b = playerA;
        }
        // A 玩家的攻击 - B 玩家的防御
        double attack = utils.getValue(a.attack); // 得到A玩家的攻击
        double defense = utils.getValue(b.defense); // 得到B玩家的防御
//        System.out.println(attack + "\t" + defense);
        return attack - defense;
}
```

Fighting.killAttack

```
/**
 * a=攻击
 * b=防御
 * kill = a 使用的技能
 *
 * @return 技能攻击的伤害, 未触发技能伤害为0
 */
double killAttack(Player a, Player b, Kill kill) {
    Utils utils = new Utils();
    // A->B
    // 判断A是否发动技能,如果不发动则返回 0
    boolean isDo = utils.isDo(kill.probability);
    if (!isDo) {
        return 0.0;
    }
    System.out.println("使用了技能" + kill.name);

    // 获取A技能的伤害值a
    double baseAttack = utils.getValue(a.attack);
    // 1. 技能伤害的%*基础伤害 得到新的伤害区间
    double[] harmsNew = new double[2];
    harmsNew[0] = baseAttack * kill.harms[0]; // 1.3 - 1.5
    harmsNew[1] = baseAttack * kill.harms[1]; // 1.5
    // 2. 由新的伤害区间得到具体的伤害数据
    double killAttack = utils.getValue(harmsNew);
    // 获取B的防御值
    double defense = utils.getValue(b.defense);
    // 是否触发特技, 如果有特技则发动,防御值=0
    if ("破防".equals(kill.kill)) {
        defense = 0;
    }
    System.out.println(killAttack + "\t" + defense);
    return killAttack - defense;
}
```

```
/**
 * @return true 表示回合结束, false 表回合没有结束
 */
public boolean round(int sed) {
    // a->b
    Player a = null;
    Player b = null;
    // 判定攻击者和防御者
    if (sed == 0) {
        a = playerA;
        b = playerB;
    } else {
        a = playerB;
        b = playerA;
    }
    // 普通攻击： b的血量>=0
    double baseVal = baseAttack(sed);
    b.life -= baseVal;
    System.out.printf("%s对%s使用了普通攻击,伤害值%.2f\n", a.name, b.name, baseVal);
    if (b.life <= 0) {
        return true;
    }

    // 技能功能 for
    for (int i = 0; i < a.kills.length; i++) {
        double killVal = killAttack(a, b, a.kills[i]);
        b.life -= killVal;
        System.out.printf("%s对%s使用了%s,伤害值%.2f\n", a.name, b.name, a.kills[i].name, killVal);
        // 每次技能发动后判断血量
        if (b.life <= 0) {
            return true;
        }
    }

    return false;
}
```

```
/**
 * 战斗的方法
 */
public void fight() {

    for (int i = 0; i < 8; i++) {
        System.out.println("第" + (i + 1) + "回合开始:");
        boolean flag = round(0); // 攻
        if (flag) {
            System.out.println(playerA.name + "战斗胜利！！");
            return;
        }
        flag = round(1); // 防御
        if (flag) {
            System.out.println(playerB.name + "战斗胜利！！");
            return;
        }
    }
    System.out.println("平局！！");


}
```

测试

```java
        Kill k1 = new Kill();
        k1.name = "攻杀剑法";
        k1.probability = 0.5;
        k1.harms = new double[2];
        k1.harms[0] = 1.3;
        k1.harms[1] = 1.5;

        Kill k2 = new Kill();
        k2.name = "刺杀剑法";
        k2.probability = 0.4;
        k2.harms = new double[2];
        k2.harms[0] = 1;
        k2.harms[1] = 1;
        k2.kill = "破防";

        Kill k3 = new Kill();
        k3.name = "烈火剑法";
        k3.probability = 0.3;
        k3.harms = new double[2];
        k3.harms[0] = 3.0;
        k3.harms[1] = 3.0;

        Kill[] kills = {k1, k2, k3};

        // 测试普通攻击
        Player a = new Player();
        a.name = "关羽";
        a.kills = kills;

        Player b = new Player();
        b.name = "张飞";
        b.kills = kills;


        Fighting fighting = new Fighting();

        fighting.playerA = a;
        fighting.playerB = b;

//        double val1 = fighting.baseAttack(0);
//        System.out.println(val1);
//        double val2 = fighting.baseAttack(0);
//        System.out.println(val2);

//        double val = fighting.killAttack(a, b, a.kills[2]);
//        System.out.println(val);

        fighting.fight();
```