游戏说明：

玩家可以操纵自己的坦克，通过发射子弹击毁敌机，敌机用同样的方式击毁玩家的坦克。

基于java版的坦克大战实现思路：

1．0：游戏界面设计

--游戏的界面采用JPanel 游戏的主要活动区在这个JPanel上进行

--游戏的界面素材采用java代码的paint方法绘制坦克 炸弹效果导入外部图片素材

首先新建一个JPanel，绘制出坦克和子弹，主意，坦克的绘制必须确定一个基准点，此后坦克的位置依赖与此坐标。

因此，现在需要一个MyPanel类来承载游戏的界面可视化区域。该类继承自JPanel类。

MyPanel类的paint()方法可以很有效的绘制出坦克和子弹，注意到repaint()方法可以很方便的执行paint()方法，游戏的核心就是让显示器不停的刷新当前页面。通过repaint()很方便能做到这点。

而让MyPanel实现Runnable接口是很有必要的，游戏开始之后，我们让repaint()可以自动的运行，而不是通过响应某种事件。

现在，有了坦克和子弹之后，下一步就是让JFrame能监听到玩家按下空格键，这个很容易可以实现，当监听到玩家的事件之后，监视器马上执行发射子弹行为。这个行为是由坦克自身发出的，并且考虑到所有的坦克都有发射子弹的功能，为此，需要先写我们的Tank类。这个类是敌人坦克和我方坦克共有的，其余两者均继承自(Class) Tank类。

这个类的设计思路如下

该类定义的坦克的初始速度，方向，子弹的速度，并且具有发射子弹的功能。代码如下：

**public** **class** Tank {

**int** positionX; //坦克的初始位置x

**int** positionY; //坦克的初始位置y

**int** direct; //坦克的方向

**int** speed; //坦克速度

**int** bulletSpeed=10; //子弹速度

**public** **int** getPositionX() {

**return** positionX;

}

**public** **void** setPositionX(**int** positionX) {

**this**.positionX = positionX;

}

**public** **int** getPositionY() {

**return** positionY;

}

**public** **void** setPositionY(**int** positionY) {

**this**.positionY = positionY;

}

**public** **int** getDirect() {

**return** direct;

}

**public** **void** setDirect(**int** direct) {

**this**.direct = direct;

}

**public** **int** getSpeed() {

**return** speed;

}

**public** **void** setSpeed(**int** speed) {

**this**.speed = speed;

}

//坦克上移

**public** **void** moveUp()

{

}

//坦克向右

**public** **void** moveRight(){

}

//坦克下移

**public** **void** moveDown(){

}

//坦克左移

**public** **void** moveLeft(){

}

//坦克射击子弹行为

**public** **void** shout()

{

}

注意到shout这个方法，在用户按下空格键之后，通过监视器调用而发射子弹，但注意，子弹发射出去之后便不再受控制，为此，需要将每一个子弹都做成一个线程。出于这个缘故，并且考虑到子弹仍旧有属于自己的特性，比如速度、方向、初始位置等,需要将子弹单独做成一个类，这个类实现Runnable接口（为了后面产生其他武器，这里通过接口实现线程而不是继承而来）。当一个子弹杯创建之后，应当马上启动子弹的线程。

现在可以让我方的子弹在按下空格之后实现发射一个子弹，下一步是让玩家可以通过上下左右键控制坦克的方向。当玩家敲下相应的键之后，监视器马上调用继承自Tank的MTank(我方坦克)。这点很容易就试下，代码类似地：

//坦克上移

**public** **void** moveUp()

{

**if**(**this**.positionY>0) //保证用户不出界

**this**.positionY-=**this**.speed;

}

下面的任务是让敌人也可以发射子弹，注意，敌人的子弹是不受玩家控制的，并且要求敌人子弹不能连发，也就是说，敌人的一个子弹在死亡之前，是不容许在产生新的子弹的，为了让敌人自动发射子弹，我们改变思路，找一个容器来存放敌人的子弹，在paint里面，通过遍历容器里面的子弹来模拟自动发射，注意：一旦程序执行,paint总是在不停的调用,因为JPanel已经是个线程，在它的run()函数中，用while循环来保证paint永远运转。

考虑到线程的安全问题，采用Vector 向量来存放子弹，

Vector<Bullet> bullets=**new** Vector<Bullet>();

注意，每辆坦克都有自己的子弹容器，它是Tank类的一个属性。下一步，让坦克动起来，理所当然地，将敌方坦克做成一个线程，该线程让坦克自动运行，类似地在线程run中这样做：

**while**(**true**){

**try**{

Thread.*sleep*(30);

}

**catch**(Exception e){

}

**this**.aotoMove();

}

到目前为止：我方的坦克可以通过控制自由移动并发射子弹，敌人的坦克也实现了自由移动并自动发射子弹。

下面要做的就是，当我方坦克击中敌人坦克之后，敌人坦克消失，反之亦然。

实现这一切比想象中的还要容易，只需要在paint中加入碰撞检测。

hitTest()函数将提供检测机制。当我方的坦克发射的子弹落在敌人坦克所在的区域，我们应该让敌方的坦克消失。

为了统一并实现该要求，需要定义一个Vector存放这个游戏中所有的敌方坦克，这样在paint绘制的时候，只需要读取Vector向量来动态的产生坦克。

下面，开始具体碰撞检测：

注意：

一旦检测到我方的坦克的子弹落在敌方坦克的区域，执行某操作，产生击毁效果。

一旦检测道敌方坦克的子弹落在我方坦克的区域，执行某操作，产生击毁效果。

一旦检测到子弹飞出游戏区域，执行某操作，从向量中移除该子弹

一旦检测到敌人的坦克互相碰撞或者遇到墙壁，执行某操作，改变方向

碰撞检测的过程必须要注意的是，由于碰撞检测放在paint类，执行速度过快，在检测完毕改变方向之后，一定要有一个延迟，不能马上判断，否则会出现“无头坦克”，到处乱撞。

解决的方法有：在坦克移动的时候，用for循环+Thread.sleep来控制坦克在一次移动中必须走完指定的路线。

2：在改变方向后，有一个小小的延迟，保证在此期间，坦克有一个微小的挣脱力来挣脱碰撞带来的困扰。

最后，在增加游戏开始的画面：新建一个JPanel，当游戏开始的时候显示这个界面，通过times%2的方法让这个界面的字符串有个闪的过程。

将MyPanel的创建放到一个合适的逻辑中。而不是一开始就。当执行MyPanel的时候，记得先移除新建的JPanel。

截止目前：游戏的基本功能已经完结。V\_1.0





