```java
package hevs.fragil.patapon.physics;

import com.badlogic.gdx.Gdx;
import com.badlogic.gdx.math.Vector2;

import ch.hevs.gdx2d.components.physics.primitives.PhysicsPolygon;

public class BodyPolygon extends PhysicsPolygon implements CollidedObject {
    static Vector2 dimensions =  new Vector2(3,80);
    int collisionGroup;
    private float life;
    static int nArrows;
    static Vector2 body[] = {
            new Vector2(-30, 0),
            new Vector2(-30, 60),
            new Vector2(0, 80),
            new Vector2(30, 60),
            new Vector2(30, 0)
    };

    public BodyPolygon(Vector2 position, int collisionGroup, int life) {
        //Ca c'est vraiment super !
        super("arrow"+nArrows, position, body,  1000f, 0f, 1f, true);
        this.life = life;
        getBody().setBullet(true);
        this.collisionGroup = collisionGroup;
        setCollisionGroup(collisionGroup);
        nArrows++;
    }
    public void moveToLinear(int position, double travelTime) {
        travelTime *= 1000;
        double distanceToTravel = position - (int)getBodyPosition().x;
        double globalSpeed = distanceToTravel / travelTime;
        int fps = Gdx.graphics.getFramesPerSecond();

        // Check if this speed will cause overshoot in the next time step.
        // If so, we need to scale the speed down to just enough to reach
        // the target point.
        double stepDistance = globalSpeed * (1.0/fps);
        if ( Math.abs(stepDistance) > Math.abs(distanceToTravel) )
            globalSpeed *= ( distanceToTravel / stepDistance );

        double desiredAcceleration = Math.abs(globalSpeed) * distanceToTravel;
        double changeInAcceleration = desiredAcceleration - this.getBodyLinearVelocity().x;

        double force = this.getBodyMass() * fps * changeInAcceleration;
        this.applyBodyForceToCenter((float)force, 0, true);
    }
    @Override
    public int getCollisionGroup() {
        return collisionGroup;
    }
    @Override
    /** Return true when damage leads to death */
    public boolean applyDamage(float damage) {
        if(life > 0){
            life -= damage;
            if(life <= 0){
                kill();
                return true;
            }
            else return false;
        }
        else return false;

    }
    public float getLife(){
        return life;
    }
    public boolean isDead(){
        if(life <= 0) return true;
        else return false;
    }
    public void kill() {
        getBody().setFixedRotation(false);
        applyBodyAngularImpulse(-1300, true);
    }
}
```