1. CREACIÓN DE ENEMIGO.

Para crear al enemigo, vi que tenemos sugerido en el diagrama de clases, un enemigo que tiene escudo, y una interfaz llamada *IHaveShield.*

Me gustó la idea , empecé a revisar todo el codigo de todas las clases. lo primero a destacar es que en el game manager, habia el siguiente codigo comentado:

```
else if (sprite instanceof IHaveShield){
                temps.add(new SpriteTemp(temps, sprite.getRect().centerX(),
sprite.getRect().centerY(),
                                EXPLOSION_9_SPRITE_IMAGE, 9));
//                  if (((EnemyBarrier) sprite).impact()){
//                      itSprite.remove();
//                  }
```

El código se trata de una comprobación para ver si un enemigo tiene escudo o no (Si lo tiene, se lo añade) y para ver si le han dado un golpe (Para quitarle el escudo).

Lo descomenté para que funcionara, ya que imagino que está comentado así porque en un principio no hace falta para el funcionamiento del juego. solamente sirve si algún alumno (como yo) decide hacer a los enemigos con escudo.

luego pasé a crear el propio escudo. me he basado en las siguientes ideas. el escudo es un enemigo en sí, que no dispara, y que sigue a una nave enemiga quedandose encima de este hasta que le disparan.

```
public class EnemyBarrier extends AEnemy implements IHaveShield {
```

Tenemos que implementar el metodo impact y hacer que se pegue a la nave.
para ello creamos un atributo de la clase (targetShip) que apunta a la nave a la cual tiene que seguir el escudo.

```
private EnemyShip targetShip;
```

Para hacerme la vida mas facil, he añadido getters y setters de las coordenadas de los Sprites, ya que no me resultaban comodos a la hora de utilizarlos.

```
public void setX(int x) {
    this.x = x;
}
```

```
public int getX() {
    return x;
}


public void setY(int y) {
    this.y = y;
}


public int getY() {
    return y;
}


public int getWidth() {
    return width;
}


public int getHeight() {
    return height;
}
```

Luego me puse a terminar la clase de EnemyBarrier. y le añadí el update() y el impact()

Al final he conseguido que siga a la nave de estas dos formas. (Lo aclaro porque viste que en clase tenia bastantes dudas de como hacerlo):

-Con el atributo de la nave a la que sigue (targetShip)
-Haciendo referencia a las coordenadas de targetShip en el update() del escudo.

2. <u>CREACIÓN DEL NIVEL:</u>

Primero para crear el nivel, tenemos que ir al enemyspawner.

En el comprobador de nivel de la clase Enemyspawner, tenemos que crear nuestro nivel 4.

```
        case 4:
            enemies: crearEnemigosNivelJuan(gameRect);
    }
```

Creando el case, te das cuenta de que hay que crear un metodo para inicializar los enemigos que spawnean en el nivel.

le he llamado:

```
  private static void crearEnemigosNivelJuan(Rect gameRect) {}
```

Para mantenerlo sencillo, voy a coger los enemigos del nivel 1 pero les voy a añadir el escudo que he hecho antes.

Lo unico a destacar de aqui, es que con un bucle for, creo un escudo para cada nave del nivel

```
        …
    for (EnemyShip ship : enemyShips) {
        EnemyBarrier barrier = new EnemyBarrier(ENEMYBARRIER4_SPRITE_IMAGE, 1, 1,
ship);
        enemies.add(barrier);
    }

    enemies.addAll(enemyShips);


    EnemyShipGroup eg1 = new EnemyShipGroup(gameRect, enemyShips);
    eg1.setXSpeed(vx);
    enemies.add(eg1);

    return enemies;
  }
```

Tambien he metido la foto del escudo, que ya estaba implementado en las constantes del proyecto.

```
        ENEMYBARRIER4_SPRITE_IMAGE
```

En appConsts, cambio el numero de niveles. ahora son 4.