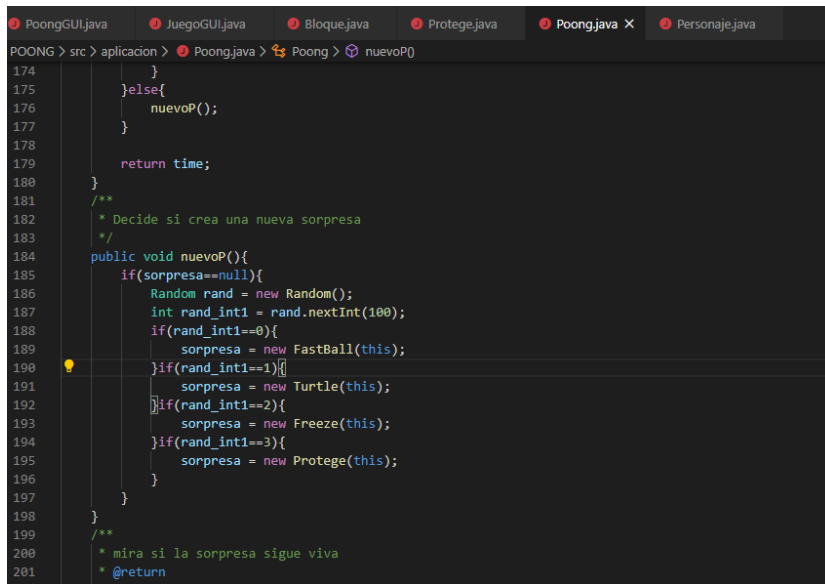


NOTA ESPERADA : 3.0

EXTENDIENDO:

Se crea un nuevo poder que da un escudo contra bloques

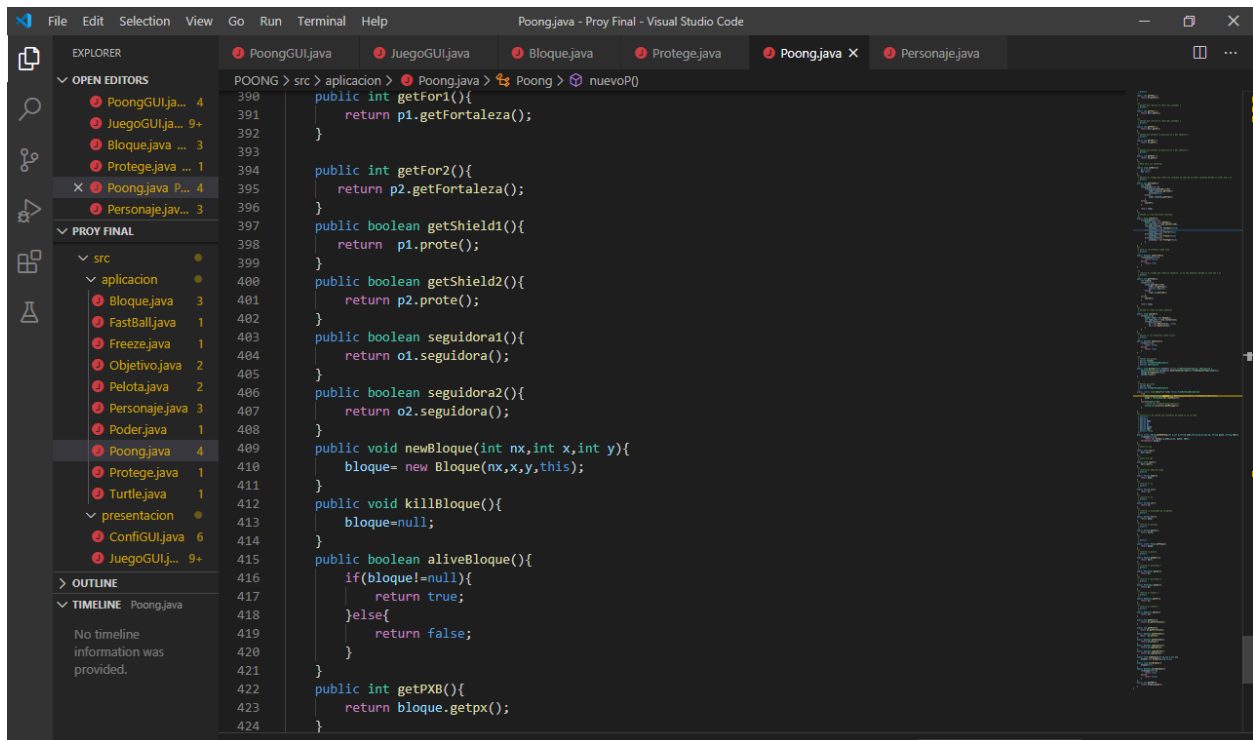
Clase poong



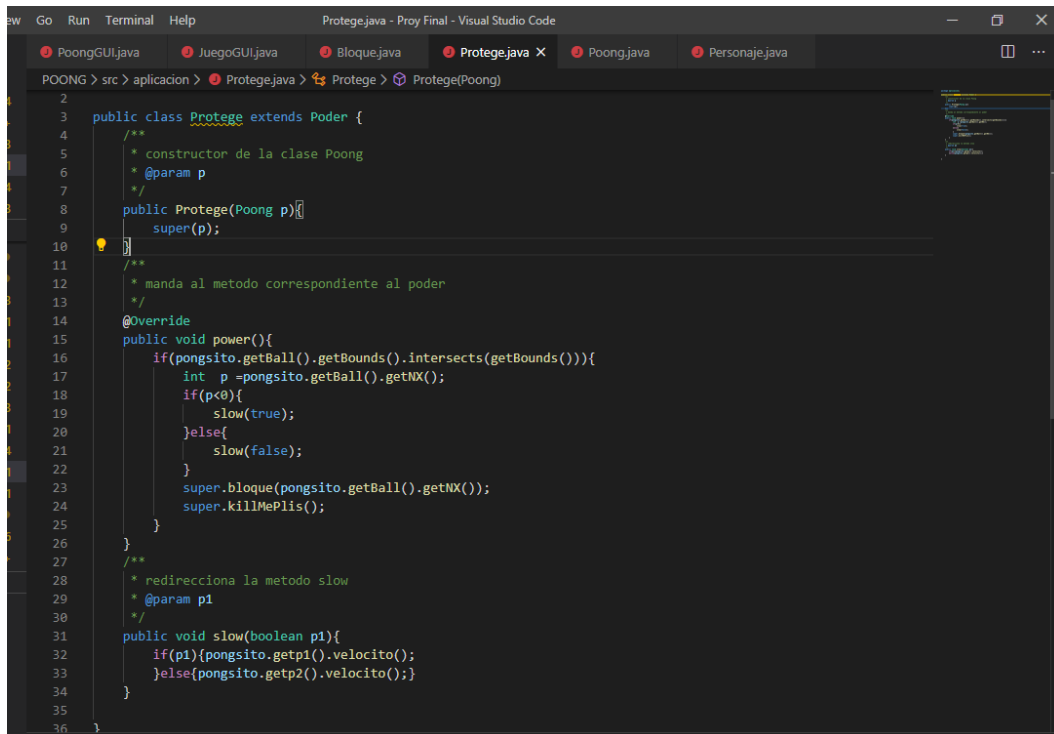
```
174     }
175   }else{
176     nuevoP();
177   }
178
179   return time;
180 }
181 /**
182  * Decide si crea una nueva sorpresa
183  */
184 public void nuevoP(){
185     if(sorpresa==null){
186         Random rand = new Random();
187         int rand_int1 = rand.nextInt(100);
188         if(rand_int1==0){
189             sorpresa = new FastBall(this);
190         }if(rand_int1==1){
191             sorpresa = new Turtle(this);
192         }if(rand_int1==2){
193             sorpresa = new Freeze(this);
194         }if(rand_int1==3){
195             sorpresa = new Protege(this);
196         }
197     }
198 }
199 /**
200  * mira si la sorpresa sigue viva
201  * @return
```

Como no se tenían implementados los bloques ni la fortaleza se trato de implementar

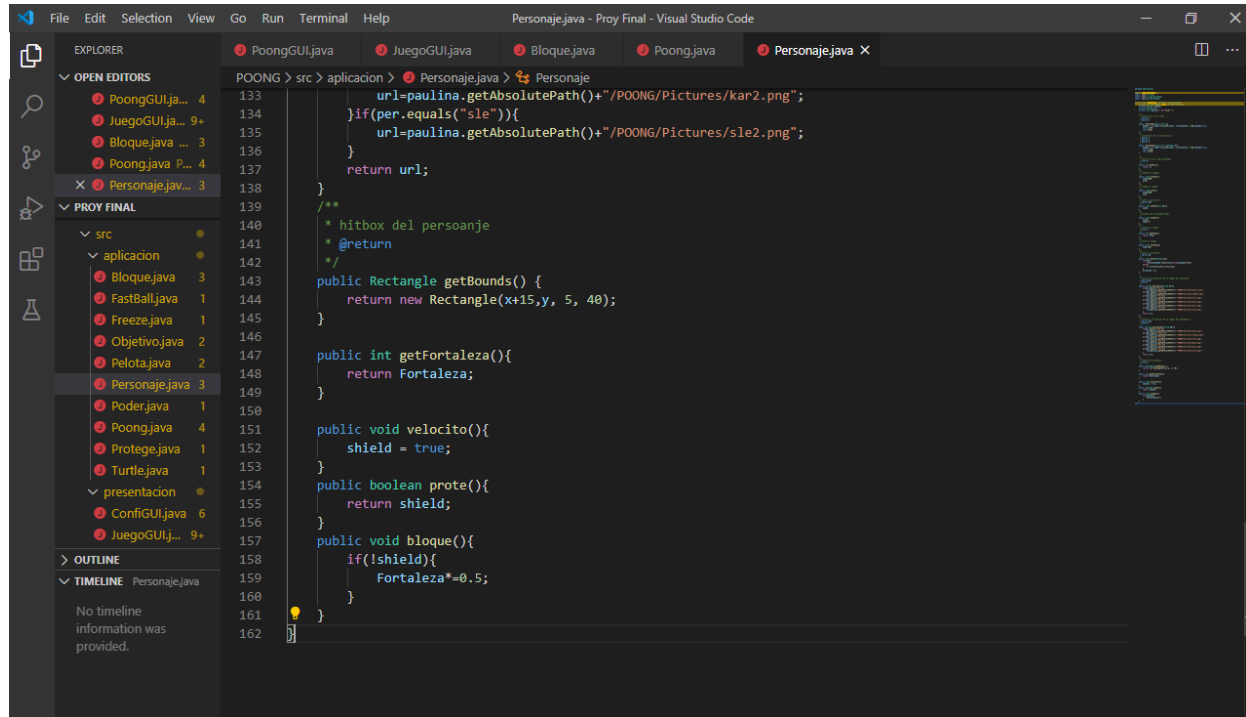
Métodos adicionales creados en la clase poong:



Clase del nuevo poder:

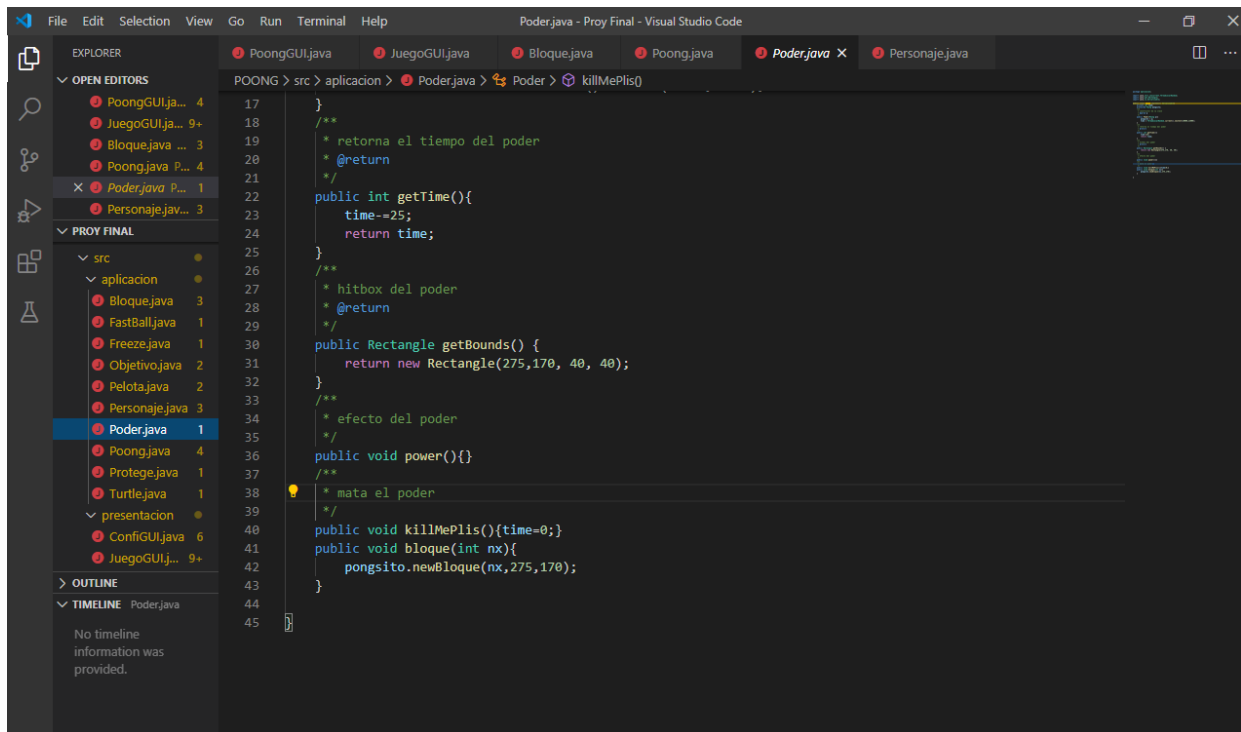


Métodos adicionales en la clase personaje:

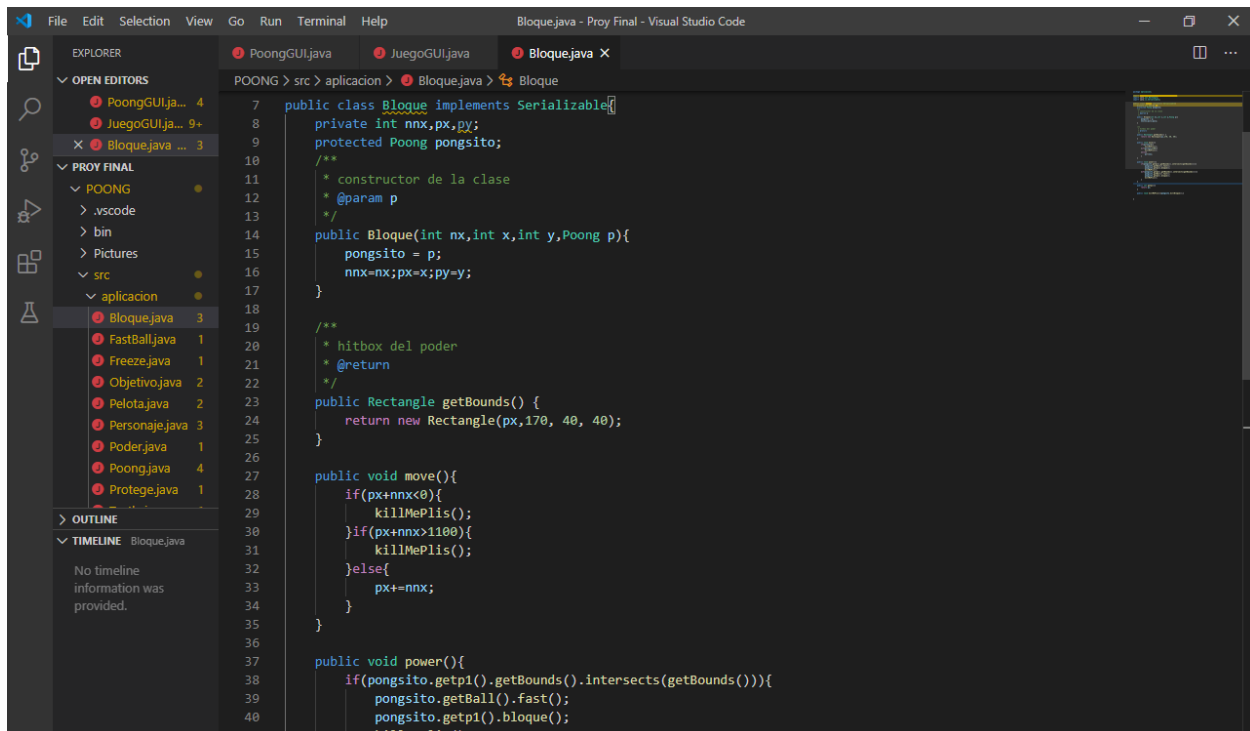


```
133     url=paulina.getAbsolutePath()+"/POONG/Pictures/kar2.png";
134 }if(per.equals("sle")){
135     url=paulina.getAbsolutePath()+"/POONG/Pictures/sle2.png";
136 }
137     return url;
138 }
139 /**
140  * hitbox del persoanje
141  * @return
142  */
143 public Rectangle getBounds() {
144     return new Rectangle(x+15,y, 5, 40);
145 }
146
147 public int getFortaleza(){
148     return Fortaleza;
149 }
150
151 public void velocito(){
152     shield = true;
153 }
154 public boolean prote(){
155     return shield;
156 }
157 public void bloque(){
158     if(!shield){
159         Fortaleza*=0.5;
160     }
161 }
162 }
```

En la clase padre de la sorpresa se creo un método que crea bloques , este método se ejecuta cada vez que la pelota colisiona con una sorpresa



Se creo la clase bloque

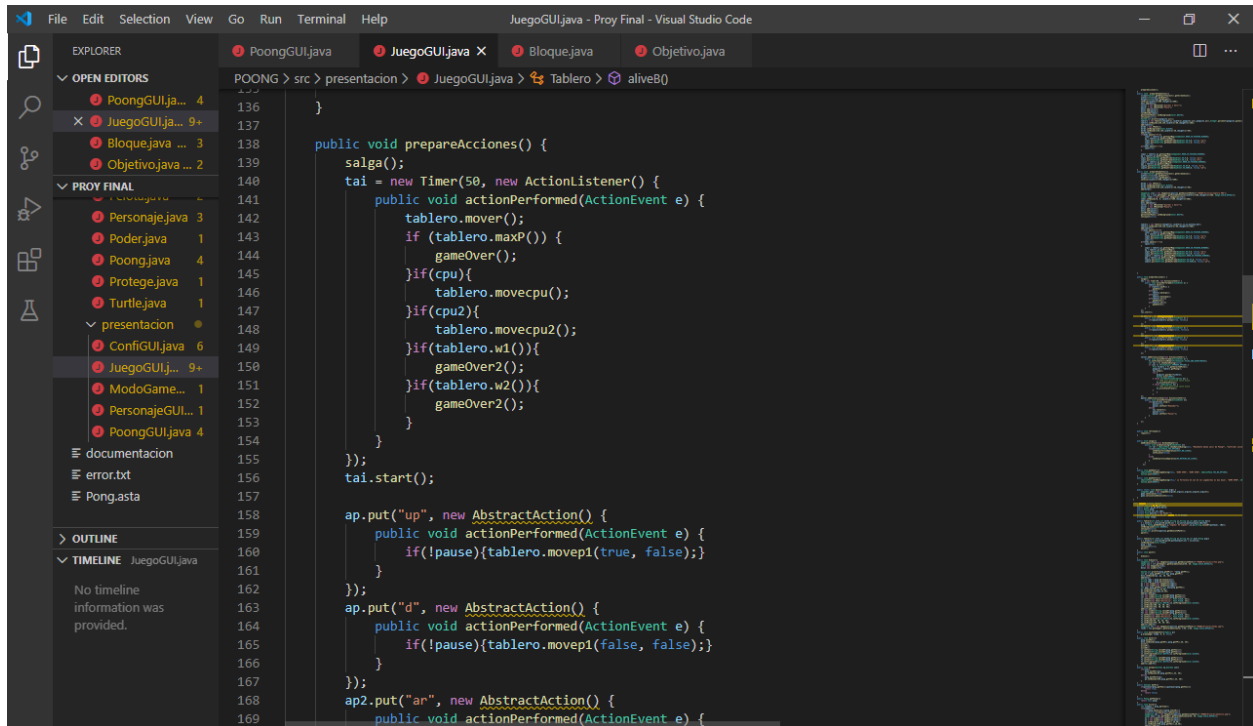


```
28     if(px+nnx<0){
29         killMePlis();
30     }if(px+nnx>1100){
31         killMePlis();
32     }else{
33         px+=nnx;
34     }
35 }
36
37
38 public void power(){
39     if(pongsito.getp1().getBounds().intersects(getBounds())){
40         pongsito.getBall().fast();
41         pongsito.getp1().bloque();
42         killMePlis();
43     }if(pongsito.getp2().getBounds().intersects(getBounds())){
44         pongsito.getBall().fast();
45         pongsito.getp2().bloque();
46         killMePlis();
47     }
48 }
49
50 public int getpx(){
51     return px;
52 }
53
54 public void killMePlis(){pongsito.killBloque();}
55
56 }
```

Cuando se crea un objetivo se va a un random que decide si este sigue o no a la pelota con respecto al eje y

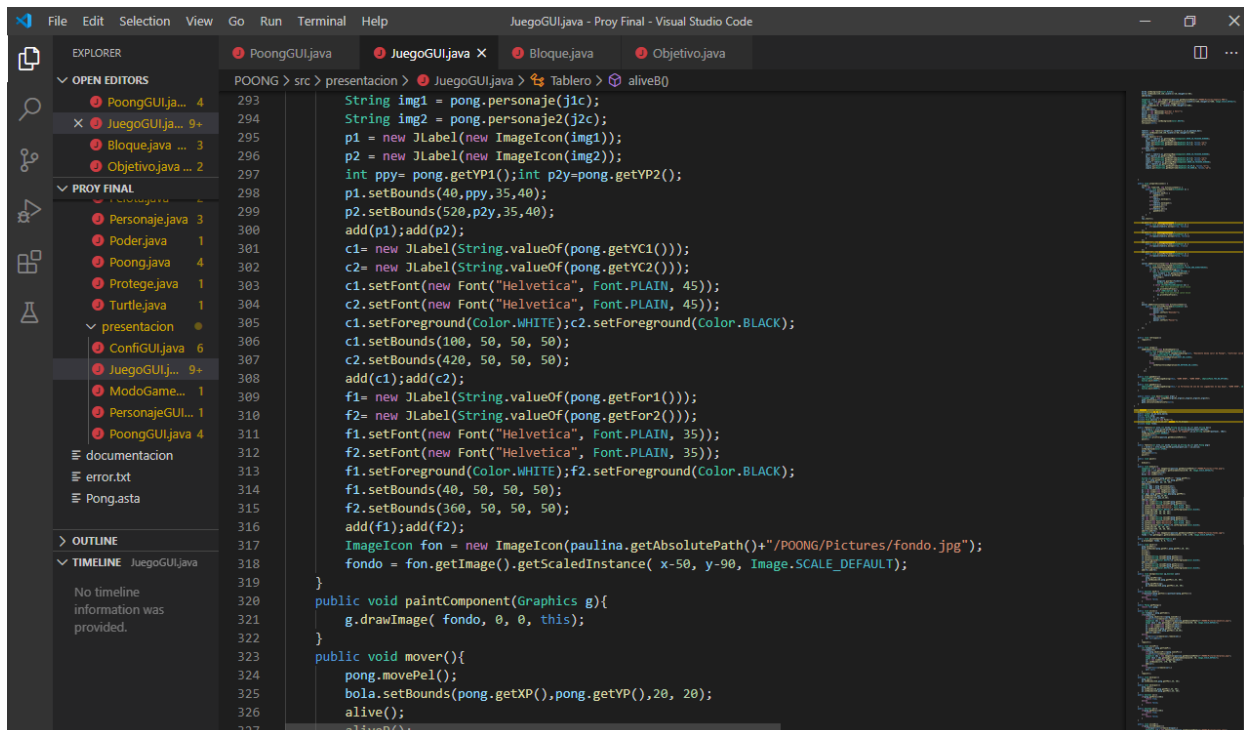
```
12
13
14 public Objetivo(int alto){
15     hi = alto;
16     x=0;
17     tiempo = 10000;
18     randomY();
19 }
20
21 /**
22  * constructor de la clase objetivo para el jugador 1
23  * @param alto
24  * @param p1
25  */
26 public Objetivo(int alto,boolean p1){
27     hi = alto;
28     x=500;
29     tiempo = 10000;
30     randomY();
31 }
32
33 /**
34  * hubica el objetivo en una posicion y random
35  */
36 public void randomY(){
37     Random rand = new Random();
38     int rand_int1 = rand.nextInt(hi-125);
39     y = rand_int1;
40     Random rand2 = new Random();
41     int rand_int2 = rand.nextInt(3);
42     if(rand_int2==0){
43         snitch=false;
44     }if(rand_int2==1){
45         snitch=true;
46     }
47 }
```

En la capa de presentacion se valida la foraleza de los personajes y otras cosas



```
136 }
137
138 public void prepareAcciones() {
139     salga();
140     tai = new Timer(50, new ActionListener() {
141         public void actionPerformed(ActionEvent e) {
142             tablero.mover();
143             if (tablero.maxP()) {
144                 gameOver();
145             }if(cpu){
146                 tablero.movecpu();
147             }if(cpu2){
148                 tablero.movecpu2();
149             }if(tablero.w1()){
150                 gameOver2();
151             }if(tablero.w2()){
152                 gameOver2();
153             }
154         }
155     });
156     tai.start();
157
158     ap.put("up", new AbstractAction() {
159         public void actionPerformed(ActionEvent e) {
160             if(!pause){tablero.movep1(true, false);}
161         }
162     });
163     ap.put("d", new AbstractAction() {
164         public void actionPerformed(ActionEvent e) {
165             if(!pause){tablero.movep1(false, false);}
166         }
167     });
168     ap2.put("ar", new AbstractAction() {
169         public void actionPerformed(ActionEvent e) {
```

En la clase tablero se crean JLabels para las fortalezas de cada jugador y otro para los bloques



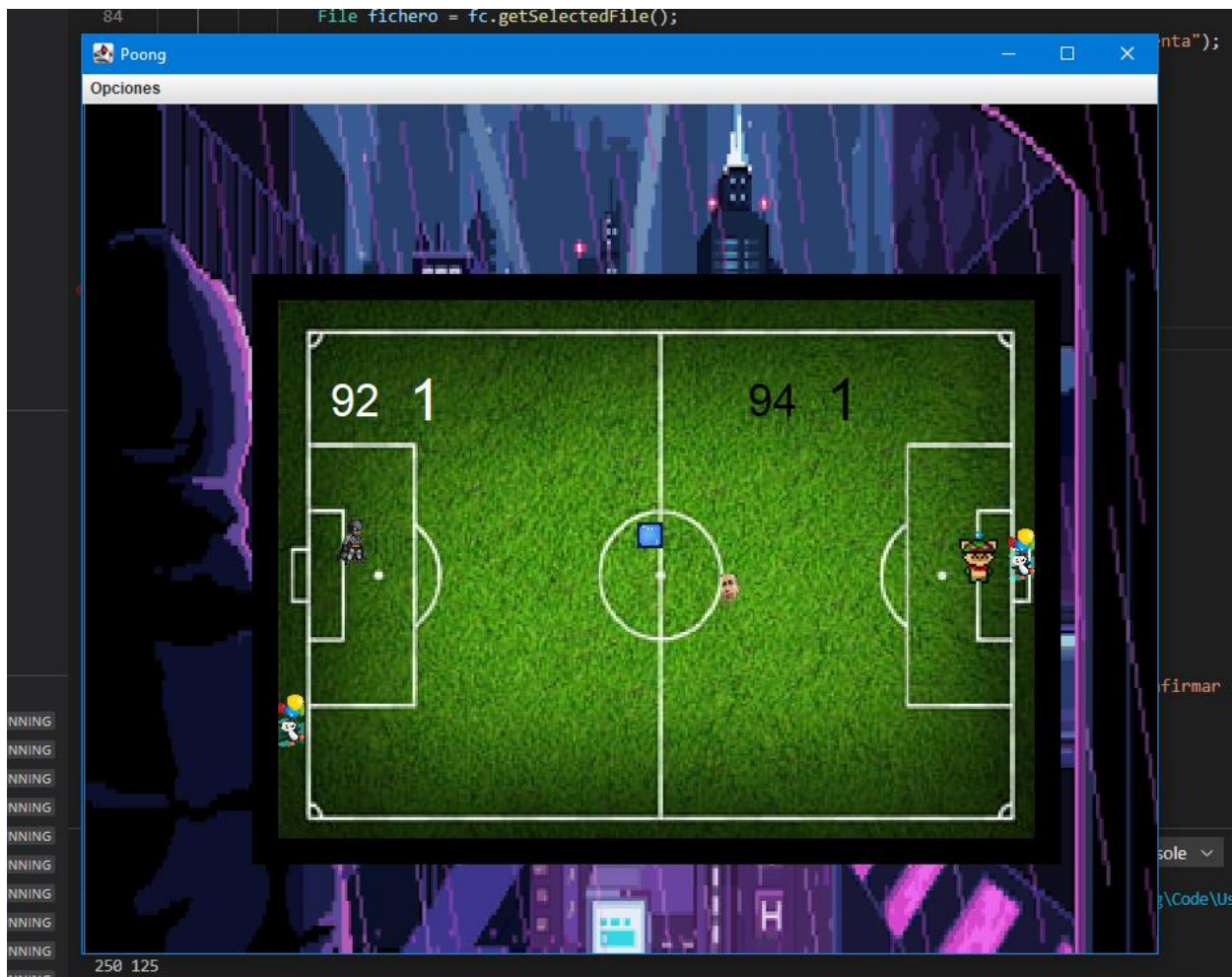
```
293 String img1 = pong.personaje(j1c);
294 String img2 = pong.personaje2(j2c);
295 p1 = new JLabel(new ImageIcon(img1));
296 p2 = new JLabel(new ImageIcon(img2));
297 int ppy= pong.getYP1();int p2y=pong.getYP2();
298 p1.setBounds(40,ppy,35,40);
299 p2.setBounds(520,p2y,35,40);
300 add(p1);add(p2);
301 c1= new JLabel(String.valueOf(pong.getYC1()));
302 c2= new JLabel(String.valueOf(pong.getYC2()));
303 c1.setFont(new Font("Helvetica", Font.PLAIN, 45));
304 c2.setFont(new Font("Helvetica", Font.PLAIN, 45));
305 c1.setForeground(Color.WHITE);c2.setForeground(Color.BLACK);
306 c1.setBounds(100, 50, 50, 50);
307 c2.setBounds(420, 50, 50, 50);
308 add(c1);add(c2);
309 f1= new JLabel(String.valueOf(pong.getFor1()));
310 f2= new JLabel(String.valueOf(pong.getFor2()));
311 f1.setFont(new Font("Helvetica", Font.PLAIN, 35));
312 f2.setFont(new Font("Helvetica", Font.PLAIN, 35));
313 f1.setForeground(Color.WHITE);f2.setForeground(Color.BLACK);
314 f1.setBounds(40, 50, 50, 50);
315 f2.setBounds(360, 50, 50, 50);
316 add(f1);add(f2);
317 ImageIcon fon = new ImageIcon(paulina.getAbsolutePath()+"POONG/Pictures/fondo.jpg");
318 fondo = fon.getImage().getScaledInstance( x-50, y-90, Image.SCALE_DEFAULT);
319
320 public void paintComponent(Graphics g){
321     g.drawImage( fondo, 0, 0, this);
322 }
323 public void mover(){
324     pong.movePel();
325     bola.setBounds(pong.getXP(),pong.getYP(),20, 20);
326     alive();
327     aliveP();
```

```

public void aliveB(){
    if(pong.aliveBloque()){
        if(bloque!=null){remove(bloque);}
        ImageIcon cr7 = new ImageIcon(paulina.getAbsolutePath()+"/POONG/Pictures/logo.jpg");
        Image sca = cr7.getImage().getScaledInstance(20,20, Image.SCALE_DEFAULT);
        cr7 = new ImageIcon(sca);
        bloque= new JLabel(cr7);
        bloque.setBounds(pong.getPXB(), 170, 20, 20);
        add(bloque);
    }
    repaint();
}
}

```

SE VE asi:



Lastimosamente el bloque no se mueve y no se porque :C pero es el JLabel La colisión si la detecta.

Alguno objetivos (pinguinos) se mueven en y siguiendo la pelota

-Los personajs tienen un bollean shield que se poje true cuando agarran el poder que los portege , si tienen esto no pierden fortaleza al entrar en contacto con un bloque

REFACTORIZACION:

Ventajaas: Cumplimos con varias Funcionalidades que son fáciles de ajustar o extender ej:Sorpresas

Desventajas: No funciona el abra

Patrones:

El patrón **Observer** puede ser utilizado cuando hay objetos que dependen de otro, necesitando ser notificados en caso de que se produzca algún cambio en él.

Ese patron fue implementado para el comportamiento de la cpu que se mueva en el eje y con respecto a pelota , entonces el movimiento de la cp depende dl movimiento de la pelota