Clase Bala

```
import java.awt.Color;
import java.awt.Graphics;
/**
 * Clases Balas. Super clase
 * para crear disparos de las naves
 * @param poscicion: Coordenadas del disparo
 * escala: escala para establecer el tamaño del disparo
 * color: Color del disparo.
 * @author JMCoca
 * @version 2012
 */
public class Balas {
  //campos
  private Coordenada posicion;
  private int escala;
  private int estado;
  protected Color color;

  /**
   * Construcor de la clase bala
   * @param posicio en el tablero
   * estado inicial vivo=1
   */
  public Balas(Coordenada posicion) {
      this.posicion = posicion;
      escala =16; //campo escala por siqueremos cambiar el tamañoe
      estado=1;
  }

  //consultas
  public int getCoordenadaX() {return posicion.getX();}
  public int getCoordenadaY() {return posicion.getY();}
  public int getEstado() {return estado;}

  /**
   * Control del movimiento Vertical de la bala
   * @param pasos que se desplaza hotizontalmente
   */
  public void moverVertical(int pasos) {
        this.posicion.mueveY(pasos);
  }

  /**
   * Nave Alcanzada
   * si es una bomba invasora y alcanza
   * al defensor
   * cambia de color a amarillo
```

```
   */
  public void naveAlcanzada() {
        estado=0;
        color=Color.yellow;
  }

    /**
   * pinta las balas  es un rectangulo de medidas la mitad
   * que un cuadrado de la nave.
   * el +4 es para centrarlo en el morro de la nave
   */
  public void pintar(Graphics g) {
       g.setColor(color);
       g.fill3DRect((posicion.getX()*escala)+4,posicion.getY()*escala,escala/2,escala,true);
  }
}
```

Subclase Bomba

```
import java.awt.Color;
/**
 * clase Bomba
 * @author JMCoca
 * @version 2012
 */
public class Bomba extends Balas {
  /**
   * Constructor del defensor.
   * @param posicion Coordenadas en el tablero
   */
  public Bomba(Coordenada posicion) {
        super(posicion);
        color = Color.magenta; //Color de la bomba
    }
}
```

Subclase Disparo

```
import java.awt.Color;
/**
 * Clase disparo subclase de bala
 *
 * @author JMCoca
 * @version 2011
 */
public class Disparo extends Balas {
  /**
   * Constructor del defensor.
   * @param posicion Coordenadas en el tablero
   */
```

```
  public Disparo(Coordenada posicion) {
        super(posicion);
        color = Color.red;
  }
}
```

Clase Nave

```
import java.awt.Color;
/**
 * Clase padre de las naves
 * tiene un estado (vivo o muerto)
 * @param escala: por si nos interesan otras dimensiones en el juego.
 * @param posicion: Coordendas para situar la nave en el plano
 * @author JMCoca
 * @version 2012
 */
public class Nave {
  protected int escala;

  /**
   * campos
   */
  protected int estado;

  protected Coordenada posicion;

  protected Color color;

  public Nave(Coordenada posicion) {
     this.posicion = posicion;
     escala=16;
     estado=1; // en principio nave estado vivo
  }

  // consultas
  public int getCoordenadaX() {return posicion.getX();}
  public int getCoordenadaY() {return posicion.getY();}
  public int getEstado() {return estado;}

  /**
   * Control del movimiento Horizontal de la nave
   * @param pasos que se desplaza hotizontalmente
   */
  public void moverHorizontal(int pasos) {
        posicion.mueveX(pasos);
  }

  /**
   * Control del movimiento Vertical de la nave
```

```
   * @param pasos que se desplaza Verticalmente
   */
  public void moverVertical(int pasos) {
        posicion.mueveY(pasos);
  }

  /**
   * Cambio de estado
   * @param 1 o 0 Vivo/muerto
   */
  public void setEstado(int i) {
        estado = i;
  }

  /**
   * Nave Alcanzada
   * Cuando una nave
   * es alcanzada estado muerto=0;
   */
  public void naveAlcanzada() {
        estado = 0;
  }

}
```

Subclase Defensor

```
import java.awt.Color;
import java.awt.Graphics;
/**
 * Clase defensor
 * LLama a la clase super
 * asigna color y forma
 * en este caso dos rectangulos
 * @author JMCoca
 * @version 2012
 */
public class Defensor extends Nave {
  /**
   * Constructor del defensor.
   * @param posicion Coordenadas en el tablero
   */
  public Defensor(Coordenada posicion) {
        super(posicion);
        color=Color.blue;
  }

  /**
   * posiciona la pieza en el tablero (la pinta)
   * @param g.
```

```
   */
  public void pintar(Graphics g) {
        int x =posicion.getX();
        int y =posicion.getY();
        g.setColor(color);
        g.fill3DRect( x*escala,y*escala,escala,escala,true);
        g.fill3DRect(( (x-1)*escala),(y+1)*escala,escala*3,escala,true);
  }

  /**
   * Nave Alcanzada
   */
  public void naveAlcanzada() {
      super.naveAlcanzada();  //llama al metodo de la super clase
      color = Color.yellow;
  }

}
```

Subclase Invasor

```
import java.awt.Color;
import java.awt.Graphics;
/**
 * Clase invasor
 * formada por cuatro cuadrados
 * de color verde
 *
 * @author JMCoca
 * @version 2012
 */
public class Invasor extends Nave
{
    public int estado;
    /**
     * Constructor del defensor.
     * @param posicion Coordenadas en el tablero
     */
    public Invasor (Coordenada posicion){
        super(posicion);
    }

    /**
     * posiciona la pieza en el tablero (la pinta)
     * @param g.
     */

    public void pintar(Graphics g){
        int x =posicion.getX();
        int y =posicion.getY();
        g.setColor(Color.green);
```

```
        g.fill3DRect(x*escala ,(y*escala) ,escala,escala,true);
        g.fill3DRect( (x+1)*escala ,(y*escala) ,escala,escala,true);
        g.fill3DRect( (x-1)*escala ,(y*escala) ,escala,escala,true);
        g.fill3DRect( (x*escala) ,(y+1)*escala ,escala,escala,true);
    }
}
```

Clase coordenda

```
/**
 * Clase Coordenadas.
 * Posiciona y mueven las naves y los disparos.
 * @author JoseM Coca de la Torre
 * version 2012
 */
public class Coordenada {
  //coordendas
  private int x,y;
  /**
   * Constructor de la clase Coordenda
   * @param fila y
   * @param columna x
   */
  public Coordenada(int x, int y) {
        this.x = x;
        this.y = y;
  }

  // consultas
  public int getX() {return x;}  //devuelve Coordenada x
  public int getY() {return y;}  //devuelve Coordenada y

  //Metodos de movimiento de las piezas

  public void mueveX(int i) {x+=i;} // Movimiento horizontal
  public void mueveY(int i) {y+=i;} // Movimiento Vertical
}
```

Clase SpaceInvader_IGU

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

/**
 * Clase Principal. LLama a la Clase PanelDeJuego
 * @autor JmCoca
 * @version mayo 2012
 */
public class SpaceInvader_IGU{
```

```
//CAMPOS
private JFrame ventana;
private PanelDeJuego panelDeJuego;
/**
 * Programa
 */
public static void main(String args[]){ new SpaceInvader_IGU();}

/**
 * Ejecuta la aplicacion
 * la muestra en pantalla
 */
public SpaceInvader_IGU(){
   construirVentana();
   while (!panelDeJuego.juegoFin()){
      ejecutar();
   }
}

/**
 * Crea la ventana Swing y su contenido.
 */
private void construirVentana()
{
   ventana = new JFrame("Invasores del Espacio JMCoca Beta");
   ventana.setResizable(false);
   panelDeJuego = new PanelDeJuego(); // panel de Juego
   ventana.add(panelDeJuego);
   ventana.pack();
   ventana.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   ventana.addKeyListener(new KeyAdapter()
     {//control de teclas
         public void keyPressed(KeyEvent e) {

            if(e.getKeyCode()==KeyEvent.VK_O){
                panelDeJuego.movHorizontal(-1);//desplazamiento a la izquierda
            }else   if(e.getKeyCode()==KeyEvent.VK_P){
               panelDeJuego.movHorizontal(1);//desplazamiento a la derecha
            }else   if(e.getKeyCode()==KeyEvent.VK_SPACE){
               if (!panelDeJuego.hayDisparo()){
                  panelDeJuego.disparaDefensor();
               }
            }
         }
   });
   //para que la aplicacion aparezca al inicio en el centro de la pantalla
   Dimension d = Toolkit.getDefaultToolkit().getScreenSize();
   ventana.setLocation(d.width/2 - ventana.getWidth()/2, d.height/2 - ventana.getHeight()/2);

   ventana.setVisible(true);
}
```

```
  /**
   * LLamada al bucle del juego
   * primero un pausa de 1/2 segundo
   */
   public void ejecutar(){
      wait(500);//pausa anterior a que comienze el juego
      while (!panelDeJuego.juegoFin()){
         panelDeJuego.movimientosJuego();
         wait(200);                    //control de la velocidad del juego
         panelDeJuego.disparaInvasor();
       }
  }

  /**
   * Para controlar el timer
   * @param tiempo en milisegundos
   */
   public void wait(int milliseconds){
      try
      {
         Thread.sleep(milliseconds);
      }
      catch (InterruptedException e)
      {
         // ignoring exception at the moment
      }
   }
}
```

Clase panelDeJuego

```
import java.awt.*;
import javax.swing.*;
import java.awt.Graphics;
import java.util.Random;
/**
 * PanelDeJuego muestra el tablero, le añade un fondo color negro
 * como tablero y añade las piezas, del juego
 *
 * @author  Jose M Coca de la Torre
 * @version mayo 2012
 */
public class PanelDeJuego extends JPanel {
  private int ancho,alto,escala,sentido;
  private int nInvasores;
  private int cInvasores;
  private Defensor defensor;

  private Invasor[][] flotaInvasora;//matriz de invasores
  private int filaInv,columnInv;
```

```
  private Balas disparoDef;
  private Balas bombaInvasor;
  private boolean hayDisparo;
  private boolean hayBombaInvasor;
  private boolean juegoFin;

  /**
   * Constructor del panel de juego
   */
  public PanelDeJuego() {
       juegoFin = false; //Inicializacion de los campos
       escala = 16;
       ancho = 80;
       alto = 40;
       defensor = new Defensor(new Coordenada(ancho / 2, alto - 3)); //crea el denfesor

       filaInv = 4;
       columnInv = 9;
       nInvasores = filaInv * columnInv;
       cInvasores = nInvasores;
       flotaInvasora = new Invasor[columnInv][filaInv];

       for(int i = 0; i < columnInv; i++){                //generacion de la flota invasora
          for(int j = 0; j < filaInv; j++){
                flotaInvasora[i][j] = new Invasor(new Coordenada((5 * i) + 5, 2 +(3 * j)));
          }
       }
       hayDisparo = false;
       hayBombaInvasor = false;
       sentido = 1;//sentido inicial del movimiento de los invasores

  }

  //  metodos privados
  /**
   * Movimiento continuo de los invasores
   */
  private void moverInvasores() {
       for(int a = 0; a < columnInv; a++){ //Cambio de sentido de los Invasores
          for(int b = 0; b < filaInv; b++){
             if(!puedeMoverNaveH(flotaInvasora[a][b], sentido)){
                for(int i = 0; i < columnInv; i++){ //antes del cambio de sentido baja la flota de
invasores
                   for(int j = 0; j < filaInv; j++){
                      flotaInvasora[i][j].moverVertical(1);
                      if((flotaInvasora[i][j].getCoordenadaY() == alto - 1) && flotaInvasora[i]
[j].getEstado() == 1)
                      { // los invasores llegan a la tierra
                         System.out.print("la flota invasora a llegado a la tierra...");
                         juegoFin = true;
                      }
```

```
                }
                sentido = -sentido;
            }
        }
    }
}

for(int i = 0; i < columnInv; i++){ //comprobacion que no colisiona invasor con defensor
    for(int h = 0; h < filaInv; h++){
        if(flotaInvasora[i][h].getEstado() == 1){
            if(flotaInvasora[i][h].getCoordenadaY() == alto - 4 && //Nave invasora colisiona nave
defensora
                (flotaInvasora[i][h].getCoordenadaX() == defensor.getCoordenadaX())){
                juegoFin = true;
                System.out.print("Has sido alcazado por un kamikaze");
            }
        }
        flotaInvasora[i][h].moverHorizontal(sentido); //Movimiento horizontal de los invasores
    }
}
}

/**
 * Metodo de comprobacion de impacto bala/nave o nave/nave
 * @param bala. proyectil
 * @param nave.
 */
private void hayImpacto(Balas bala, Nave nave) {
    if(bala.getEstado() == 1){
        int x = bala.getCoordenadaX();
        int y = bala.getCoordenadaY();
        int t = nave.getCoordenadaX();
        int z = nave.getCoordenadaY();
        if(((y == z) &&((x == t) ||(x == t - 1) ||(x == t + 1))) ||     //condicion para que el disparo
            ((x == t) &&(y == z - 1)))                          //alcance la nave: bien la coordenda (x,y)
            {                                       //bien cualquiera de las tres adyacentes
            nave.naveAlcanzada();
            bala.naveAlcanzada();
            nave.setEstado(0);
            if(bala instanceof Disparo){ // si alcanza una nave invasora la elimina (cambio de
estado)
                hayDisparo = false;
                cInvasores--;
                if(cInvasores == 0){
                    System.out.println("Has eliminado la flota invasora, salvando a la humanidad
de un destino incierto");
                    juegoFin = true;
                }
            }
            else if(bala instanceof Bomba){ // si la bomba alcanza al defensor fin del juego
                System.out.println("Game Over");
                defensor.naveAlcanzada();
```

```
                    juegoFin = true;
                }
            }
        }
    }

    /**
     * restriccion de movimientos de las naves
     * @return boolean
     */
    private boolean puedeMoverNaveH(Nave nave, int mov) {
        boolean mueve = true;
        if((nave.getEstado() == 1) &&(nave.getCoordenadaX() + mov > ancho - 2 ||
nave.getCoordenadaX() + mov <= 0)){
            mueve = false;
        }
        return mueve;
    }

    /**
     * Genera un numero aletorio
     * dentro del rango 0-n
     * @param n
     * @return devuelve un entero aleatorio
     */
    private int nAleatorio(int n) {
        Random rnd = new Random();
        int z = rnd.nextInt(n);
        return z;
    }

    //metodos publicos
    /**
     * Mecánica automática del juego
     * Movimiento de los invasores
     * Movimiento de los disparos.
     */
    public void movimientosJuego() {
        moverInvasores(); // Bucle del movimiento de los Invasores del espacio
        if(disparoDef != null) { //Disparo defensor
            disparoDef.moverVertical(-1);
            for(int a = 0; a < columnInv; a++){
                for(int b = 0; b < filaInv; b++){
                    if(flotaInvasora[a][b].getEstado() == 1){//solo para los que estan vivos
                        hayImpacto(disparoDef, flotaInvasora[a][b]);
                    }
                }
            }
            if(disparoDef.getCoordenadaY() < 0){// disparo sale del tablero
                disparoDef = null;
                hayDisparo = false;
            }
```

```
        }

        if(!hayBomba()){
              disparaInvasor();
        }else if(bombaInvasor != null)
        { //Disparo Invasor
           bombaInvasor.moverVertical(1);
           hayImpacto(bombaInvasor, defensor);
           if(bombaInvasor.getCoordenadaY() > alto)// disparo sale del tablero
           {
              bombaInvasor = null;
              hayBombaInvasor = false;
           }
        }
        repaint();
    }

    /**
     * Desplazamiento horizontal del defensor
     * @param x. desplazamiento
     */
    public void movHorizontal(int x) {
        if(puedeMoverNaveH(defensor, x)){
           defensor.moverHorizontal(x);
           repaint();
        }
    }

    /**
     * Llamada para crear
     * el disparo del defensor
     */
    public void disparaDefensor() {//crea el diasparo defensor
        Coordenada posicion = new Coordenada(defensor.getCoordenadaX(),
defensor.getCoordenadaY());//consulta posicion del defensor
        disparoDef = new Disparo(posicion);
        hayDisparo = true;
    }

    /**
     * Llamada para crear el disparo
     * de los invasores de forma aleatoria.
     * probabilidad de  disparo
     * 1/10 cada 1/200 ms
     */
    public void disparaInvasor() { //crea las bombas de los invasores de forma aleatoria
        Random rnd = new Random();
        int z = rnd.nextInt(9);
        if(z == 4){
           if(bombaInvasor == null){
              int c = nAleatorio(columnInv); //fila aleatoria
              int f = nAleatorio(filaInv);
```

```
            if(flotaInvasora[c][f].getEstado() == 1){ //consulta posicion del Invasor.
                Coordenada posicionInvasor = new Coordenada(flotaInvasora[c]
[f].getCoordenadaX() + 1, flotaInvasora[c][f].getCoordenadaY() + 1);
                bombaInvasor = new Bomba(posicionInvasor);
                hayBombaInvasor = true;
              }
            }
          }
  }

  // Consultas
  /**
   * Para que el contenedor de la ventana estableza las dimensiones
   *
   * @return dimension, que necesita el Panel
   */
  public Dimension getPreferredSize() {
      return new Dimension(ancho * escala, alto * escala);
  }

  /**
   * Controla el bucle de juego
   * @return boolean
   */
  public boolean juegoFin() {
      return juegoFin;
  }

  /**
   * Consulta para que no
   * exista mas de un disparo a la vez
   */
  public boolean hayDisparo() {
      return hayDisparo;
  }

  /**
   * Consulta para que no ex
   * que no haya mas de una bomba a la vez
   */
  public boolean hayBomba() {
      return hayBombaInvasor;
  }

  /**
   * Pinta el tablero, las naves y los disparos
   * @param g .contexto grafico que dibuja el componente.
   */
  public void paintComponent(Graphics g) {
      g.setColor(Color.black); //pinta el tablero
      for(int i = 0; i < ancho; i++){
         for(int j = 0; j < alto; j++){
```

```
                g.fill3DRect(i * escala, j * escala, escala, escala, true);
            }
        }

        defensor.pintar(g); //pinta el defensor

        if(bombaInvasor != null){
            bombaInvasor.pintar(g);
        }

        for(int i = 0; i < columnInv; i++){ //pinta los invasores del espacio
            for(int j = 0; j < filaInv; j++){
                if(flotaInvasora[i][j].getEstado() == 1){
                    flotaInvasora[i][j].pintar(g);
                }
            }
        }

        if(disparoDef != null){//pinta el disparo defensor
            if(disparoDef.getEstado() == 1){
                disparoDef.pintar(g);
            }
        }
    }

}
```