```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;

public class Skeet extends Applet implements Runnable
{
  // Skeet shooting demo
  // Show motion under gravity, simple collision detection

  Sprite sprites[] = new Sprite[8];
  // Define constants for the array slots
  public static final int DISK = 0;
  public static final int BULLET = 1;
  public static final int FRAG_BASE = 2;

  Disk target;
  Bullet bullet;
  Gun gun;

  float hit_limit = (float)(Disk.RADIUS+Bullet.RADIUS);
  static int launch_delay = 0;

  // set applet size
  public static final int WIDTH = 500;
  public static final int HEIGHT = 400;


  Thread anim = null;

  Image offscreen;
  Graphics offgr;

  public void init()
  {
    // Fill in the sprites array
    // Notice I also create references to the Disk
    // and Bullet which retain the base classes.
    sprites[DISK] = target = new Disk();
    sprites[BULLET] = bullet = new Bullet();
    for (int i=0; i<6; i++)
        sprites[FRAG_BASE+i] = new Frag(i*60, 60);
    // create the gun
    gun = new Gun(bullet);

    addMouseListener(gun);
    addMouseMotionListener(gun);

    // Set up double buffer objects
    offscreen = createImage(WIDTH, HEIGHT);
    offgr = offscreen.getGraphics();
  }

  public void update(Graphics g)
  {
        paint(offgr);
        g.drawImage(offscreen,0,0,null);
  }

  public void paint(Graphics g)
  {
        g.setColor(Color.cyan);
        g.fillRect(0,0,WIDTH,HEIGHT);

        gun.paint(g);
        for (int i=0; i<sprites.length; i++)
                sprites[i].paint(g);
  }

  public void start()
  {
        anim = new Thread(this);
        anim.start();
  }

  public void stop()
  {
        anim = null;
  }

  public void run()
  {
    while (anim != null)
    {
      for (int i = 0; i < sprites.length; i++)
        sprites[i].updateSprite();
      // Check for collisions here
      if (bullet.isActive() && target.isCloserThan(bullet, hit_limit))
      {
        bullet.suspend();
        // We'll pass the sprites array containing the Frag objects
        // to the target and let it set the appropriate position/velocity
        // for the pieces.
        target.hit(sprites, FRAG_BASE);
        launch_delay = 25; // Set a counter to wait a bit before the next disk
      }
      // Check if we're ready for a new Disk
      launch();
      repaint();
      try {
        Thread.sleep(40);
      } catch(InterruptedException e) {}
    }
  }

  void launch()
  {
        // If target is not active, reset and resume target.
        if (!target.isActive() && (--launch_delay < 0))
        {
                target.setPosition(500f,100f);
                target.setVelocity(-20.0f, -18.0f);
                target.resume();
        }
  }
}
```

```
class Sprite
{
  float x,y;
  float dx,dy;
  public static final float GRAVITY = 2.0f;

  boolean active=false, visible=false;

  void updatePosition()
  {
        x += dx;
        y += dy;

        dy += GRAVITY;
  }

  void setPosition(float a, float b)
  {
        x = a; y = b;
  }

  void setVelocity(float a, float b)
  {
    dx = a; dy = b;
  }

  boolean isCloserThan(Sprite s, float dist)
  {
        // Return true if Sprite s is closer to
        // the current Sprite (this) than specified
        // distance dist.
    float dx = x-s.x;
    float dy = y-s.y;
    return dx*dx+dy*dy < dist*dist;
  }

  boolean isActive()
  {
        return active;
  }

  void suspend()
  {
    active = false; visible = false;
  }

  void resume()
  {
    active = true; visible = true;
  }

  void updateSprite() {}
  void paint(Graphics g) {}
}
```

```
class Disk extends Sprite
{
  Color color = Color.red;
  public static final int RADIUS = 20;

  void updateSprite()
  {
    if (active)
    {
      updatePosition();
      // If we've dropped off the screen, suspend the Disk
          if (y > Skeet.HEIGHT)
        suspend();
    }
  }

  void hit(Sprite sprites[], int base)
  {
    int i;

    // turn off the disk and all of the frags
    // (and set each frag to the current disk position
        suspend();
        for (i = 0; i < 6; i++)
        {
                sprites[base + i].suspend();
                sprites[base + i].setPosition(x,y);
        }
        // Now, slightly modify the velocity for each frag
        // so that the pieces will scatter.
        // We take advantage of the fact that we control everything
        // to slow down the pieces a bit so it looks cooler!
        float slow_dx = dx/3.0f;
        float slow_dy = dy/2.0f;
        float scale = 2.0f; // this adjusts how fast all the pieces scatter
        sprites[base + 0].setVelocity(slow_dx+1.2f*scale,slow_dy-0.5f*scale);
        sprites[base + 1].setVelocity(slow_dx+0.0f*scale,slow_dy-1.8f*scale);
        sprites[base + 2].setVelocity(slow_dx-2.0f*scale,slow_dy-1.5f*scale);
        sprites[base + 3].setVelocity(slow_dx-1.0f*scale,slow_dy+0.0f*scale);
        sprites[base + 4].setVelocity(slow_dx+0.0f*scale,slow_dy+0.5f*scale);
        sprites[base + 5].setVelocity(slow_dx+1.5f*scale,slow_dy+0.5f*scale);
        for (i = 0; i < 6; i++)
          sprites[base + i].resume();
  }

  void paint(Graphics g)
  {
    if (visible)
    {
      g.setColor(color);
      g.fillOval((int)x-RADIUS, (int)y-RADIUS, 2*RADIUS, 2*RADIUS);
    }
  }
}
```

```
class Frag extends Sprite
{
  Color color = Color.red;
  public static final int RADIUS = 20;
  int start_angle, arc_angle;

  Frag(int s, int a)
  {
        start_angle = s;
        arc_angle = a;
  }

  void updateSprite()
  {
    if (active)
    {
      updatePosition();
      // If we've dropped off the screen, suspend the Frag
          if (y > Skeet.HEIGHT)
        suspend();
    }
  }

  void paint(Graphics g)
  {
    if (visible)
    {
      g.setColor(color);
      g.fillArc((int)x-RADIUS, (int)y-RADIUS, 2*RADIUS, 2*RADIUS, start_angle, arc_ang
le);
    }
  }
}

class Bullet extends Sprite
{
  Color color = Color.black;
  public static final int RADIUS = 4;

  void updateSprite()
  {
    if (active)
    {
      updatePosition();
      // If we've dropped off the screen, suspend the bullet
      if (y > Skeet.HEIGHT)
        suspend();
    }
  }

  void paint(Graphics g)
  {
    if (visible)
    {
      g.setColor(color);
      g.fillOval((int)x-RADIUS, (int)y-RADIUS, 2*RADIUS, 2*RADIUS);
    }
  }
}
```

```
class Gun implements MouseListener, MouseMotionListener
{
  // set up left and right limits on gun movement
  final int x_lo = 5;
  final int x_hi = Skeet.WIDTH-40;

  int x = x_lo;
  final int y = Skeet.HEIGHT-23;

  Color color = Color.black;

  Bullet bullet;

  Gun(Bullet b)
  {
        bullet = b;
  }

  void paint(Graphics g)
  {
        g.setColor(color);
        g.fillRect(x-5,y,10,20);
  }

  void fireBullet()
  {
    bullet.setPosition(x, y);
    bullet.setVelocity(0.0f, -40.0f);
    bullet.resume();
  }

  public void mousePressed(MouseEvent e)
  {
    if (!bullet.isActive())
      fireBullet();
  }
  public void mouseReleased(MouseEvent e){}
  public void mouseClicked(MouseEvent e){}
  public void mouseEntered(MouseEvent e){}
  public void mouseExited(MouseEvent e){}

  public void mouseMoved(MouseEvent e)
  {
    // Move the gun to the current mouse location
    x = e.getX();
    // Check for limits
    if (x < x_lo)
        x = x_lo;
    else if (x > x_hi)
        x = x_hi;
  }
  public void mouseDragged(MouseEvent e){}
}
```