Juego de Rol

Main:   
Crea un objeto de la clase Gestor Principal y luego ejecuta el juego con el método inicializar y ejecuta el método que inicia un bucle repetitivo.

GestorPrincipal:  
Tiene el constructor, donde se le pasa el ancho, el alto y el nombre de la ventana.  
Ademas tiene 4 metodos, iniciarJuego(), iniciarBuclePrincipal(), dibujar() y actualizar().

* **IniciarJuego():** Instancia 3 objetos ademas de poner en true una variable booleana para saber si el juego se esta ejecutando.  
  Esos 3 objetos son SuperficieDibujo, Ventana, y GestoEstados.  
  **SuperficieDibujo():** Acepta 2 parámetros, el ancho y el alto. Es una clase que extiende a Canvas que le va a permitir crear el área de dibujo o captura de acciones del usuario.  
  El constructor recibe entonces el ancho y el alto, además crea una instancia de la clase Raton (va a manejar el mouse y sus acciones)  
  Luego implemente estos métodos heredados de Canvas

// Muestra el mouse

setCursor(raton.obtenerCursor());

// Ignora la orden de repintado que pueda enviar el SO para solo // permitir lo que indique nuestra app  
setIgnoreRepaint(true);

// Creamos la dimensión con los parametros enviados  
setPreferredSize(new Dimension(ancho, alto));

// Agregamos un evento que escuche o capture acciones del   
// teclado.

addKeyListener(GestorControles.*teclado*);

// Permitimos que haga foco en la ventana de nuestra app  
setFocusable(true);

// Solicitamos el foco desde nuestra ventana  
requestFocus();

Dibujar(): Este método es llamado desde el Gestor Principal cuando el buclePrincipal dibuja la ventana. Acepta como parámetro un GestorEstado??  
Primero crea una estrategia de buffers, con esto le decimos a la app con que cantidad de imágenes bufereadas puede trabajar, esto va a mejorar la performance de como se muestran las imágenes.

BufferStrategy buffer = getBufferStrategy();  
if (buffer == null){  
 createBufferStrategy(3);  
 return;  
}

Luego crea una clase grafico del tipo Graphics al cual se le asigna la estrategia de buffer anterior y luego se inicializan algunos parametros de ese objeto grafico.

Graphics graphics = buffer.getDrawGraphics();  
// Define el color de pintado

graphics.setColor(Color.*BLACK*);

// Crea un rectángulo lleno desde la posición 0,0 hasta el ancho // x alto  
graphics.fillRect(0,0, ancho, alto);  
// Llama al dibujar del gestordeEstados y le manda el grafico

gestorEstados.dibujar(graphics);

// Sincroniza la ventana, generalmente se hace automáticamente // pero cuando se trabaja con animaciones es preferible   
// indicarlo, es parte del AWT  
Toolkit.*getDefaultToolkit*().sync();  
// Liberamos el objeto de memoria.  
graphics.dispose();  
// Mostramos el buffer generado.  
buffer.show();

La mayoría es parte de Java, lo único creado por nosotros es el método dibujar del Gestor de Estados.  
Gestor de Estados: El gestor de estados va a indicarle a la AWT que ventana esta activa para mostrar, si el juego, el menú, la presentación, etc.  
Va a tener un arreglo con todos los estados que definamos, cada estado tendrá un método dibujar y otro actualizar, por lo tanto el arreglo será de una interface EstadisJuego.