1. Informe sobre el Código

1.1. Descripción General

El código presentado consta de dos clases en Java: `Punto` y `Circulo`. Estas clases se utilizan para representar puntos en un plano cartesiano y círculos respectivamente. La clase `Circulo` hereda de la clase `Punto`, lo que implica que un objeto de tipo `Circulo` hereda todas las propiedades y métodos de un objeto `Punto`.

1.2. Clase `Punto`

1.2.1. Atributos

- `private double x`: Representa la coordenada x del punto.
- `private double y`: Representa la coordenada y del punto.

1.2.2. Constructor

- `public Punto(double x, double y)`: Inicializa las coordenadas (x, y) del punto con los valores proporcionados.

1.2.3. Métodos

- `public double distancia(Punto otroPunto)`: Calcula la distancia entre el punto actual y otro punto dado utilizando la fórmula de distancia euclidiana.
- `public double getX()`: Obtiene la coordenada x del punto.
- `public double getY()`: Obtiene la coordenada y del punto.
- `public void setY(double y)`: Establece la coordenada y del punto.
- `public void setX(double x)`: Establece la coordenada x del punto.

1.3. Clase `Circulo`

1.3.1. Atributos

- `private double radio`: Representa el radio del círculo.

1.3.2. Constructor

- `public Circulo(double x, double y, double radio)`: Inicializa las coordenadas (x, y) del centro del círculo y su radio. Utiliza la llamada al constructor de la clase base `Punto` mediante `super(x, y)`.

1.3.3. Métodos

- `public double getRadio()`: Obtiene el radio del círculo.
- `public void setRadio(double radio)`: Establece el radio del círculo.

1.4. Ejemplo de Uso

```
\begin{lstlisting}
\begin{lstlisting}[language=Java]
public class EjemploUso {
  public static void main(String[] args) {
    // Crear un punto en el plano cartesiano
    Punto puntoA = new Punto(1.0, 2.0);

    // Obtener las coordenadas del punto
    double coordenadaX = puntoA.getX();
    double coordenadaY = puntoA.getY();

    // Crear un crculo con centro en el puntoA y radio 3.0
    Circulo circuloA = new Circulo(coordenadaX, coordenadaY, 3.0);

    // Obtener el radio del crculo
    double radioCirculoA = circuloA.getRadio();

    // Establecer nuevas coordenadas al puntoA
    puntoA.setX(4.0);
    puntoA.setY(5.0);

    // Calcular la distancia entre el puntoA y el centro del crculoA
    double distancia = puntoA.distancia(new Punto(coordenadaX, coordenadaY));
  }
}
```

En este ejemplo, se muestran instancias de las clases `Punto` y `Circulo`, así como el acceso a sus métodos para obtener y establecer valores, y calcular la distancia entre el punto y el centro del círculo.

2. Clase Soldado

- Los atributos para la clase soldado son:

```
  private String nombre;
  private int fila;
  private int nivelAtaque = random(5);
  private int nivelDefensa = random(5);
  private int columna;
  private int nivelVida;
  private int vidaActual;
  private int velocidad;
  private String actitud;
  private boolean vive;
  private String team;
```

- Cada atributo que lo requiere, tiene sus metodos setters y getters para encapsular la información.
- Se usan 3 constructores sobrecargados que son los siguientes:

```
  public Soldado(String t) {
    team = t;
    velocidad = 0;
    vive = true;
    actitud = "ataque";

  }
  public Soldado(int v, String t) {
```

```
    team = t;
    velocidad = v;
    vive = true;
    actitud = "ataque";
  }
  public Soldado(int v, int nV, String t) {
    team = t;
    vive = true;
    velocidad = v;
    nivelVida = nV;
    actitud = "ataque";
  }
```

- Estos constructores nos servirán cuando vayamos a crear soldados con distintos datos base.
- Adicionalmente tenemos los metodos de accion del soldado:

```
public void atacar() {
  actitud = "ofensiva";
}
public void defender() {
  actitud = "defensiva";
}
public void huir() {
  actitud = "fuga";
  velocidad += 2;
}
public void avanzar() {
  velocidad += 1;
}

public void serAtacado() {
  vidaActual -= 1;
  if(vidaActual == 0) morir();
}
public void morir() {
  vive = false;
}

public void retroceder() {
  if (velocidad > 0) {
    velocidad = 0;
    actitud = "defensiva";
  } else if (velocidad == 0) {
    velocidad = -1;
  }
}
```

- Las acciones cambian los estados de los atributos del soldado, a los que accederemos despues con los metodos accesores:

```
public String getTeam() {
  return team;
}

public void setNombre(String n) {
  nombre = n;
}

public void setFila(int f) {
```

```
  fila = f;
}

public void setColumna(int c) {
  columna = c;
}

public void setNivelVida(int p) {
  nivelVida = p;
}

public String getNombre() {
  return nombre;
}

public int getFila() {
  return fila;
}

public int getColumna() {
  return columna;
}

public int getNivelVida() {
  return nivelVida;
}

public int getNivelAtaque() {
  return nivelAtaque;
}

public int getNivelDefensa() {
  return nivelDefensa;
}

public void setNivelAtaque(int n) {
  nivelAtaque = n;
}

public void setNivelDefensa(int n) {
  nivelDefensa = n;
}

public boolean isLive() {
  return vive;
}
```

- Estos metodos accesores nos servirán para manejar la lógica interna del videojuego
- Adicionalmente tenemos algunos metodos auxiliares que usamos en la misma clase:

```
public String toString() {
  return "Nombre: " + nombre +
  " | Ubicacion: " + fila + ", " + columna +
  " | nivelVida: " + nivelVida +
  " | Estado: " + (vive ? "Vivo" : "Muerto") +
  " | Actitud: "+ actitud +"\n" ;
}
private int random(int n) {
  return (int) (Math.random() * n + 1);
}
```

3. Clase VideoJuego

La clase VideoJuego contiene la logica principal y es la clase que contiene el metodo main, por lo tanto, el que debería llamarse para ejecutar el juego. La estructura de la clase esta dada de la siguiente manera:

- Metodos:
- juegoRapido
- juegoPersonalizado
- crearSoldado
- eliminarSoldado
- clonarSoldado
- modificarSoldado
- compararSoldado
- intercambiarSoldado
- verSoldado
- verEjercito
- sumarNiveles
- jugar
- volver
- main
- fillTable
- addSoldado
- printTable
- printArr
- random
- printMayorNivel
- printPromedioPuntos
- printPuntosAll
- printSoladosOrdenados
- printRankingPointsBubble
- printRankingPointsSelect
- getWinner
- randomWinner
- mover
- atacar

A continuación veremos la explicación de los metodos principales para el funcionamiento.