



**SEP**

**SES**

**TecNM**

---

## **INSTITUTO TECNOLÓGICO DE TOLUCA**

**“PR2 Herencia y Relación de Objetos”**

**PROGRAMACION AVANZADA**

**INGENIERIA EN MECATRONICA**

**PRESENTA:**

<b>ABRAHAM SÁNCHEZ JIMÉNEZ</b>	<b>22280836</b>
<b>EDUARDO HERNÁNDEZ SALVADOR</b>	<b>22280890</b>
<b>DANIEL DE LA CRUZ VELAZQUEZ</b>	<b>22280933</b>
<b>SANTIAGO ROJAS SAAVEDRA 2</b>	<b>2280856</b>

**EQUIPO “NEURONA CIBERNÉTICA”**

**PROFESOR:**

**JESUS AURELIO CASTRO MAGAÑA**

**GRUPO: 190002**

```
package rpg.entities;

import rpg.enums.Stats;

import javax.swing.*;
import java.util.HashMap;

/**
 * Clase abstracta que representa a un personaje en el juego. Typo: In word 'Clase'
 *
 * @author [AbrahamDell]
 */
public abstract class GameCharacter { 10 inheritors  👤 AbrahamDell
    /**
     * Nombre del personaje. Typo: In word 'Nombre'. Typo: In word 'personaje'.
     */
    protected String name;

    /**
     * Estadísticas del personaje. Typo: In word 'Estadísticas'. Typo: In word 'pe
     */
    protected HashMap<Stats, Integer> stats;

    /**
     * Constructor que inicializa el personaje con un nombre y estadísticas vacías.

```

© ElderSlime.java   © DemonLord.java   © Dragon.java   © AncientDragon.java   © ShadowAs

public abstract class GameCharacter { 10 inheritors   AbrahamDell

/\*\*

  \* Constructor que inicializa el personaje con un nombre y estadísticas vacías. Typo: In word 'nombre'.

  \*

  \* @param name Nombre del personaje. Typo: In word 'Nombre'. Typo: In word 'personaje'.

  \*/

public GameCharacter(String name) {   AbrahamDell

  this.name = name;

  this.stats = new HashMap<>();

  initCharacter();

}

/\*\*

  \* Método abstracto que debe ser implementado por las clases que hereden de GameCharacter.

  \* Inicializa las estadísticas del personaje. Typo: In word 'Inicializa'. Typo: In word 'personaje'.

  \*/

protected abstract void initCharacter(); 1 usage   9 implementations   AbrahamDell

/\*\*

  \* Verifica si el personaje está vivo. Typo: In word 'Verifica'. Typo: In word 'personaje'.

  \*

  \* @return True si el personaje tiene HP mayor a 0, false en caso contrario. Typo: In word 'personaje'.

  \*/

public boolean isAlive() { 4 usages   AbrahamDell

  return stats.get(Stats.HP) > 0;

}

ElderSlime.java

DemonLord.java

Dragon.java

AncientDragon.java

ShadowAssas

```

public abstract class GameCharacter { 10 inheritors  AbrahamDell

    /**
     * Ataca a un enemigo. Typo: In word 'Ataca'. Typo: In word 'enemigo'.
     *
     * @param enemy Enemigo a atacar. Typo: In word 'Enemigo'. Typo: In word 'atacar'.
     */
    public void attack(GameCharacter enemy) { 8 overrides  AbrahamDell
        String message = "";
        String enemyName = enemy.getName();
        int damage = this.stats.get(Stats.ATTACK) - enemy.getStats().get(Stats.DEFENSE);
        int newHP = enemy.getStats().get(Stats.HP);
        if (damage > 0) {
            newHP = enemy.getStats().get(Stats.HP) - damage;
            enemy.getStats().put(Stats.HP, newHP);
            message += String.format("%s ataca a %s por %d de daño! A %s le quedan %d HP.%n", Ty
                this.name, enemyName, damage, enemyName, newHP);
        } else {
            message += String.format("%s ataca a %s pero no hace daño! A %s le quedan %d HP.%n",
                this.name, enemyName, enemyName, newHP);
        }
        JOptionPane.showMessageDialog(parentComponent: null, message);
    }
}

```

```
© ElderSlime.java    © DemonLord.java    © Dra

public abstract class GameCharacter { 10 inheritors

    /**
     * Obtiene el nombre del personaje. Typo: In
     *
     * @return Nombre del personaje. Typo: In wo
     */
    public String getName() {  ⚡ AbrahamDell
        return name;
    }

    /**
     * Obtiene las estadísticas del personaje.
     *
     * @return Estadísticas del personaje. Typo
     */
    public HashMap<Stats, Integer> getStats() {
        return stats;
    }
}
```

me.java

© DemonLord.java

© Dragon.java

© AncientDrag

```
package rpg.entities;
```

```
import rpg.enums.Stats;
```

```
/**
```

```
 * Clase que representa al jugador. Typo: In word 'Clase'.
```

```
 *
```

```
 * @author [AbrahamDell]
```

```
 */
```

```
public class Player extends GameCharacter { 3 usages  ⓘ AbrahamDell
```

```
    /**
```



```
    * Constructor que inicializa el jugador con un nombre.
```

```
    *
```

```
    * @param name Nombre del jugador. Typo: In word 'Nombre'.
```

```
    */
```

```
    public Player(String name) { 1 usage  ⓘ AbrahamDell
```

```
        super(name);
```

```
    }
```

```
    /**
```

```
    * Inicializa las estadísticas del jugador. Typo: In word 'Inicializa'.
```

```
    */
```

```
    @Override 1 usage  ⓘ AbrahamDell
```

```
    protected void initCharacter() {
```

```
        this.stats.put(Stats.MAX_HP, 100);
```

lime.java

© DemonLord.java

© Dragon.java

```
public class Player extends GameCharacter { 3 usa
    *
    * @param name Nombre del jugador. Typo: In
    */
    public Player(String name) { 1 usage  👤 Abraha
        super(name);
    }

    /**
     * Inicializa las estadísticas del jugador.
     */
    @Override 1 usage  👤 AbrahamDell
    protected void initCharacter() {
        this.stats.put(Stats.MAX_HP, 100);
        this.stats.put(Stats.HP, 100);
        this.stats.put(Stats.ATTACK, 35);
        this.stats.put(Stats.DEFENSE, 15);
    }
}
```

```
package rpg.enums;
```

```
/**
```

```
 * Enumeración que representa los tipos de enemigos.
```

```
 *
```

```
 * @author [AbrahamDell]
```

```
 */
```

```
public enum EnemyType { 20 usages  👤 AbrahamDell
```

```
    /**
```

```
     * Enemigo básico. Typo: In word 'Enemigo'. Typo
```

```
     */
```

```
    BASIC, 2 usages
```

```
    /**
```

```
     * Enemigo medio. Typo: In word 'Enemigo'. Typo
```

```
     */
```

```
    MEDIUM, 2 usages
```

```
    /**
```

```
     * Enemigo final. Typo: In word 'Enemigo'.
```

```
     */
```

```
    FINAL, 2 usages
```

```
    /**
```

```
     * Enemigo secreto. Typo: In word 'Enemigo'. Ty
```

```
     */
```

```
    SECRET 2 usages
```

```
 }
```



ShadowAssassin.java

GameCharacter.java

Player.java

```
package rpg.enums;
```

```
/**
```

```
 * Enumeración que representa las estadísticas de los personajes.
```

```
 *
```

```
 * @author [AbrahamDell]
```

```
 */
```

```
public enum Stats {  ⓘ AbrahamDell +1
```

```
    /**
```

```
     * Puntos de vida máximos. Typo: In word 'Puntos'. Typo: In
```

```
     */
```

```
    MAX_HP, 9 usages
```

```
    /**
```

```
     * Puntos de vida actuales. Typo: In word 'Puntos'. Typo: In
```

```
     */
```

```
    HP,
```

```
    /**
```

```
     * Ataque. Typo: In word 'Ataque'.
```

```
     */
```

```
    ATTACK,
```

```
    /**
```

```
     * Defensa. Typo: In word 'Defensa'.
```

```
     */
```

```
    DEFENSE
```

```
}
```

agon.java

© ShadowAssassin.java


© GameCharacter.java

© Player.java

```
package rpg.utils;

import rpg.entities.enemies.Enemy;
import rpg.entities.enemies.basic.BasicSlime;
import rpg.entities.enemies.basic.RookieGoblin;
import rpg.entities.enemies.medium.EliteGoblin;
import rpg.entities.enemies.medium.ElderSlime;
import rpg.entities.enemies.finals.Dragon;
import rpg.entities.enemies.finals.DemonLord;
import rpg.entities.enemies.secret.AncientDragon;
import rpg.entities.enemies.secret.ShadowAssassin;

import javax.swing.*; Unused import statement.
import java.util.Random;

/**
 * Clase que proporciona métodos para generar números aleatorios.
 * 
 * @author [AbrahamDell]
 */
public class Randomize { 2 usages  ⓘ AbrahamDell
    /**
     * Generador de números aleatorios. Typo: In word 'Generador'.
     */
    private static Random random = new Random(); 2 usages F
```

n.java

© ShadowAssassin.java

© GameCharacter.j

```
public class Randomize { 2 usages  AbrahamDell  
    private static Random random = new Random();  
  
    /**  
     * Obtiene un booleano aleatorio. Typo: In w  
     *  
     * @return Booleano aleatorio. Typo: In word  
     */  
    public static boolean getRandomBoolean() { n  
        return random.nextBoolean();  
    }  
  
    /**  
     * Obtiene un enemigo aleatorio. Typo: In wo  
     *  
     * @return Enemigo aleatorio. Typo: In word  
     */  
    public static Enemy getRandomEnemy() { 1 usag  
        Enemy[] enemies = {  
            new BasicSlime(),  
            new RookieGoblin(),  
            new EliteGoblin(),  
            new ElderSlime(),  
            new Dragon(),  
            new DemonLord(),
```

on.java    © ShadowAssassin.java    © GameCharacter.java

```
public class Randomize { 2 usages    AbrahamDell  
    /**  
     * Obtiene un enemigo aleatorio.    Typo: In word 'Obt  
     *  
     * @return Enemigo aleatorio.    Typo: In word 'Enemig  
     */  
    public static Enemy getRandomEnemy() { 1 usage    Ab  
        Enemy[] enemies = {  
            new BasicSlime(),  
            new RookieGoblin(),  
            new EliteGoblin(),  
            new ElderSlime(),  
            new Dragon(),  
            new DemonLord(),  
            new AncientDragon(),  
            new ShadowAssassin()  
        };  
        return enemies[random.nextInt(enemies.length)];  
    }  
}
```

```

.java      © GameCharacter.java      © Player.java      © Enemy.java

package rpg.entities.enemies.basic;
import javax.swing.*;

import rpg.entities.GameCharacter;
import rpg.entities.enemies.Enemy;
import rpg.enums.EnemyType;
import rpg.enums.Stats;

/**
 * Clase que representa a un Slime básico. Typo: In w
 *
 * @author [AbrahamDell]
 */
public class BasicSlime extends Enemy { 2 usages  ⓘ Ab
    /**
     * Constructor que inicializa el Slime básico. Ty
     */
    public BasicSlime() { 1 usage  ⓘ AbrahamDell
        super( name: "Basic Slime", EnemyType.BASIC);
    }

    /**
     * Obtiene el botín del Slime básico. Typo: In w
     */
    @Override no usages  ⓘ AbrahamDell

```

```

java  GameCharacter.java  Player.java  EnemyType.java  Stats.java  Randomize.java  BasicSlime.java x
public class BasicSlime extends Enemy { 2 usages  AbrahamDell
    @Override no usages  AbrahamDell
    public void getLoot() {
        JOptionPane.showMessageDialog( parentComponent: null, message: "El Slime básico suelta una pequeña cantidad de oro.");
    }

    /**
     * Inicializa las estadísticas del Slime básico. Typo: In word 'Inicializa'. Typo: In word 'estadísticas'. Typo: In word
     */
    @Override 1 usage  AbrahamDell
    protected void initCharacter() {
        this.stats.put(Stats.MAX_HP, 20);
        this.stats.put(Stats.HP, 20);
        this.stats.put(Stats.ATTACK, 2);
        this.stats.put(Stats.DEFENSE, 1);
    }

    /**
     * Ataca a un enemigo. Typo: In word 'Ataca'. Typo: In word 'enemigo'.
     *
     * @param enemy Enemigo a atacar. Typo: In word 'Enemigo'. Typo: In word 'atacar'.
     */
    @Override  AbrahamDell
    public void attack(GameCharacter enemy) {
        String enemyName = enemy.getName();
        // ...
    }
}

```

```

java  GameCharacter.java  Player.java  EnemyType.java  Stats.java  Randomize.java  BasicSlime.java x
public class BasicSlime extends Enemy { 2 usages  AbrahamDell
    protected void initCharacter() {
        this.stats.put(Stats.DEFENSE, 1);
    }

    /**
     * Ataca a un enemigo. Typo: In word 'Ataca'. Typo: In word 'enemigo'.
     *
     * @param enemy Enemigo a atacar. Typo: In word 'Enemigo'. Typo: In word 'atacar'.
     */
    @Override  AbrahamDell
    public void attack(GameCharacter enemy) {
        String enemyName = enemy.getName();
        int damage = this.stats.get(Stats.ATTACK) - enemy.getStats().get(Stats.DEFENSE);
        int newHP = enemy.getStats().get(Stats.HP) - damage;
        enemy.getStats().put(Stats.HP, newHP);
        JOptionPane.showMessageDialog( parentComponent: null, String.format("%s ataca a %s por %d de daño! A %s le quedan %d HP.%n",
            this.name, enemyName, damage, enemyName, newHP));
    }
}

```

GameCharacter.java

© Player.java

ⓔ EnemyType.java

```
package rpg.entities.enemies.basic;
import javax.swing.*;

import rpg.entities.GameCharacter;
import rpg.entities.enemies.Enemy;
import rpg.enums.EnemyType;
import rpg.enums.Stats;

/**
 * Clase que representa a un Goblin novato. Typo: In w
 *
 * @author [AbrahamDell]
 */
public class RookieGoblin extends Enemy { 2 usages  Ⓜ Ab
    /**
     * Constructor que inicializa el Goblin novato. Ty
     */
    public RookieGoblin() { 1 usage  Ⓜ AbrahamDell
        super( name: "Rookie Goblin", EnemyType.BASIC);
    }

    /**
     * Obtiene el botín del Goblin novato. Typo: In wo
     */
    @Override no usages  Ⓜ AbrahamDell
```

```

GameCharacter.java  Player.java  EnemyType.java  Stats.java  Randomize.java  BasicSlime.java  Rookie
public class RookieGoblin extends Enemy { 2 usages  AbrahamDell

    @Override no usages  AbrahamDell
    public void getLoot() {
        JOptionPane.showMessageDialog(parentComponent: null, message: "El Goblin Novato suelta una espada oxidada.");
    }

    /**
     * Inicializa las estadísticas del Goblin novato. Typo: In word 'Inicializa'. Typo: In word 'estadísticas'.
     */
    @Override 1 usage  AbrahamDell
    protected void initCharacter() {
        this.stats.put(Stats.MAX_HP, 15);
        this.stats.put(Stats.HP, 15);
        this.stats.put(Stats.ATTACK, 3);
        this.stats.put(Stats.DEFENSE, 1);
    }

    /**
     * Ataca a un enemigo. Typo: In word 'Ataca'. Typo: In word 'enemigo'.
     *
     * @param enemy Enemigo a atacar. Typo: In word 'Enemigo'. Typo: In word 'atacar'.
     */
    @Override  AbrahamDell
    public void attack(GameCharacter enemy) {
        String enemyName = enemy.getName();

```

```

GameCharacter.java  Player.java  EnemyType.java  Stats.java  Randomize.java  BasicSlime.java  RookieGoblin.java x
public class RookieGoblin extends Enemy { 2 usages  AbrahamDell
    protected void initCharacter() {
        this.stats.put(Stats.DEFENSE, 1);
    }

    /**
     * Ataca a un enemigo. Typo: In word 'Ataca'. Typo: In word 'enemigo'.
     *
     * @param enemy Enemigo a atacar. Typo: In word 'Enemigo'. Typo: In word 'atacar'.
     */
    @Override  AbrahamDell
    public void attack(GameCharacter enemy) {
        String enemyName = enemy.getName();
        int damage = this.stats.get(Stats.ATTACK) - enemy.getStats().get(Stats.DEFENSE);
        int newHP = enemy.getStats().get(Stats.HP) - damage;
        enemy.getStats().put(Stats.HP, newHP);
        JOptionPane.showMessageDialog(parentComponent: null, String.format("%s ataca a %s por %d de daño! A %s le quedan %d HP.%n",
            this.name, enemyName, damage, enemyName, newHP));
    }
}

```



Player.java EnemyType.java Stats.java

```

package rpg.entities.enemies.finals;
import javax.swing.*;

import rpg.entities.GameCharacter;
import rpg.entities.enemies.Enemy;
import rpg.enums.EnemyType;
import rpg.enums.Stats;

/**
 * Clase que representa al Señor Demonio. Typo: In w
 *
 * @author [AbrahamDell]
 */
public class DemonLord extends Enemy { 2 usages ⓘ Ab
    /**
     * Constructor que inicializa al Señor Demonio.
     */
    public DemonLord() { 1 usage ⓘ AbrahamDell
        super(name: "Demon Lord", EnemyType.FINAL);
    }

    /**
     * Obtiene el botín del Señor Demonio. Typo: In
     */
    @Override no usages ⓘ AbrahamDell

```

```

Player.java  EnemyType.java  Stats.java  Randomize.java  BasicSlime.java  RookieGoblin.java  DemonLord.java
public class DemonLord extends Enemy { 2 usages  AbrahamDell

    @Override no usages  AbrahamDell
    public void getLoot() {
        JOptionPane.showMessageDialog( parentComponent: null, message: "El Señor Demonio suelta un artefacto poderoso.");
    }

    /**
     * Inicializa las estadísticas del Señor Demonio. Typo: In word 'Inicializa'. Typo: In word 'estadísticas'. Typo:
     */
    @Override 1 usage  AbrahamDell
    protected void initCharacter() {
        this.stats.put(Stats.MAX_HP, 120);
        this.stats.put(Stats.HP, 120);
        this.stats.put(Stats.ATTACK, 20);
        this.stats.put(Stats.DEFENSE, 15);
    }

    /**
     * Ataca a un enemigo. Typo: In word 'Ataca'. Typo: In word 'enemigo'.
     *
     * @param enemy Enemigo a atacar. Typo: In word 'Enemigo'. Typo: In word 'atacar'.
     */
    @Override  AbrahamDell
    public void attack(GameCharacter enemy) {
        String enemyName = enemy.getName();

```

```

Player.java  EnemyType.java  Stats.java  Randomize.java  BasicSlime.java  RookieGoblin.java  DemonLord.java x
public class DemonLord extends Enemy { 2 usages  AbrahamDell ✓
    protected void initCharacter() {
        this.stats.put(Stats.DEFENSE, 15);
    }

    /**
     * Ataca a un enemigo. Typo: In word 'Ataca'. Typo: In word 'enemigo'.
     *
     * @param enemy Enemigo a atacar. Typo: In word 'Enemigo'. Typo: In word 'atacar'.
     */
    @Override  AbrahamDell
    public void attack(GameCharacter enemy) {
        String enemyName = enemy.getName();
        int damage = this.stats.get(Stats.ATTACK) - enemy.getStats().get(Stats.DEFENSE);
        int newHP = enemy.getStats().get(Stats.HP) - damage;
        enemy.getStats().put(Stats.HP, newHP);
        JOptionPane.showMessageDialog( parentComponent: null, String.format("%s ataca a %s por %d de daño! A %s le quedan %d HP.%n",
            this.name, enemyName, damage, enemyName, newHP));
    }
}

```

nemyType.java

Stats.java

Randomize.java

```
package rpg.entities.enemies.finals;
import javax.swing.*;

import rpg.entities.GameCharacter;
import rpg.entities.enemies.Enemy;
import rpg.enums.EnemyType;
import rpg.enums.Stats;

/**
 * Clase que representa al Dragón. Typo: In word
 *
 * @author [AbrahamDell]
 */
public class Dragon extends Enemy { 2 usages ⓘ A
    /**
     * Constructor que inicializa al Dragón. Typo: In word
     */
    public Dragon() { 1 usage ⓘ AbrahamDell
        super( name: "Dragon", EnemyType.FINAL);
    }

    /**
     * Obtiene el botín del Dragón. Typo: In word
     */
    @Override no usages ⓘ AbrahamDell
```

emyType.java   Stats.java   Randomize.java   BasicSlime.java   RookieGoblin.java   DemonLord.java   **Dragon.java**

```

public class Dragon extends Enemy { 2 usages  AbrahamDell

    @Override no usages  AbrahamDell
    public void getLoot() {
        JOptionPane.showMessageDialog( parentComponent: null, message: "El Dragón suelta una espada legendaria.");
    }

    /**
     * Inicializa las estadísticas del Dragón. Typo: In word 'Inicializa'. Typo: In word 'estadísticas'.
     */
    @Override 1 usage  AbrahamDell
    protected void initCharacter() {
        this.stats.put(Stats.MAX_HP, 100);
        this.stats.put(Stats.HP, 100);
        this.stats.put(Stats.ATTACK, 15);
        this.stats.put(Stats.DEFENSE, 10);
    }

    /**
     * Ataca a un enemigo. Typo: In word 'Ataca'. Typo: In word 'enemigo'.
     *
     * @param enemy Enemigo a atacar. Typo: In word 'Enemigo'. Typo: In word 'atacar'.
     */
    @Override  AbrahamDell
    public void attack(GameCharacter enemy) {
        String enemyName = enemy.getName();
    }

```

emyType.java   Stats.java   Randomize.java   BasicSlime.java   RookieGoblin.java   DemonLord.java   **Dragon.java** ×

```

public class Dragon extends Enemy { 2 usages  AbrahamDell 17
    protected void initCharacter() {
        this.stats.put(Stats.DEFENSE, 10);
    }

    /**
     * Ataca a un enemigo. Typo: In word 'Ataca'. Typo: In word 'enemigo'.
     *
     * @param enemy Enemigo a atacar. Typo: In word 'Enemigo'. Typo: In word 'atacar'.
     */
    @Override  AbrahamDell
    public void attack(GameCharacter enemy) {
        String enemyName = enemy.getName();
        int damage = this.stats.get(Stats.ATTACK) - enemy.getStats().get(Stats.DEFENSE);
        int newHP = enemy.getStats().get(Stats.HP) - damage;
        enemy.getStats().put(Stats.HP, newHP);
        JOptionPane.showMessageDialog( parentComponent: null, String.format("%s ataca a %s por %d de daño! A %s le quedan %d HP.%n",
            this.name, enemyName, damage, enemyName, newHP));
    }
}

```

```
Stats.java Randomize.java BasicSlime.java

package rpg.entities.enemies.medium;
import javax.swing.*;

import rpg.entities.GameCharacter;
import rpg.entities.enemies.Enemy;
import rpg.enums.EnemyType;
import rpg.enums.Stats;

/**
 * Clase que representa a un Slime anciano. Typo: In
 *
 * @author [AbrahamDell]
 */
public class ElderSlime extends Enemy { 2 usages  ⓘ Abr
    /**
     * Constructor que inicializa el Slime anciano. T
     */
    public ElderSlime() { 1 usage  ⓘ AbrahamDell
        super(name: "Elder Slime", EnemyType.MEDIUM);
    }

    /**
     * Obtiene el botín del Slime anciano. Typo: In w
     */
    @Override no usages  ⓘ AbrahamDell
```

Stats.java Randomize.java BasicSlime.java RookieGoblin.java DemonLord.java Dragon.java

```

public class ElderSlime extends Enemy { 2 usages AbrahamDell

    @Override no usages AbrahamDell
    public void getLoot() {
        JOptionPane.showMessageDialog(parentComponent: null, message: "El Slime Anciano suelta un amuleto mágico.");
    }

    /**
     * Inicializa las estadísticas del Slime anciano. Typo: In word 'Inicializa'. Typo: In word 'estadísticas'. Typo: In word 'Slime'. Typo: In word 'anciano'.
     */
    @Override 1 usage AbrahamDell
    protected void initCharacter() {
        this.stats.put(Stats.MAX_HP, 40);
        this.stats.put(Stats.HP, 40);
        this.stats.put(Stats.ATTACK, 6);
        this.stats.put(Stats.DEFENSE, 3);
    }

    /**
     * Ataca a un enemigo. Typo: In word 'Ataca'. Typo: In word 'enemigo'.
     *
     * @param enemy Enemigo a atacar. Typo: In word 'Enemigo'. Typo: In word 'atacar'.
     */
    @Override AbrahamDell
    public void attack(GameCharacter enemy) {
        String enemyName = enemy.getName();
    }

```

Stats.java Randomize.java BasicSlime.java RookieGoblin.java DemonLord.java Dragon.java ElderSlime.java

```

public class ElderSlime extends Enemy { 2 usages AbrahamDell

    protected void initCharacter() {
        this.stats.put(Stats.DEFENSE, 3);
    }

    /**
     * Ataca a un enemigo. Typo: In word 'Ataca'. Typo: In word 'enemigo'.
     *
     * @param enemy Enemigo a atacar. Typo: In word 'Enemigo'. Typo: In word 'atacar'.
     */
    @Override AbrahamDell
    public void attack(GameCharacter enemy) {
        String enemyName = enemy.getName();
        int damage = this.stats.get(Stats.ATTACK) - enemy.getStats().get(Stats.DEFENSE);
        int newHP = enemy.getStats().get(Stats.HP) - damage;
        enemy.getStats().put(Stats.HP, newHP);
        JOptionPane.showMessageDialog(parentComponent: null, String.format("%s ataca a %s por %d de daño! A %s le quedan %d HP.%n",
            this.name, enemyName, damage, enemyName, newHP));
    }

```

```
Randomize.java    © BasicSlime.java    © RookieGoblin.java

package rpg.entities.enemies.medium;
import javax.swing.*;

import rpg.entities.GameCharacter;
import rpg.entities.enemies.Enemy;
import rpg.enums.EnemyType;
import rpg.enums.Stats;

/**
 * Clase que representa a un Goblin elite. Typo: In word
 *
 * @author [AbrahamDell]
 */
public class EliteGoblin extends Enemy { 2 usages  👤 AbrahamDell

    /**
     * Constructor que inicializa el Goblin elite. Typo
     */
    public EliteGoblin() { 1 usage  👤 AbrahamDell
        super( name: "Elite Goblin", EnemyType.MEDIUM);
    }

    /**
     * Obtiene el botín del Goblin elite. Typo: In word
     */
    @Override no usages  👤 AbrahamDell
```

```
Randomize.java BasicSlime.java RookieGoblin.java DemonLord.java Dragon.java ElderSlime.java
public class EliteGoblin extends Enemy { 2 usages AbrahamDell
    @Override no usages AbrahamDell
    public void getLoot() {
        JOptionPane.showMessageDialog(parentComponent: null, message: "El Goblin Élite suelta una espada fina.");
    }

    /**
     * Inicializa las estadísticas del Goblin elite. Typo: In word 'Inicializa'. Typo: In word 'estadísticas'.
     */
    @Override 1 usage AbrahamDell
    protected void initCharacter() {
        this.stats.put(Stats.MAX_HP, 30);
        this.stats.put(Stats.HP, 30);
        this.stats.put(Stats.ATTACK, 5);
        this.stats.put(Stats.DEFENSE, 2);
    }

    /**
     * Ataca a un enemigo. Typo: In word 'Ataca'. Typo: In word 'enemigo'.
     *
     * @param enemy Enemigo a atacar. Typo: In word 'Enemigo'. Typo: In word 'atacar'.
     */
    @Override AbrahamDell
    public void attack(GameCharacter enemy) {
        String enemyName = enemy.getName();
        // this.stats.put(Stats.ATTACK, enemy.getStats().get(Stats.DEFENSE));
    }
}
```

```
Randomize.java BasicSlime.java RookieGoblin.java DemonLord.java Dragon.java ElderSlime.java EliteGoblin.java x
public class EliteGoblin extends Enemy { 2 usages AbrahamDell
    protected void initCharacter() {
        this.stats.put(Stats.DEFENSE, 2);
    }

    /**
     * Ataca a un enemigo. Typo: In word 'Ataca'. Typo: In word 'enemigo'.
     *
     * @param enemy Enemigo a atacar. Typo: In word 'Enemigo'. Typo: In word 'atacar'.
     */
    @Override AbrahamDell
    public void attack(GameCharacter enemy) {
        String enemyName = enemy.getName();
        int damage = this.stats.get(Stats.ATTACK) - enemy.getStats().get(Stats.DEFENSE);
        int newHP = enemy.getStats().get(Stats.HP) - damage;
        enemy.getStats().put(Stats.HP, newHP);
        JOptionPane.showMessageDialog(parentComponent: null, String.format("%s ataca a %s por %d de daño! A %s le quedan %d HP.%n",
            this.name, enemyName, damage, enemyName, newHP));
    }
}
```



```
BasicSlime.java  RookieGoblin.java  DemonLord.java

package rpg.entities.enemies.secret;
import javax.swing.*;

import rpg.entities.GameCharacter;
import rpg.entities.enemies.Enemy;
import rpg.enums.EnemyType;
import rpg.enums.Stats;

/**
 * Clase que representa al Dragón antiguo. Typo: In word
 *
 * @author [AbrahamDell]
 */
public class AncientDragon extends Enemy { 2 usages  ⓘ AbrahamDell
    /**
     * Constructor que inicializa al Dragón antiguo. Typo
     */
    public AncientDragon() { 1 usage  ⓘ AbrahamDell
        super(name: "Ancient Dragon", EnemyType.SECRET);
    }

    /**
     * Obtiene el botín del Dragón antiguo. Typo: In word
     */
    @Override no usages  ⓘ AbrahamDell
```

BasicSlime.java    © RookieGoblin.java    © DemonLord.java    © Dragon.java    © ElderSlime.java    © EliteGoblin.java    © AncientDragon.java

```
public class AncientDragon extends Enemy { 2 usages  ⓘ AbrahamDell

    @Override no usages  ⓘ AbrahamDell
    public void getLoot() {
        JOptionPane.showMessageDialog(parentComponent: null, message: "El Dragón Antigo deja caer una espada mítica.");
    }

    /**
     * Inicializa las estadísticas del Dragón antiguo. Typo: In word 'Inicializa'. Typo: In word 'estadísticas'. Typo:
     */
    @Override 1 usage  ⓘ AbrahamDell
    protected void initCharacter() {
        this.stats.put(Stats.MAX_HP, 150);
        this.stats.put(Stats.HP, 150);
        this.stats.put(Stats.ATTACK, 25);
        this.stats.put(Stats.DEFENSE, 20);
    }

    /**
     * Ataca a un enemigo. Typo: In word 'Ataca'. Typo: In word 'enemigo'.
     *
     * @param enemy Enemigo a atacar. Typo: In word 'Enemigo'. Typo: In word 'atacar'.
     */
    @Override ⓘ AbrahamDell
    public void attack(GameCharacter enemy) {
        String enemyName = enemy.getName();
```

BasicSlime.java    © RookieGoblin.java    © DemonLord.java    © Dragon.java    © ElderSlime.java    © EliteGoblin.java    © AncientDragon.java

```
public class AncientDragon extends Enemy { 2 usages  ⓘ AbrahamDell

    protected void initCharacter() {
        this.stats.put(Stats.DEFENSE, 20);
    }

    /**
     * Ataca a un enemigo. Typo: In word 'Ataca'. Typo: In word 'enemigo'.
     *
     * @param enemy Enemigo a atacar. Typo: In word 'Enemigo'. Typo: In word 'atacar'.
     */
    @Override ⓘ AbrahamDell
    public void attack(GameCharacter enemy) {
        String enemyName = enemy.getName();
        int damage = this.stats.get(Stats.ATTACK) - enemy.getStats().get(Stats.DEFENSE);
        int newHP = enemy.getStats().get(Stats.HP) - damage;
        enemy.getStats().put(Stats.HP, newHP);
        JOptionPane.showMessageDialog(parentComponent: null, String.format("%s ataca a %s por %d daño! %s tiene %d HP restantes.%n",
            this.name, enemyName, damage, enemyName, newHP));
    }
}
```

n.java

© DemonLord.java

© Dragon.java

© ElderSlime.java

```
package rpg.entities.enemies.secret;
import javax.swing.*;

import rpg.entities.GameCharacter;
import rpg.entities.enemies.Enemy;
import rpg.enums.EnemyType;
import rpg.enums.Stats;

/**
 * Clase que representa al Asesino de las Sombras. Typo:
 *
 * @author [AbrahamDell]
 */
public class ShadowAssassin extends Enemy { 2 usages  ⓘ AbrahamDell
    /**
     * Constructor que inicializa al Asesino de las Sombras.
     */
    public ShadowAssassin() { 1 usage  ⓘ AbrahamDell
        super( name: "Shadow Assassin", EnemyType.SECRET);
    }

    /**
     * Obtiene el botín del Asesino de las Sombras. Typo:
     */
    @Override no usages  ⓘ AbrahamDell
```

```

n.java    © DemonLord.java    © Dragon.java    © ElderSlime.java    © EliteGoblin.java    © AncientDragon.java    © ShadowAssassin
public class ShadowAssassin extends Enemy { 2 usages  ⚡ AbrahamDell
    @Override no usages  ⚡ AbrahamDell
    public void getLoot() {
        JOptionPane.showMessageDialog( parentComponent: null, message: "El Asesino de las Sombras deja caer una daga rara.");
    }

    /**
     * Inicializa las estadísticas del Asesino de las Sombras. Typo: In word 'Inicializa'. Typo: In word 'estadísticas'. T
     */
    @Override 1 usage  ⚡ AbrahamDell
    protected void initCharacter() {
        this.stats.put(Stats.MAX_HP, 180);
        this.stats.put(Stats.HP, 180);
        this.stats.put(Stats.ATTACK, 30);
        this.stats.put(Stats.DEFENSE, 25);
    }

    /**
     * Ataca a un enemigo. Typo: In word 'Ataca'. Typo: In word 'enemigo'.
     *
     * @param enemy Enemigo a atacar. Typo: In word 'Enemigo'. Typo: In word 'atacar'.
     */
    @Override ⚡ AbrahamDell
    public void attack(GameCharacter enemy) {
        String enemyName = enemy.getName();
        // this.damage = this.stats.get(Stats.ATTACK) - enemy.getStats().get(Stats.DEFENSE);
    }
}
src > rpg > entities > enemies > secret > © ShadowAssassin 12:24 CRLF

```

```

n.java    © DemonLord.java    © Dragon.java    © ElderSlime.java    © EliteGoblin.java    © AncientDragon.java    © ShadowAssassin.java ×
public class ShadowAssassin extends Enemy { 2 usages  ⚡ AbrahamDell
    protected void initCharacter() {
        this.stats.put(Stats.DEFENSE, 25);
    }

    /**
     * Ataca a un enemigo. Typo: In word 'Ataca'. Typo: In word 'enemigo'.
     *
     * @param enemy Enemigo a atacar. Typo: In word 'Enemigo'. Typo: In word 'atacar'.
     */
    @Override ⚡ AbrahamDell
    public void attack(GameCharacter enemy) {
        String enemyName = enemy.getName();
        int damage = this.stats.get(Stats.ATTACK) - enemy.getStats().get(Stats.DEFENSE);
        int newHP = enemy.getStats().get(Stats.HP) - damage;
        enemy.getStats().put(Stats.HP, newHP);
        JOptionPane.showMessageDialog( parentComponent: null, String.format("%s ataca a %s por %d daño! %s tiene %d HP restantes.%n",
            this.name, enemyName, damage, enemyName, newHP));
    }
}

```

DemonLord.java

© Dragon.java

© ElderSlime.java

© EliteGoblin.jav

```
package rpg.entities.enemies;

import rpg.entities.GameCharacter;
import rpg.enums.EnemyType;

/**
 * Clase abstracta que representa a un enemigo. Typo: In word 'Cla
 *
 * @author [AbrahamDell]
 */
public abstract class Enemy extends GameCharacter { 8 inheritors  ⓘ Ab
    /**
     * Tipo de enemigo. Typo: In word 'Tipo'. Typo: In word 'enem
     */
    protected EnemyType type; 2 usages

    /**
     * Constructor que inicializa el enemigo con un nombre y tipo.
     *
     * @param name Nombre del enemigo. Typo: In word 'Nombre'. Ty
     * @param type Tipo de enemigo. Typo: In word 'Tipo'. Typo: I
     */
    public Enemy(String name, EnemyType type) { ⓘ AbrahamDell
        super(name);
        this.type = type;
    }
}
```

```

DemonLord.java    © Dragon.java    © ElderSlime.java    © EliteGoblin.java    © AncientDragon.java

public abstract class Enemy extends GameCharacter { 8 inheritors  ⓘ AbrahamDell +1
    super(name);
    this.type = type;
}

/**
 * Método abstracto que debe ser implementado por las clases que hereden de Enemy.
 * Obtiene el botín del enemigo. Typo: In word 'Obtiene'. Typo: In word 'botín'.
 */
public abstract void getLoot(); no usages 8 implementations ⓘ AbrahamDell    Method 'getLoot()' is r

/**
 * Obtiene el tipo de enemigo. Typo: In word 'Obtiene'. Typo: In word 'tipo'. Ty
 *
 * @return Tipo de enemigo. Typo: In word 'Tipo'. Typo: In word 'enemigo'.
 */
public EnemyType getType() { no usages ⓘ AbrahamDell    Method 'getType()' is r
    return type;
}
}

```

```

© Dragon.java    © ElderSlime.java    © EliteGoblin.java    © AncientDragon.java    © ShadowAssassin.java    © Enemy.java

package rpg;

import rpg.entities.Player;
import rpg.entities.enemies.Enemy;
import rpg.utils.Randomize;

import javax.swing.*;
import java.util.Random; Unused import statement.

/**
 * Clase que representa el juego. Typo: In word 'Clase'. Typo: In word 'representa'. Typo: In word 'juego'.
 *
 * @author [AbrahamDell]
 */
public class Game { ⓘ AbrahamDell
    /**
     * Método principal que inicia el juego. Typo: In word 'Método'. Typo: In word 'inicia'. Typo: In word 'juego'.
     *
     * @param args Argumentos de la línea de comandos. Typo: In word 'Argumentos'. Typo: In word 'comandos'.
     */
    public static void main(String[] args) { ⓘ AbrahamDell
        Player player = new Player(name: "Zeus");
        Enemy enemy = Randomize.getRandomEnemy();
        JOptionPane.showMessageDialog(parentComponent: null, message: "Estas luchando contra un " + enemy.getName());
        while (player.isAlive() && enemy.isAlive()) {

```

og > src > rpg > ⓘ Game

```

© Dragon.java    © ElderSlime.java    © EliteGoblin.java    © AncientDragon.java    © ShadowAssassin.java    © Enemy.java

public class Game {
    @param args Argumentos de la línea de comandos. Tipo: In word 'Argumentos'. Tipo: In word 'comandos'.
    */
    public static void main(String[] args) {
        Player player = new Player( name: "Zeus");
        Enemy enemy = Randomize.getRandomEnemy();
        JOptionPane.showMessageDialog( parentComponent: null, message: "Estas luchando contra un " + enemy.getName());
        while (player.isAlive() && enemy.isAlive()) {
            player.attack(enemy);
            if (enemy.isAlive()) {
                enemy.attack(player);
            }
        }
        if (player.isAlive()) {
            JOptionPane.showMessageDialog( parentComponent: null, message: "Ganaste!");
        } else {
            JOptionPane.showMessageDialog( parentComponent: null, message: "Perdiste!");
        }
    }
}

```

