

PLANET WARS



Autors:

Iván Figueredo y Sergi Simón

04/05/22 - 19/05/22



ÍNDICE

PLANET WARS	1
Introducción	3
Contenido técnico	4
Clase Main	4
Clase Battle	4
Clase Planet	5
Clase BaseDatos	6
Clase Abstracta Ship	6
Interface MilitaryUnit	6
Clases LightHunter, HeavyHunter, BattleShip, ArmoredShip	7
Clases MissileLauncher, IonCannon, PlasmaCannon	7
Clases VentanaBuild, VentanaBuildDefense, VentanaShip, VentanaInicio, VentanaUpgradeTechnology, VentanaViewBattleReports	7
Glossario	8
FAQ	9
Webgrafía / Bibliografía	10



Introducción

El videojuego Planet Wars es un RTS de gestión de recursos. En el que el jugador deberá de administrar un planeta, en el que, tendrá que tener en cuenta riesgos y su economía ya que, cada pocos ciclos del planeta será atacado por varios imperios galácticos rivales por el control de su galaxia.

En este videojuego podrás administrar tu planeta haciendo uso de un sistema de economía en el que tendrás que administrar dos tipos diferentes de recursos, el metal y el deuterio.

Además deberás de hacer uso de los paneles de gestión del planeta para poder comprar mejoras y comprar naves con las que podrás aumentar el nivel de tu imperio pudiendo aguantar todos los ataques de los imperios enemigos.

Durante el transcurso del juego te irán atacando diversos imperios galácticos enemigos, tu objetivo será aplacar esos ataques y mantener tu imperio.



Contenido técnico

Clase Main

Objetos instanciados:

- Main()
- Planet()
- Timer()
- TimerTask()

En la clase Main tenemos los siguientes métodos:

- ViewThread(), este método actualizará las labels del JPanel de alerta, este método llama a createEnemyArmy para generar las naves que atacarán el planeta.
- createEnemyArmy(), este método generará una flota enemiga semi proceduralmente, en base a unos valores preestablecidos

Clase Battle

Objetos instanciados en la clase battle:

- Planet();
- BaseDatos();
- Battle();
- Main();

En la clase Main tendremos los siguientes métodos:

- getCostFleets, en este método le pasamos los dos arrays de arraylist con las flotas, este nos devolverá un array de ints con el precio de los ejércitos que le hemos pasado en metal y en deuterio.
- getInitialUnits, en este método le pasamos los dos arrays de arraylist con las flotas, este nos devolverá un array con la cantidad de unidades iniciales con las que empieza la batalla.
- getUnits, en este método le pasamos los dos arrays de arraylist con las flotas, este nos devolverá un array con la cantidad de unidades iniciales con las que cuenta el arraylist al momento de llamar al método.

En la clase Main tendremos los siguientes atributos en privado:

- ArrayList<MilitaryUnit>[] planetArmy, en esta variable añadiremos los objetos de naves y defensas.
- ArrayList<MilitaryUnit>[] enemyArmy, en esta variable añadiremos los objetos de naves.
- battleDevelopment, en este String almacenaremos los pasos de las batallas
- actualNumberUnitsPlanet, este es la variable en la que almacenaremos el número de unidades en las que tiene el planeta
- enemyDrops, unidades que ha perdido el enemigo en la batalla.
- planetDrops, unidades que ha perdido el usuario en la batalla.



Clase Planet

La clase planet tiene los siguientes atributos:

- int technologyDefense;
- int technologyAttack;
- int metal;
- int deuterium;
- int upgradeDefenseTechnologyDeuteriumCost;
- int upgradeAttackTechnologyDeuteriumCost;
- ArrayList<MilitaryUnit>[] army = new ArrayList[7];

Además de los getter/setters de los mismos.

La clase planet tiene un constructor al que no necesitaremos pasarle ningun atributo a la hora de inicializarlo pero internamente de inicializará todas las posiciones del array de ArrayList de la clase Military Unit.

La clase planet tiene los siguientes métodos implementados:

- upgradeTechnologyDefense(int n), este método añade n niveles a la tecnología de defensa, si tenemos los materiales para hacerlo, sino calculará y comprará todos los que nos de el material. Este método lanza excepciones de tipo “ResourceException”.
- upgradeTechnologyAttack(int n), este método añade n niveles a la tecnología de ataque, si tenemos los materiales para hacerlo, sino calculará y comprará todos los que nos de el material. Este método lanza excepciones de tipo “ResourceException”.
- public void addLighthHunter(int n), este método añade n naves a la flota, si tenemos los materiales para hacerlo, sino calculará y comprará todos los que nos de el material. Este método lanza excepciones de tipo “ResourceException”.
- public void addHeavyHunter(int n) , este método añade n naves a la flota, si tenemos los materiales para hacerlo, sino calculará y comprará todos los que nos de el material. Este método lanza excepciones de tipo “ResourceException”.
- public void addBattleShip(int n), este método añade n naves a la flota, si tenemos los materiales para hacerlo, sino calculará y comprará todos los que nos de el material. Este método lanza excepciones de tipo “ResourceException”.
- public void addArmoredShip(int n), este método añade n naves a la flota, si tenemos los materiales para hacerlo, sino calculará y comprará todos los que nos de el material. Este método lanza excepciones de tipo “ResourceException”.
- public void addMissileLauncher(int n), este método añade n naves a la flota, si tenemos los materiales para hacerlo, sino calculará y comprará todos los que nos de el material. Este método lanza excepciones de tipo “ResourceException”.
- public void addIonCannon(int n), este método añade n naves a la flota, si tenemos los materiales para hacerlo, sino calculará y comprará todos los que nos de el material. Este método lanza excepciones de tipo “ResourceException”.
- public void addPlasmaCannon(int n), este método añade n naves a la flota, si tenemos los materiales para hacerlo, sino calculará y comprará todos los que nos de el material. Este método lanza excepciones de tipo “ResourceException”.



- public void refreshPlanetStats()

Esta clase también implementa la Excepcion “Resource Exception”.

Clase BaseDatos

La clase de base de datos implementa la interfaz Variables.

La clase BaseDatos tiene los siguientes métodos:

- getShipValues()
- getPlanetDeuterium()
- getPlanetMetal()
- setPlanetMetal()
- setPlanetDeuterium()
- getPlanetName()
- setPlanetAtkTech()
- getPlanetAtkTech()
- setPlanetDefTech()
- getPlanetDefTech()
- addShip()
- addDefense()

Clase Abstracta Ship

La clase abstracta Ship contiene un constructor al que le pasaremos el int armor, int initialArmor, int baseDamage.

Interface MilitaryUnit

- abstract int attack(), este método nos devolverá el poder de ataque que tenga la unidad.
- abstract void takeDamage(int receivedDamage), este método le deberemos pasar el daño recibido para restarle a la vida actual de la nave.
- abstract int getActualArmor(), este método nos devolverá el blindaje restante de la nave un ataque.
- abstract int getMetalCost(), este método nos devolverá el coste en metal de la unidad que deseamos crear.
- abstract int getDeuteriumCost(), este método nos devolverá el coste en deuterium de la unidad que deseamos crear.
- abstract int getChanceGeneratinWaste(), este método nos genera un número que lo usaremos para calcular la probabilidad de generar residuos al destruir una nave.
-



- abstract int getChanceAttackAgain(), este método nos genera un numero que lo usaremos para calcular la probabilidad que tiene x nave para poder efectuar otro disparo.
- abstract void resetArmor(), este método inicializará el blindage.

Clases LightHunter, HeavyHunter, BattleShip, ArmoredShip

Todas estas clases heredan de Ship que a su vez por herencia implementan la Interfaz MilitaryUnit.

Además todas contienen 2 constructores, 1 para la creación de naves por el usuario, al que le pasaremos los atributos de la civilización y un segundo constructor al que no le pasaremos nada para generar las naves de los enemigos que te van atacando proceduralmente.

Clases MissileLauncher, IonCannon, PlasmaCannon

Todas estas clases heredan de la clase Defense, y por herencia implementan la interfaz MilitaryUnit.

Además todas contienen 1 constructor al que le pasaremos la armadura de la Defensa y el daño base.

Clases VentanaBuild, VentanaBuildDefense, VentanaShip, VentanaInicio, VentanaUpgradeTechnology, VentanaViewBattleReports

Las clases de ventanas tienen un constructor en el que se declararán y se setearán todas los componentes necesarios para la interfaz, tales como botones, labels.



Glossario

JFrame: Ventana del juego

JPanel: Paneles dentro de la ventana del juego.

Label: Una label o etiqueta es un componente que muestra un String en un JPanel.

RTS: Los juegos RTS son juegos de estrategia en tiempo real, usualmente suelen ser RTS de gestión de recursos ya que deben de administrar los recursos de la zona/planeta para poder generar una estrategia en tiempo real.

Inicializar: Restablecer una variable a su valor inicial.

Instanciar: Es la particularización, realización específica u ocurrencia de una determinada clase.

Objeto: Instancia de una clase.

Método: Un método es un conjunto de órdenes y/o cálculos guardados para poder ser utilizado sin tener que recurrir a escribir el mismo código una y otra vez.

Constructor: Un constructor es un código cuya misión es inicializar, con o sin con los valores predeterminados, un objeto de una clase.

Clase Abstracta: Una clase abstracta es una clase la cual no puede ser instanciada pero sirve de superclase para otras clases gracias a que se heredan todos los métodos y variables, siempre teniendo en cuenta el scope.

Interfaz: Una interfaz en Java es una colección de métodos abstractos y propiedades constantes. En las interfaces se especifica qué se debe hacer pero no su implementación.

Scope o también llamado visibilidad es la propiedad de una clase, método o atributo la cual hace que desde determinadas partes del código no se pueda acceder a un valor y modificarlo.

Procedural: Una generación procedural significa que ha sido generado aleatoriamente siguiendo unos parámetros y una configuración muy concreta.



FAQ

- ¿Cómo puedo arrancar el juego?

Descarga el release y en la carpeta que lo hayas abres una terminal y ejecutas el siguiente comando:

```
java -jar PlanetWars.jar
```

Lo siguiente es disfrutar y divertirse jugando a PlanetWars

- ¿Cómo guardo las partidas?

Los datos de las partidas se van guardando a medida que el jugador va jugando

- ¿Se guardan automáticamente los progresos?

Si, además se guardan los datos en la base de datos al presionar el botón “Exit”



Webgrafía / Bibliografía

M2:

<http://www.rebellionrider.com/create-table-ddl-with-execute-immediate-in-oracle-database-part-2/>

M2: https://docs.oracle.com/cd/B13789_01/appdev.101/b10807/13_elems017.htm

M3:

<https://es.stackoverflow.com/questions/166467/error-java-sql-sqlexception-%C3%8Dndice-de-columna-no-v%C3%A1lido>

M3: <https://docs.oracle.com/javase/7/docs/api/java/sql/ResultSet.html>

M3: <https://www.ibm.com/docs/es/i/7.1?topic=resultsets-example-resultset-interface>

M3: [https://programacion.net/articulo/acceso_a_bases_de_datos_\[jdbc\]_93/6](https://programacion.net/articulo/acceso_a_bases_de_datos_[jdbc]_93/6)

M3: <http://www.java2s.com/Code/Java/Database-SQL-JDBC/OracleJDBCselect.htm>

M3: <https://stackoverflow.com/questions/8778422/java-oracle-jdbc-select-statement>

M3:

<https://www.jairogarciarincon.com/clase/interfaces-de-usuario-con-java-swing/componentes-jmenubar-jmenu-y-jmenuitem>

M3: https://www.tutorialspoint.com/how_to_add_jtable_to_panel_in_java_swing

M3:

<https://stackoverflow.com/questions/12543206/java-menu-item-enabling-within-event-listener>