IDLEMON GAME DOCUMENTATION

A note on how to play:
The game is run from the Main.java class. Download the .zip file found in the Source Code folder and compile the program to play! Before you run the Main.java class, you can set which world challenge you would like to play at the top of the JavaFX.java class. Once you run the Main.java class, an interactive window will pop up with the current resource values printed over the top of the world and one of each (there are three) Idlemon will be randomly located in the world. Click on the Idlemon and then click on either of the two buttons located at the bottom right of the window to either level up the Idlemon to individually convert more resources, or to buy a new version of that Idlemon to have more collectively converting resources. Figure out which method results in the quickest time a world can be satisfied!

Idlemon Key

- Flame Idlemon

- Grass Idlemon

- Water Idlemon

interface State

This interface serves as a source for extensibility to add observers to the game.

- public void update();
    - could be implemented for multiple players
- public void addObserver();
    - could be implemented for multiple players
- public long timeCompleted();
    - returns the time the challenge was completed in

abstract class World implements State

This abstract class serves as an implementation of the State interface and declares the methods defined in its child classes.

- public abstract void setInitialResources();
    - set start amount of resources

- public abstract Map<String,Integer> getResources();
    - return resources
- public abstract void modifyResources(String name, Integer amount);
    - modify amount of resources
- public abstract void setStartIdlemon();
    - set start amount/kind of idlemon
- public abstract ArrayList<Idlemon> getIdlemon();
    - return idlemon
- public abstract void addIdlemon();
    - modify amount of idlemon
- public abstract void checkWin();
    - check if challenge has been satisfied

class World1 extends World

This child class of the abstract class World serves as the definition of the FleurDeFlamme challenge, where one must get an absurd amount of flame resources to satisfy the win condition.

- String name = "FleurDeFlamme";
    - name of World challenge
- Map<String,Integer> resources = new HashMap<>();
    - map of resources
- ArrayList<Idlemon> idlemon;
    - arraylist of idlemon
- Integer flame;
    - amount of flame resource
- Integer water;
    - amount of water resource
- Integer grass;
    - amount of grass resource
- long startTime;
    - take note of start time
- long now;
    - take note of current time
- long elapsed;
    - take note of total time
- public String getName();
    - return the name of the World challenge
- public void World1()
    - constructor
- public void setInitialResources()
    - set start amount of resources

- public Map<String, Integer> getResources()
    - return resources
- public void modifyResources(String name, Integer amount)
    - modify amount of resources
- public void setStartIdlemon()
    - set start amount/kind of idlemon
- public ArrayList<Idlemon> getIdlemon()
    - return idlemon
- public void addIdlemon()
    - modify amount of idlemon
- public void checkWin()
    - check if challenge has been satisfied
- void update();
    - could be implemented for multiple players
- void addObserver();
    - could be implemented for multiple players
- long timeCompleted();
    - returns the time the challenge was completed in

class World2 extends World

This child class of the abstract class World serves as the definition of the WunderbareWasserfall challenge, where one must obtain a dangerous amount of water resources to satisfy the win condition.

- String name = "WunderbareWasserfall";
    - name of World challenge
- Map<String,Integer> resources = new HashMap<>();
    - map of resources
- ArrayList<Idlemon> idlemon;
    - arraylist of idlemon
- Integer flame;
    - amount of flame resource
- Integer water;
    - amount of water resource
- Integer grass;
    - amount of grass resource
- long startTime;
    - take note of start time
- long now;
    - take note of current time
- long elapsed;
    - take note of total time

- public String getName();
    - return the name of the World challenge
- public void World2()
    - constructor
- public void setInitialResources()
    - set start amount of resources
- public Map<String, Integer> getResources()
    - return resources
- public void modifyResources(String name, Integer amount)
    - modify amount of resources
- public void setStartIdlemon()
    - set start amount/kind of idlemon
- public ArrayList<Idlemon> getIdlemon()
    - return idlemon
- public void addIdlemon()
    - modify amount of idlemon
- public void checkWin()
    - check if challenge has been satisfied
- void update();
    - could be implemented for multiple players
- void addObserver();
    - could be implemented for multiple players
- long timeCompleted();
    - returns the time the challenge was completed in

abstract class Idlemon

This abstract class makes use of the Template pattern to create and declare the methods that make up an Idlemon.

- World world;
- constructor
    - Is of the format:
        - (String name, int CurrentLevel, int consumeCost, int produceAmount, Integer Cost, ArrayList<String> consumeResources,ArrayList<String> produceResources, World world)
- public void convert();
    - This is a template method. It calls consume(), produce(), and then special(). Consume() and Produce() are always the same, while special is currently void, but can be modified to add a special rule to the conversion process.
- public void consume();
    - Calls modifyResources() to decrease the worlds resources based on the amount consumed by the Idlemon. The consumption amount is scaled by 1/current level.

- public void produce();
    - Calls modifyResources() to increase the world's resources based on the amount produced by the Idlemon. The production amount is scaled by the current level.
- public void special();
    - This is the hook for the convert method (a template method). This can be modified, to insert special rules that are run each time an Idlemon converts resources from one form to another.
- public int getLevel();
    - Returns the current level of the Idlemon.
- public void levelUp();
    - Increments the level of the Idlemon. It also calls modifyResources() to detract from the resources the cost of the Idlemon.
- public Map getCost();
    - Returns a map where the key is the resource type must be spent to buy a new Idlemon, and the value is the amount of that resource.

class Blue extends Idlemon

This is an extension of Idlemon and creates the Water Idlemon which converts flame to water.

- constructor
    - Takes in the world, then makes a call to the super class constructor (the same format as above
- public Idlemon buyNew();
    - Calls modifyResources() and subtracts the cost of buying a new Idlemon from the world's resources. It then returns a new level 1 GreedIdlemon.

class Red extends Idlemon

This is an extension of Idlemon and creates the Flame Idlemon which converts grass to water.

- constructor
    - Takes in the world, then makes a call to the super class constructor (the same format as above
- public Idlemon buyNew();
    - Calls modifyResources() and subtracts the cost of buying a new Idlemon from the world's resources. It then returns a new level RedIdIdlemon.

class Green extends Idlemon

This is an extension of Idlemon and creates the Grass Idlemon which converts water to grass.

- constructor
    - Takes in the world, then makes a call to the super class constructor (the same format as above
- public Idlemon buyNew();

- Calls modifyResources() and subtracts the cost of buying a new Idlemon from the world's resources. It then returns a new level GreenIdIdlemon.