Sam Schock

12/11/20

CIT244 – Final Project

IdleR\_API

This API streamlines the process of making incremental, or idle, games. Incremental games are, in essence, resource management simulators. They are complex to code but simple to play. Typically, they involve setting up a series of resource generators and managing them as they slowly build up resources over time. Famous examples include Cookie Clicker and Universal Paperclips.

With this API, you can build a basic idle game just by defining the resources and conversion factors. The code takes care of tracking resource values and formatting the GUI. In addition, the classes and methods are robust enough that if you wanted to make a more complex game (featuring upgrades, special timed events, etc.), you could.

First, let’s introduce the main objects in this API: Resource, Factor, and Delta.

Resource

Resources are the currency of the game. Represent money, raw materials, goods produced, etc.

*Variables*

String **name** – name of the resource

String **desc** – description

Int **max** – max value of quant

Int **min** – minimum value of quant

Int **quant** – quantity of the resource is stored here!

Boolean **lock** – allows for certain resources to be locked or unlocked at different game stages

Boolean **hasMax** – indicates whether a value has been entered for max. Set to false by default. Automatically set to true by calling setMax method.

*Methods*

Setters and getters are fairly standard.

Setters for **max** and **min** contain control structures to prevent invalid parameters.

**setQuant** contains a control structure to make sure it is not outside any defined max or min.

|  |  |  |  |
| --- | --- | --- | --- |
| **Method Name** | **Return Type** | **Args** | **Description** |
| Resource() | resource | String n, Boolean l, Boolean m | Constructor; by default, sets desc to “”, min to 0, quant to 0 |
| modQuant | void | Int amt | Adds ‘amt’ to quant |

Factor

Factor is short for 'factory'. Factors produce and/or consume resources. Factors can be purchased and sold using resources.

*Variables*

String **name** – name of the resource

String **desc** – description

Delta[] **cost** – the cost of purchasing this factor

Delta[] **produce** – resource(s) produced by this factor

Delta[] **consume** – resource(s) consumed by this factor

*Note:* Delta arrays are used so that multiple resources can be assigned!

Int **quant** – how many of this factor you own

Double **sellFactor** – proportion of purchase cost received when this factor is sold

Boolean **hasProd** – indicates whether this factor produces resources. Automatically set to true by calling setProduce.

Boolean **hasCon** - indicates whether this factor consumes resources. Automatically set to true by calling setConsume.

Boolean **canSell** – sets whether this factor can be sold

Boolean **lock** – allows for certain factors to be locked or unlocked at different game stages

*Methods*

Setters and getters are fairly standard. **setProduce()** and **setConsume()** automatically set **hasProd** and **hasCon**, respectively, to true.

|  |  |  |  |
| --- | --- | --- | --- |
| **Method Name** | **Return Type** | **Args** | **Description** |
| Factor() | factor | String name, Delta[] cost, boolean canSell, boolean lock | Constructor; by default, sets desc to “”, quant to 0, sellFactor to 0.5, hasCon to false, hasProd to false |
| getCostInt() | int | n/a | Returns the int value associated with the factor’s cost Delta |
| getCostString() | String | n/a | Returns the String value associated with the factor’s cost Delta |
| Convert() | void | n/a | Consumes & produces resources in accordance with this factor’s Deltas |
| Buy() | void | int amt | Verifies sufficient resources, adds amt factors, deducts cost resources |
| Sell() | void | int amt | Verifies sufficient factors, reduces amt factors, adds cost resources (times sellFactor) |

Delta

Delta represents a transaction of Resources, such as the cost of purchase, the materials consumed, or the goods produced. I chose the name Delta because it represents change in mathematics, but not in a specific direction!

*Variables*

Resource **rec** – the resource to be exchanged

Int **quant** – the amount to be exchanged

*Methods*

Setters, getters, & constructor are standard. There are no additional methods.

Contained in the folder are 4 java files:

**IdleR.java** is the main API class file. It contains all the classes and methods described above.

**IdleGameTemplate.java** is a template to build an idle game. It contains comments to guide the user through building their set of Resources, Deltas, and Factors. It also contains a large amount of code dedicated to taking those building blocks and formatting them into a GUI containing individual buy/sell buttons for each factor, and a readout that displays the amount of resources. The code also implements a timeline loop, which executes **convert()** commands and updates the resources once per second.

**WidgetFactoryDemo.java** is a 4 factor, 4 resource idle game that implements all the classes & methods designed in the API.



**CookieClicker.java** is another demo that replicates the basic function of classic idle game, Cookie Clicker.

