



Class Helper

- String `_name` — name of the Helper
- int `_level` — current level of the Helper
- String `_upgrade` — current tier of upgrades

METHODS

- `Helper()` — default constructor that initializes `_level` and `_upgrade`
- `String name()` — accessor for `_name`
- `int level()` — accessor for `_level`
- `String upgrade()` — accessor for `_upgrade`
- `void levelUp()` — indicates that the Helper has leveled up by setting the `_level` to `_level + 1`.
Note: the actual increases will be handled in the subclasses.
- `void upgradeUp()` — indicates that the Helper has leveled up by setting the `_upgrade` to `_upgrade + 1`. Note: the actual effects will be handled in the subclasses.
- `String[] stats()` — returns the stats of a Helper (name, level, and upgrade tier) in a 1-D array

The superclass, with Miner, Engineer, and Gambler as its subclasses. What else is there to say?

Class Miner

- `int _baseGoldPS` — base value of the gold rate
- `int _additionGoldPS` — addition bonus received from upgrades
- `int _multiGoldPS` — multiplier bonus received from upgrades

METHODS

- `Miner()` — default constructor. Initializes `_baseGoldPS`, `_additionGoldPS`, and `_multiGoldPS`.
- `Miner(String name)` — overloaded constructor. Initializes `_name`.
- `additionGold()` — accessor for the addition variable.
- `multiGold()` — accessor for the multiplier variable
- `addMutate(int addend)` — modifies the addition variable
- `multiMutate(int multi)` — modifies the multi variable

A Helper that produces gold per second.

Class Engineer

- `int _baseGoldPKP` — base value of the gold rate
- `int _additionGoldPKP` — addition bonus received from upgrades
- `int _multiGoldPKP` — multiplier bonus received from upgrades

METHODS

- `Engineer()` — default constructor. Initializes `_baseGoldPKP`, `_additionGoldPKP`, and `_multiGoldPKP`.
- `Engineer(String name)` — overloaded constructor. Initializes `_name`.
- `additionGold()` — accessor for the addition variable.
- `multiGold()` — accessor for the multiplier variable
- `addMutate(int addend)` — modifies the addition variable
- `multiMutate(int multi)` — modifies the multi variable

A Helper that produces gold per key press.

Class Gambler

- `int _luck` — the amount of luck that helps increase the chances of winning the lottery
- `final int _glevel` — caps the `_level` value at 1

METHODS

- `Gambler()` — default constructor. Initializes `_luck` and `_glevel` (which is the `_level` value).
- `Gambler(String name)` — overloaded constructor. Initializes `_name`.
- `int level()` — overwritten accessor. Returns `_glevel` instead of `_level`.

A peculiar type of `Helper` that does not produce gold; instead, it exists to increase the chances of winning the lottery. In addition, the `Gambler` does not level up; hence the overwritten accessor of `level()`.

Class DataStorage

- `String[][] _miners` — 2-D array with a list of miners, detailing their names, levels, upgrades, and upgrade type
- `String[][] _engineers` — 2-D array with a list of engineers, detailing their names, levels, upgrades, and upgrade type
- `String[][] _gamblers` — 2-D array with a list of gamblers, detailing their names, upgrades, and upgrade type

METHODS

- `String minerList()` — returns a `String` containing a formatted table using `_miners`
- `String engineerList()` — returns a `String` containing a formatted table using `_engineers`
- `String gamblerList()` — returns a `String` containing a formatted table using `_gamblers`
- `void addMiner (Miner m)` — adds a `Miner` to `_miners`. This includes names, levels, upgrades, and upgrade type.
- `void addEngineer (Engineer e)` — adds an `Engineer` to `_engineers`. This includes names, levels, upgrades, and upgrade type.
- `void addGambler (Gambler g)` — adds a `Gambler` to `_gamblers`. This includes names, upgrades, and upgrade type.
- `String sortList(String[][] helper, int col)` — sorts in ascending order based on the selected columns. Rows are switched to the appropriate spots

Example of the formatted table as outputted in the terminal (for miners):

Name	Level	Upgrade Tier	Gold
Bill	2	I	30 (Additive)
Maggie	20	III (max)	1.3 (Multiplicative)
Raunak	5	II	45 (Additive)

This class essentially acts as our data storage — all Helpers are tracked using this object. When another Helper is purchased, it is added to the array of the Helper subclass it belongs to. If a certain subclass of Helper needs to be seen by the user, then that respective `<Helper>List()` method is invoked.

Class Events

METHODS

- Boolean isEarthquake() — determines if there should be an earthquake event
- Boolean isThunderstorm() — determines if there should be a thunderstorm event
- Boolean isBankruptcy() — determines if there should be a bankruptcy event
- Boolean isRobbery() — determines if there should be a robbery event
- Boolean jackpot(double luck) — determines if there should be a lottery event, based on the luck value

NOTE: All methods are static

A class that provides the RNG-based event activations.

Class UserInterface

METHODS

- void introUI() — prints out the “GOLFINGER” ASCII art
- void mainUI() — prints out the UI for the main action (pressing SPACE)
- void storeUI() — prints out the store menu
- void helpUI() — prints out a set of instructions/controls for the game

NOTE: All methods are static

This class is essentially stores our templates for the terminal, and are printed upon function call. This is an effort to reduce clutter in class Woo

Class Store

METHODS

- `buyMiner()` — “purchases” or creates a new Miner object for a given amount of gold, starting from level 1 and upgrade I
- `buyEngineer()` — “purchases” or creates a new Engineer object for a given amount of gold, starting from level I
- `buyGambler()` — “purchases” or creates a new Gambler object for a given amount of gold, starting from level I
- `upgradeMiner(Miner m)` — gives all Miners a stat upgrade (boosts one of its instance variables permanently, for all current and future objects of that class)
- `upgradeEngineer(Engineer e)` — gives all Engineer a stat upgrade (boosts one of its instance variables permanently, for all current and future objects of that class)
- `upgradeGambler(Gambler g)` — gives all Gamblers a stat upgrade (boosts one of its instance variables permanently, for all current and future objects of that class)
- `levelMiner(Miner m)` — increases the level of one Miner, and thus improves its gold producing capabilities
- `levelEngineer(Engineer e)` —increases the level of one Engineer, and thus improves its gold producing capabilities
- `levelGambler(Gambler g)` — increases the level of one Gambler, and thus improves its gold producing capabilities
- `buyPowerUp()` — allows user to purchase a power up, which temporarily boosts stats

Class Woo

`_gold` - unspent accumulated gold

`_baseGoldPKP` - final variable = 1 [starting amount per “click”]

`_additiveGoldPKP` - total amount additive engineers contribute

`_multiGoldPKP` - total amount multiplicative engineers contribute

`_totalGoldPKP` - (`_baseGoldPKP` + `_additiveGoldPKP`) * `_multiGoldPKP`

`_additiveGoldPS` - total amount additive miners contribute

`_multiGoldPS` - total amount multiplicative miners contribute

`_totalGoldPS` - `_additiveGoldPKP` * `_multiGoldPKP`

`_baseLuck` - final variable = 0.0005 [probability of winning lottery]

`_additiveLuck` - total amount additive gamblers contribute

`_multiLuck` - total amount multiplicative engineers contribute

`_totalLuck` - (`_baseLuck` + `_additiveLuck`) * `_multiLuck`

`_stats` - array of player statistics

`_stats[0]` - total gold ever earned

`_stats[1]` - number of miners

`_stats[2]` - number of engineers

`_stats[3]` - number of gamblers

`_stats[4]` - number of spaces entered by player

`_stats[5]` - number of powerups activated by player