Instructors Manual for Gilded Rose for Exploratory Testing

By: Maaret Pyhäjärvi
Based on work code by Emily Bache
Special thanks to: Elizabeth Zagroba, Joep Schuurkes, Eva Nanyonga, Erik Kasakya

Notes:

- *The environment setup needs to be prepped -- code coverage in IDE particularly*
- *Guide to be systematic, checklist of things you could have a test on would be good*
- *Test naming: test negativequalityisnotaccepted*

Purpose of the exercise:

- Teach designing tests for coverage: spec, code and risk
- Introduce exploratory testing on unit test scope

Timebox: 1,5 hrs to run the exercise

Setup:

- For students: remote control sharing on zoom for instructors computer solo, pair or ensemble
- For instructor:
    - Setup for python / (Setup for java)
        - IDE: visual studio code for python; eclipse for java (most visual code coverage)
        - Starter test samples assert, approvals, parameterized in own files
        - Requirements without the request to extend functionality
        - Visual code coverage tool
    - Solo, pair or ensemble composition

Exercise background:

- Originally presented as multilanguage refactoring exercise to lock functionality with tests and then clean up - oracle "works as before"
- For testers makes sense to extend beyond current implementation to feedback relevant to spec & code as (exploratory) test design exercise

Session structure:

- (5 min) show and tell: one test case from the developer, code, spec → how to run the test
- (5 min) individual reading of the spec
- (50 min) add tests to cover spec
- (10 min) code coverage as complete
- (10 min) sum up as combination-approvaltests
- (10 min) debriefing: group and instructor point out successes and mistakes as summary

Variations:

- Start with combination-approvaltests → moves focus to accidental discovery of spec in action
- Start with parametrized test → removes focus from naming a test and changes understanding of spec

Facilitation tips

Pay attention to unusual successes (listed for reference from previous sessions) and usual mistakes. Keep the work moving forward without removing mistakes the students can try figuring out themselves.

Unusual successes

* Using code for names and understanding that it only knows what it knows
* Glancing through the code in the beginning
* Leaving out scenario with time moving forward, clear failures on start/stop
* combinations - extra discovery
* understanding context to build around this
* good pairing without specific roles
* choosing good values to test with - asymmetry

Usual mistakes -> responses from the instructor

- Getting stuck - not understanding what "test this" means
    - → point out that designing function calls with values and expected values to assert is the work
    - → point out you expect 10 or more examples before this program is covered with test
    - → allow for copied tests, refactored tests, parametrized tests (only if someone knows them already) and approvaltests but default to copied tests first
- Getting stuck - preparing a whole plan of everything while not understanding that it's going to be many tests
    - → say we expect many tests
    - → guide to creating one example of things mentioned in the spec
    - → emphasize implementation (covers all scenarios) and testing (uncovers all scenarios)
- Getting stuck - not understanding do (with input) - verify -structure or input-output connection
    - → explain theory line by line with code comments
    - → focus on an example
- Getting stuck - starting off without seeing it first work
    - → allow to write one that does not test the right things and then enforce basic "see it work" case; return to first case after understanding functionality
- Getting stuck - 15 tests to create in 50 minutes leaves 3 minutes per test
    - → track time to complete test and lighten up requirements if pace is slow
    - → possible to drop good naming requirement at first and return to names after 5 tests when better understanding of the problem
    - → possible to drop copy-pasting tests and write just values with parameterized tests or approvaltests

    - → possible to introduce "works in production" as oracle to avoid time on bugs on spec
- Finding bugs that are bugs in test - item short names in spec and long names in code for Sulfuras & Backstage passes
    - → point out that when something fails visibly, problem can be in tests
    - → point out item names in implementation
- Unable to decide what to do with failing test
    - → guide to writing a BUG comment and leave behind a passing test (for coverage)
- Lot in weeds of special cases - over focusing on error message based input validation or special inputs
    - → ask if they know the baseline of how it can work
    - → ask what portion of spec they already cover
    - → encourage brainstorming on meaning of spec but turning it to action by exploring with the tests
- Not having the basic knowledge - not knowing what to change in code, not knowing how to read results
    - → explain only when needed!
    - → explain how to as you notice the student is not moving due to confusion
    - → explain changing just name and values
    - → explain unique naming and noticing red
    - → explain reading test output
    - → explain how to run tests from shortcuts, command line , test list in IDE only as they are needed
        - Python: pytest test_filename.py::testname runs single test on command line
        - Java in Eclipse/mac: cmd+shift+F11 runs all tests or one test when name is selected. ctrl+shift+F11 runs same tests under coverage.
    - → explain language quirks like java/semicolons and python/tabs
- Having trouble naming things
    - → introduce bad and quick naming and that naming is a process where you improve things as soon as you know better, call students to return to improve naming
- Having trouble understanding spec
    - → point out not noticing the shorthand of sulfuras and backstage passes in spec vs. long form in code
    - → point out not understanding concept of sell in (days not dates) through calling attention to value types being integers
    - → point out we can test on the ambiguity of limits in the spec with both values
- Having trouble extending test ideas from what has been said in spec
    - Unusual value combinations

    - Unusual values - but not too much of them
- Having trouble with the values the students themselves choose
    - → point out asymmetric test data to better follow what value is what
- Going out of scope of exercise in time-consuming way
    - → introduce boundaries on moving to parameterized tests or inventory management / multi-item lists as combinations to test to scope them out
- Usual misses on coverage on spec/code
    - → brie quality grows by 2 when it's sell in days passes - implicit in spec
    - → input validation is not error messages but not changing the value when it was illegal unless it turns legal in one step

<u>Tech trouble</u>

- zoom control sharing does not work at all or is preventing being hands on keyboard
    - → revisit the roles in ensemble / pair to include on other work than being hands
- running 1st test fails unexpectedly
    - → test first green just before session
    - → fix on the fly, this time needed activating the virtual environment as global environment had broken latest of one of the tools

One example of tests added in order of adding them:

- Missing: tests around max value for quality when quality is growing (Aged Brie)

```
# -*- coding: utf-8 -*-
import pytest

from gilded_rose import Item, GildedRose

def test_something():
    items = [Item("something", 0, 0)]
    gilded_rose = GildedRose(items)
    gilded_rose.update_quality()
    assert(0 == items[0].quality)

# instructions said quality shouldn't be above 50, but we were able to set it to
51
"""def test_quality_over_50():
    items = [Item("something", 0, 51)]
    gilded_rose = GildedRose(items)
    assert(50 == items[0].quality)"""

# when sell_in < 0, quality degrades by 2 instead of 1
def test_quality_degrades_2x():
    items = [Item("something", 0, 10)]
    gilded_rose = GildedRose(items)
    gilded_rose.update_quality()
    assert(8 == items[0].quality)
    assert(-1 == items[0].sell_in)

def test_quality_degrades_1x():
    items = [Item("something", 1, 10)]
    gilded_rose = GildedRose(items)
    gilded_rose.update_quality()
    assert(9 == items[0].quality)
    assert(0 == items[0].sell_in)

# you can have negative quality, and you can't degrade negative quality
def test_degrading_negative_quality():
    items = [Item("something", 1, -1)]
    gilded_rose = GildedRose(items)
```

```
    gilded_rose.update_quality()
    assert(-1 == items[0].quality)
    assert(0 == items[0].sell_in)

def test_confirm_brie_aging_improves_quality():
    items = [Item("Aged Brie", 1, 10)]
    gilded_rose = GildedRose(items)
    gilded_rose.update_quality()
    assert(11 == items[0].quality)
    assert(0 == items[0].sell_in)

# Brie counts, brie doesn't
def test_fake_brie():
    items = [Item("Aged brie", 1, 10)]
    gilded_rose = GildedRose(items)
    gilded_rose.update_quality()
    assert(9 == items[0].quality)
    assert(0 == items[0].sell_in)

# see test_degrading_negative_quality, quality outside of range isn't updated
def test_brie_cant_upgrade_beyond_50():
    items = [Item("Aged Brie", 1, 50)]
    gilded_rose = GildedRose(items)
    gilded_rose.update_quality()
    assert(50 == items[0].quality)
    assert(0 == items[0].sell_in)

def test_brie_can_be_over_50():
    items = [Item("Aged Brie", 1, 51)]
    gilded_rose = GildedRose(items)
    gilded_rose.update_quality()
    assert(51 == items[0].quality)
    assert(0 == items[0].sell_in)

# TODO: increase and decrease of quality below 0 and above 50, not every case
captured yet

def test_sulfuras_sell_in_date_does_not_change():
    items = [Item("Sulfuras, Hand of Ragnaros", 1, 10)]
```

```
    gilded_rose = GildedRose(items)
    gilded_rose.update_quality()
    assert(10 == items[0].quality)
    assert(1 == items[0].sell_in)

def test_sulfuras_is_not_the_full_name():
    items = [Item("Sulfuras", 1, 10)]
    gilded_rose = GildedRose(items)
    gilded_rose.update_quality()
    assert(9 == items[0].quality)
    assert(0 == items[0].sell_in)

# this fails because neither sell_in nor quality can be strings, they must be
ints
"""def test_sulfuras_NaN():
    items = [Item("Sulfuras, Hand of Ragnaros", 1, "joep")]
    gilded_rose = GildedRose(items)
    gilded_rose.update_quality()
    assert("joep" == items[0].quality)
    assert(1 == items[0].sell_in)"""

def test_backstage_pass_beyond_10_days():
    items = [Item("Backstage passes to a TAFKAL80ETC concert", 12, 5)]
    gilded_rose = GildedRose(items)
    gilded_rose.update_quality()
    assert(6 == items[0].quality)
    assert(11 == items[0].sell_in)

def test_backstage_pass_smaller_than_10_days():
    items = [Item("Backstage passes to a TAFKAL80ETC concert", 10, 5)]
    gilded_rose = GildedRose(items)
    gilded_rose.update_quality()
    assert(7 == items[0].quality)
    assert(9 == items[0].sell_in)

def test_backstage_pass_smaller_than_5_days():
    items = [Item("Backstage passes to a TAFKAL80ETC concert", 5, 10)]
    gilded_rose = GildedRose(items)
    gilded_rose.update_quality()
```

```
    assert(13 == items[0].quality)
    assert(4 == items[0].sell_in)

def test_backstage_pass_smaller_than_0_days():
    items = [Item("Backstage passes to a TAFKAL80ETC concert", 0, 10)]
    gilded_rose = GildedRose(items)
    gilded_rose.update_quality()
    assert(0 == items[0].quality)
    assert(-1 == items[0].sell_in)


# more than one of stuff --> how does inventory management work with discrete
items
```

One example of approvaltests:

```
from approvaltests.combination_approvals import verify_all_combinations

def test_combinations():
    verify_all_combinations(do_update_quality, [
        ["something", "Aged Brie", "Aged brie", "Backstage passes to a
TAFKAL80ETC concert", "Sulfuras", "Sulfuras, Hand of Ragnaros"],
        [0, 5, 10, -1],
        [0, -1, 50, 51, 5, 6, 1]])

def do_update_quality(name, sell_in, quality):
    items = [Item(name, sell_in, quality)]
    gilded_rose = GildedRose(items)
    gilded_rose.update_quality()
    return str(items[0])
```

Combinations to approve (168 tests):

```
args: ('something', 0, 0) => 'something, -1, 0'
args: ('something', 0, -1) => 'something, -1, -1'
args: ('something', 0, 50) => 'something, -1, 48'
args: ('something', 0, 51) => 'something, -1, 49'
args: ('something', 0, 5) => 'something, -1, 3'
args: ('something', 0, 6) => 'something, -1, 4'
args: ('something', 0, 1) => 'something, -1, 0'
args: ('something', 5, 0) => 'something, 4, 0'
args: ('something', 5, -1) => 'something, 4, -1'
args: ('something', 5, 50) => 'something, 4, 49'
args: ('something', 5, 51) => 'something, 4, 50'
args: ('something', 5, 5) => 'something, 4, 4'
args: ('something', 5, 6) => 'something, 4, 5'
args: ('something', 5, 1) => 'something, 4, 0'
```

```
args: ('something', 10, 0) => 'something, 9, 0'
args: ('something', 10, -1) => 'something, 9, -1'
args: ('something', 10, 50) => 'something, 9, 49'
args: ('something', 10, 51) => 'something, 9, 50'
args: ('something', 10, 5) => 'something, 9, 4'
args: ('something', 10, 6) => 'something, 9, 5'
args: ('something', 10, 1) => 'something, 9, 0'
args: ('something', -1, 0) => 'something, -2, 0'
args: ('something', -1, -1) => 'something, -2, -1'
args: ('something', -1, 50) => 'something, -2, 48'
args: ('something', -1, 51) => 'something, -2, 49'
args: ('something', -1, 5) => 'something, -2, 3'
args: ('something', -1, 6) => 'something, -2, 4'
args: ('something', -1, 1) => 'something, -2, 0'
args: ('Aged Brie', 0, 0) => 'Aged Brie, -1, 2'
args: ('Aged Brie', 0, -1) => 'Aged Brie, -1, 1'
args: ('Aged Brie', 0, 50) => 'Aged Brie, -1, 50'
args: ('Aged Brie', 0, 51) => 'Aged Brie, -1, 51'
args: ('Aged Brie', 0, 5) => 'Aged Brie, -1, 7'    ← case the above hand-crafted tests missed
args: ('Aged Brie', 0, 6) => 'Aged Brie, -1, 8'
args: ('Aged Brie', 0, 1) => 'Aged Brie, -1, 3'
args: ('Aged Brie', 5, 0) => 'Aged Brie, 4, 1'
args: ('Aged Brie', 5, -1) => 'Aged Brie, 4, 0'
args: ('Aged Brie', 5, 50) => 'Aged Brie, 4, 50'
args: ('Aged Brie', 5, 51) => 'Aged Brie, 4, 51'
args: ('Aged Brie', 5, 5) => 'Aged Brie, 4, 6'
args: ('Aged Brie', 5, 6) => 'Aged Brie, 4, 7'
args: ('Aged Brie', 5, 1) => 'Aged Brie, 4, 2'
args: ('Aged Brie', 10, 0) => 'Aged Brie, 9, 1'
args: ('Aged Brie', 10, -1) => 'Aged Brie, 9, 0'
args: ('Aged Brie', 10, 50) => 'Aged Brie, 9, 50'
args: ('Aged Brie', 10, 51) => 'Aged Brie, 9, 51'
args: ('Aged Brie', 10, 5) => 'Aged Brie, 9, 6'
args: ('Aged Brie', 10, 6) => 'Aged Brie, 9, 7'
args: ('Aged Brie', 10, 1) => 'Aged Brie, 9, 2'
args: ('Aged Brie', -1, 0) => 'Aged Brie, -2, 2'
args: ('Aged Brie', -1, -1) => 'Aged Brie, -2, 1'
args: ('Aged Brie', -1, 50) => 'Aged Brie, -2, 50'
args: ('Aged Brie', -1, 51) => 'Aged Brie, -2, 51'
args: ('Aged Brie', -1, 5) => 'Aged Brie, -2, 7'
args: ('Aged Brie', -1, 6) => 'Aged Brie, -2, 8'
args: ('Aged Brie', -1, 1) => 'Aged Brie, -2, 3'
args: ('Aged brie', 0, 0) => 'Aged brie, -1, 0'
args: ('Aged brie', 0, -1) => 'Aged brie, -1, -1'
args: ('Aged brie', 0, 50) => 'Aged brie, -1, 48'
args: ('Aged brie', 0, 51) => 'Aged brie, -1, 49'
args: ('Aged brie', 0, 5) => 'Aged brie, -1, 3'
args: ('Aged brie', 0, 6) => 'Aged brie, -1, 4'
args: ('Aged brie', 0, 1) => 'Aged brie, -1, 0'
args: ('Aged brie', 5, 0) => 'Aged brie, 4, 0'
args: ('Aged brie', 5, -1) => 'Aged brie, 4, -1'
args: ('Aged brie', 5, 50) => 'Aged brie, 4, 49'
args: ('Aged brie', 5, 51) => 'Aged brie, 4, 50'
args: ('Aged brie', 5, 5) => 'Aged brie, 4, 4'
args: ('Aged brie', 5, 6) => 'Aged brie, 4, 5'
args: ('Aged brie', 5, 1) => 'Aged brie, 4, 0'
args: ('Aged brie', 10, 0) => 'Aged brie, 9, 0'
args: ('Aged brie', 10, -1) => 'Aged brie, 9, -1'
args: ('Aged brie', 10, 50) => 'Aged brie, 9, 49'
args: ('Aged brie', 10, 51) => 'Aged brie, 9, 50'
args: ('Aged brie', 10, 5) => 'Aged brie, 9, 4'
args: ('Aged brie', 10, 6) => 'Aged brie, 9, 5'
args: ('Aged brie', 10, 1) => 'Aged brie, 9, 0'
args: ('Aged brie', -1, 0) => 'Aged brie, -2, 0'
args: ('Aged brie', -1, -1) => 'Aged brie, -2, -1'
args: ('Aged brie', -1, 50) => 'Aged brie, -2, 48'
args: ('Aged brie', -1, 51) => 'Aged brie, -2, 49'
args: ('Aged brie', -1, 5) => 'Aged brie, -2, 3'
args: ('Aged brie', -1, 6) => 'Aged brie, -2, 4'
args: ('Aged brie', -1, 1) => 'Aged brie, -2, 0'
args: ('Backstage passes to a TAFKAL80ETC concert', 0, 0) => 'Backstage passes to a TAFKAL80ETC concert, -1, 0'
args: ('Backstage passes to a TAFKAL80ETC concert', 0, -1) => 'Backstage passes to a TAFKAL80ETC concert, -1, 0'
args: ('Backstage passes to a TAFKAL80ETC concert', 0, 50) => 'Backstage passes to a TAFKAL80ETC concert, -1, 0'
args: ('Backstage passes to a TAFKAL80ETC concert', 0, 51) => 'Backstage passes to a TAFKAL80ETC concert, -1, 0'
args: ('Backstage passes to a TAFKAL80ETC concert', 0, 5) => 'Backstage passes to a TAFKAL80ETC concert, -1, 0'
args: ('Backstage passes to a TAFKAL80ETC concert', 0, 6) => 'Backstage passes to a TAFKAL80ETC concert, -1, 0'
args: ('Backstage passes to a TAFKAL80ETC concert', 0, 1) => 'Backstage passes to a TAFKAL80ETC concert, -1, 0'
args: ('Backstage passes to a TAFKAL80ETC concert', 5, 0) => 'Backstage passes to a TAFKAL80ETC concert, 4, 3'
args: ('Backstage passes to a TAFKAL80ETC concert', 5, -1) => 'Backstage passes to a TAFKAL80ETC concert, 4, 2'
args: ('Backstage passes to a TAFKAL80ETC concert', 5, 50) => 'Backstage passes to a TAFKAL80ETC concert, 4, 50'
args: ('Backstage passes to a TAFKAL80ETC concert', 5, 51) => 'Backstage passes to a TAFKAL80ETC concert, 4, 51'
args: ('Backstage passes to a TAFKAL80ETC concert', 5, 5) => 'Backstage passes to a TAFKAL80ETC concert, 4, 8'
args: ('Backstage passes to a TAFKAL80ETC concert', 5, 6) => 'Backstage passes to a TAFKAL80ETC concert, 4, 9'
```

```
args: ('Backstage passes to a TAFKAL80ETC concert', 5, 1) => 'Backstage passes to a TAFKAL80ETC concert, 4, 4'
args: ('Backstage passes to a TAFKAL80ETC concert', 10, 0) => 'Backstage passes to a TAFKAL80ETC concert, 9, 2'
args: ('Backstage passes to a TAFKAL80ETC concert', 10, -1) => 'Backstage passes to a TAFKAL80ETC concert, 9, 1'
args: ('Backstage passes to a TAFKAL80ETC concert', 10, 50) => 'Backstage passes to a TAFKAL80ETC concert, 9, 50'
args: ('Backstage passes to a TAFKAL80ETC concert', 10, 51) => 'Backstage passes to a TAFKAL80ETC concert, 9, 51'
args: ('Backstage passes to a TAFKAL80ETC concert', 10, 5) => 'Backstage passes to a TAFKAL80ETC concert, 9, 7'
args: ('Backstage passes to a TAFKAL80ETC concert', 10, 6) => 'Backstage passes to a TAFKAL80ETC concert, 9, 8'
args: ('Backstage passes to a TAFKAL80ETC concert', 10, 1) => 'Backstage passes to a TAFKAL80ETC concert, 9, 3'
args: ('Backstage passes to a TAFKAL80ETC concert', -1, 0) => 'Backstage passes to a TAFKAL80ETC concert, -2, 0'
args: ('Backstage passes to a TAFKAL80ETC concert', -1, -1) => 'Backstage passes to a TAFKAL80ETC concert, -2, 0'
args: ('Backstage passes to a TAFKAL80ETC concert', -1, 50) => 'Backstage passes to a TAFKAL80ETC concert, -2, 0'
args: ('Backstage passes to a TAFKAL80ETC concert', -1, 51) => 'Backstage passes to a TAFKAL80ETC concert, -2, 0'
args: ('Backstage passes to a TAFKAL80ETC concert', -1, 5) => 'Backstage passes to a TAFKAL80ETC concert, -2, 0'
args: ('Backstage passes to a TAFKAL80ETC concert', -1, 6) => 'Backstage passes to a TAFKAL80ETC concert, -2, 0'
args: ('Backstage passes to a TAFKAL80ETC concert', -1, 1) => 'Backstage passes to a TAFKAL80ETC concert, -2, 0'
args: ('Sulfuras', 0, 0) => 'Sulfuras, -1, 0'
args: ('Sulfuras', 0, -1) => 'Sulfuras, -1, -1'
args: ('Sulfuras', 0, 50) => 'Sulfuras, -1, 48'
args: ('Sulfuras', 0, 51) => 'Sulfuras, -1, 49'
args: ('Sulfuras', 0, 5) => 'Sulfuras, -1, 3'
args: ('Sulfuras', 0, 6) => 'Sulfuras, -1, 4'
args: ('Sulfuras', 0, 1) => 'Sulfuras, -1, 0'
args: ('Sulfuras', 5, 0) => 'Sulfuras, 4, 0'
args: ('Sulfuras', 5, -1) => 'Sulfuras, 4, -1'
args: ('Sulfuras', 5, 50) => 'Sulfuras, 4, 49'
args: ('Sulfuras', 5, 51) => 'Sulfuras, 4, 50'
args: ('Sulfuras', 5, 5) => 'Sulfuras, 4, 4'
args: ('Sulfuras', 5, 6) => 'Sulfuras, 4, 5'
args: ('Sulfuras', 5, 1) => 'Sulfuras, 4, 0'
args: ('Sulfuras', 10, 0) => 'Sulfuras, 9, 0'
args: ('Sulfuras', 10, -1) => 'Sulfuras, 9, -1'
args: ('Sulfuras', 10, 50) => 'Sulfuras, 9, 49'
args: ('Sulfuras', 10, 51) => 'Sulfuras, 9, 50'
args: ('Sulfuras', 10, 5) => 'Sulfuras, 9, 4'
args: ('Sulfuras', 10, 6) => 'Sulfuras, 9, 5'
args: ('Sulfuras', 10, 1) => 'Sulfuras, 9, 0'
args: ('Sulfuras', -1, 0) => 'Sulfuras, -2, 0'
args: ('Sulfuras', -1, -1) => 'Sulfuras, -2, -1'
args: ('Sulfuras', -1, 50) => 'Sulfuras, -2, 48'
args: ('Sulfuras', -1, 51) => 'Sulfuras, -2, 49'
args: ('Sulfuras', -1, 5) => 'Sulfuras, -2, 3'
args: ('Sulfuras', -1, 6) => 'Sulfuras, -2, 4'
args: ('Sulfuras', -1, 1) => 'Sulfuras, -2, 0'
args: ('Sulfuras, Hand of Ragnaros', 0, 0) => 'Sulfuras, Hand of Ragnaros, 0, 0'
args: ('Sulfuras, Hand of Ragnaros', 0, -1) => 'Sulfuras, Hand of Ragnaros, 0, -1'
args: ('Sulfuras, Hand of Ragnaros', 0, 50) => 'Sulfuras, Hand of Ragnaros, 0, 50'
args: ('Sulfuras, Hand of Ragnaros', 0, 51) => 'Sulfuras, Hand of Ragnaros, 0, 51'
args: ('Sulfuras, Hand of Ragnaros', 0, 5) => 'Sulfuras, Hand of Ragnaros, 0, 5'
args: ('Sulfuras, Hand of Ragnaros', 0, 6) => 'Sulfuras, Hand of Ragnaros, 0, 6'
args: ('Sulfuras, Hand of Ragnaros', 0, 1) => 'Sulfuras, Hand of Ragnaros, 0, 1'
args: ('Sulfuras, Hand of Ragnaros', 5, 0) => 'Sulfuras, Hand of Ragnaros, 5, 0'
args: ('Sulfuras, Hand of Ragnaros', 5, -1) => 'Sulfuras, Hand of Ragnaros, 5, -1'
args: ('Sulfuras, Hand of Ragnaros', 5, 50) => 'Sulfuras, Hand of Ragnaros, 5, 50'
args: ('Sulfuras, Hand of Ragnaros', 5, 51) => 'Sulfuras, Hand of Ragnaros, 5, 51'
args: ('Sulfuras, Hand of Ragnaros', 5, 5) => 'Sulfuras, Hand of Ragnaros, 5, 5'
args: ('Sulfuras, Hand of Ragnaros', 5, 6) => 'Sulfuras, Hand of Ragnaros, 5, 6'
args: ('Sulfuras, Hand of Ragnaros', 5, 1) => 'Sulfuras, Hand of Ragnaros, 5, 1'
args: ('Sulfuras, Hand of Ragnaros', 10, 0) => 'Sulfuras, Hand of Ragnaros, 10, 0'
args: ('Sulfuras, Hand of Ragnaros', 10, -1) => 'Sulfuras, Hand of Ragnaros, 10, -1'
args: ('Sulfuras, Hand of Ragnaros', 10, 50) => 'Sulfuras, Hand of Ragnaros, 10, 50'
args: ('Sulfuras, Hand of Ragnaros', 10, 51) => 'Sulfuras, Hand of Ragnaros, 10, 51'
args: ('Sulfuras, Hand of Ragnaros', 10, 5) => 'Sulfuras, Hand of Ragnaros, 10, 5'
args: ('Sulfuras, Hand of Ragnaros', 10, 6) => 'Sulfuras, Hand of Ragnaros, 10, 6'
args: ('Sulfuras, Hand of Ragnaros', 10, 1) => 'Sulfuras, Hand of Ragnaros, 10, 1'
args: ('Sulfuras, Hand of Ragnaros', -1, 0) => 'Sulfuras, Hand of Ragnaros, -1, 0'
args: ('Sulfuras, Hand of Ragnaros', -1, -1) => 'Sulfuras, Hand of Ragnaros, -1, -1'
args: ('Sulfuras, Hand of Ragnaros', -1, 50) => 'Sulfuras, Hand of Ragnaros, -1, 50'
args: ('Sulfuras, Hand of Ragnaros', -1, 51) => 'Sulfuras, Hand of Ragnaros, -1, 51'
args: ('Sulfuras, Hand of Ragnaros', -1, 5) => 'Sulfuras, Hand of Ragnaros, -1, 5'
args: ('Sulfuras, Hand of Ragnaros', -1, 6) => 'Sulfuras, Hand of Ragnaros, -1, 6'
args: ('Sulfuras, Hand of Ragnaros', -1, 1) => 'Sulfuras, Hand of Ragnaros, -1, 1'
```