# Technical specification: Dull Dave’s misadventure

## Language for program:

Python 3.6

## Intended interface for program:

Mac terminal | Windows Powershell (must not call bash/command\_prompt specific functions in-script)

## Program class objects definitions:

“Room”

(intended instances: bedroom, kitchen, bathroom, lounge)

“Player”

(1 instance per playthrough: player)

“Item”

(intended instances: broom, j-cloth, bedsheets)

## Note:

#find a scalable way to have rooms associated with each other to determine whether user can travel to them

## Program parent class method definitions:

Class -> “Room”

Class objects:

-room\_names (tuple)

A list comprised of strings for room names

Instance Objects:

-contents (list):

List all items currently present in room

methods:

-\_\_str\_\_():

Print to screen description of room surroundings

eventMethods(command):

Logic patterns for events. Takes argument and either triggers in-built command such as “inspect <room\_name>” or

Class -> “Player”

Class objects:

-cmd\_keywords (list)

Holds a list of string-only keywords which have to be typed by the user in order to operate a command

Instance Objects:

-current\_room (string)

Holds name of the current room player character is occupying.

Generated by using an index in Room.roomNames[]

-propeties\_dict (dict):

Dict with property values (mostly Boolean) which can be indexed into with a key name.

This is relevant for eventMethods in room (called when player enters command). i.e:

Print(player.propertiesList():)

* midName = “Michael”
* postEvent = False

methods:

-\_\_str\_\_():

For a given instance of player, outputs to screen a brief string of description about Dave

-input(subject, Boolean(T/F), data\_type, min\_no, max\_no)

-lookAround():

Provides information to the user about the current room when a command is issued. Should be dynamic in some instances where the user has performed actions upon the room

-inspectRoom (roomInst):

Provides the user information about a specific item that is specified as the subject of the ‘inspect’ command. Info gathered from \_\_str\_\_ method of room instance

-move(roomInstance):

move to a new room name specified by the user. Return an error message if the user cannot move to that room as not adjacent to current rooms

-moveList():

Provides the user with a list output to screen of current rooms they can travel to. Should be generated by room names as key in a dict, and a list of other rooms stored as a list in values (or could be numbers and have a nested dict)

Class -> “item”

instance objects:

-properties\_dict (dict):

A dict of properties which can be iterated through to check if conditions for event trigger are met, i.e:

print(Broom.propertiesList):

* canSweep = True
* weapon = False
* emitsLight = False
* inInventory = False
* carryable = True

These properties can be tested by event methods

methods

-\_\_str\_\_():

use in-built \_\_str\_\_ to output to player written about the information held

## Child class defintions:

“roomP”

overwritten methods:

-\_\_str\_\_():

Overwrite the existing string of description for a given room (post-event)

Non-overwritten methods:

-contents()

“playerPs”

Overwritten methods:

-\_\_str\_\_():

New desc. (Post event)

***All other methods not overwritten***

“itemP”

overwritten methods:

-\_\_str\_\_():

Item description output to screen (post-event)

-propertiesList():

Some item properties should be overwritten post-event.

## Extra-class methods:

gameStart(Room.propertiesList): - 1st function call at startup

-Introduces user to game

-Asks for a middle name input for Dullard Dave

-Prints the new full name in a welcome message

-Tells user about current room

cmdInterpret(command):

return = None (invalid command) | tuple: (cmd keyword (string), cmd subject(string)

takes input string, splits via spaces and inspects words individually.

Flags invalid command by printing to screen & then returns None.

## Commands:

-‘go <room\_name >’ – go to a selected room:

-room names must be in stored list (printed to user) of rooms available to the user.

-The adjacent\_room list (max 4 values) printed to user should be generated by algorithm/function which determines which rooms are adjacent to the room currently in place

-For each room accessible in the game there will be a room class which constructs elements such as a list of items present in the room, and adjacent rooms the user can travel to

-The item\_list for the room class must take into account and consequently remove any items which have been picked up or otherwise displaced from the room by the player

-‘inspect <subject/item name>’ – print more detail to screen about the item/room

-Item must be listed as present in the room. The list will be provided to the user by printing a list variable in-program called ‘item\_list’

-If the item or room does not have further information available simply print to user interface a standard message informing the user, such as “I don’t know what that does!”

-An attribute of the room object

- ‘grab <item\_name>’ – take the specified item from the current room environment

* **Critical:** this command must also add the selected item to user inventory also
* If the user’s inventory has reached a specified limit the game will fail this function and give an informing message, such as ‘sorry, you are carrying too many things!’
* Standard message will be displayed to screen if the item cannot be picked up

- ‘inventory’ | ‘inv’ – output a list to game interface which displays the current items the user is carrying

* Will have a maximum amount of items it can display (inventory limit)