Team 33

Name	Roll number	Contributions
Kunal Jain	2019111037	Summarizing the responsibilities of the major classes.
Siddhant Jain	2019111038	Code smells found along with their short description
P. Sahithi Reddy	2020121011	Narrative analyzing the original design,UML
Veeral Agarwal	2019114009	Overview,Bugs

Assignment 4

OVERVIEW

The code we are examining is a code to assemble a 2D terminal game that suddenly spikes in demand for the terminal utilizing python. In the following areas we will investigate every one of the classes in the diverse python documents, their duties, the general plan of the code, the bugs and code smells we found and the refactoring needed for a superior code. The features of the game include passing obstacles, player controlling movement with asdw keys, powerups set off with console inputs, points can be gathered , a monster that should be beaten by shooting shots at it.

ORIGINAL DESIGN:

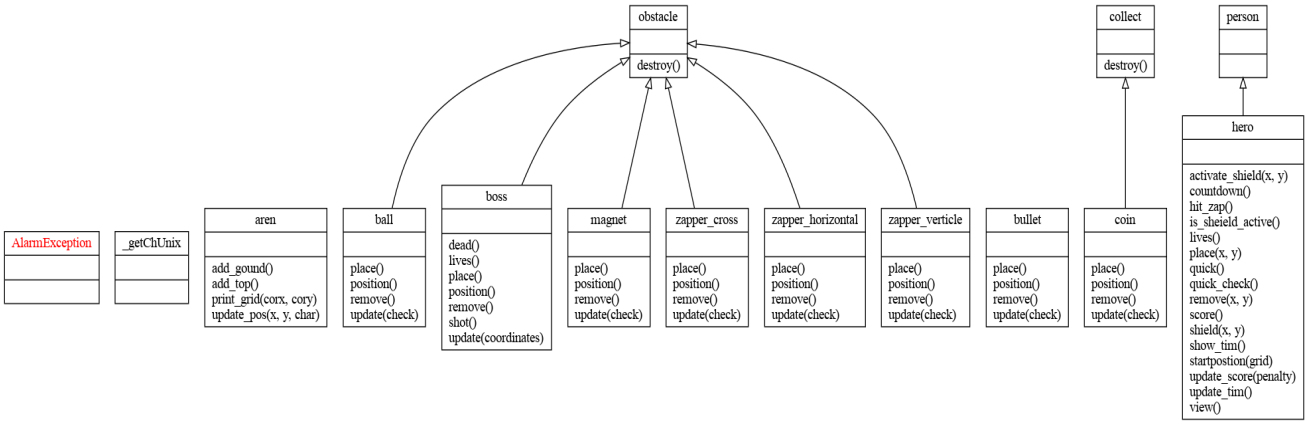
ANALYSIS , STRENGTH AND WEAKNESS:

The main file(start.py) contains major part of the code, most of the things are hard coded and repetitive, it would be tough to extend this game.In start.py file various other functions are declared, and objects have been initialised.We could group various functions based on their dependency on each other into various classes making new files for them.We can use other file to initialise all the objects for various classes.Few files were initialised but never used, they required functionalities are redeclared in start.py and were used.Almost 300 lines seem repetitive and are hardcoded, we should find a way to generalise them.Weakness of this original design is that is no modularity at all, they could’ve used classes to group similar objects and OOPS concepts for better modularity and readability of code, and there are no comments in the code.Strength is that almost everything is in one file and you don’t have to worry about files to

import, but this isn't to be considered as strength as it makes it difficult to extend.Few of the elements of the game were written in other files as classes, which are properly grouped, this can be considered as strength.

UML CLASS DIAGRAM

Here is the UML diagram according to the original design, as mentioned few classes were declared but not used.



RESPONSIBILITIES OF MAJOR CLASSES

Class	Responsibility
AlarmException	Raises exception when input is detected
aren	Outputs and handles the grid of the game
collect	Collectable objects in the game
coin	Handles generation, removal and updates of coins
_getChUnix	Class to take input
person	A movable entity in the game
hero	The main character of the game
bullet	The bullets that are shot by anyone
obstacle	Any fixed entity in the game world that is not

	moving in the game world
zapper_horizontal	Obstacle zappers that are horizontal in shape. Handles their placement, removal and updates
zapper_verticle	Obstacle zappers that are vertical in shape. Handles their placement, removal and updates
zapper_cross	Obstacle zappers that are cross shaped. Handles their placement, removal and updates
magnet	Handles the magnets that appear in the game and attract material
boss	The boss enemy, its placement, removal, lives, shooting and position throughout the game
ball	The balls in the game, their placement, removal and position throughout the game

CODE SMELLS

Code Smell Category	**Description**	
	Location	Issue
Long method	Start.py, function collision() (line 66)	Highly repetitive function spanning over 350 lines.
Extra functions	Start.py (424-441)	make_zappers_cross(), make_zappers_horizontal(), make_zappers_verticle() sould be merged into one function
Data clump	start.py	Too many global variables. Even though there's an "aren" class
Misplaced classes	obstacle.py	Magnet, boss, ball are not obstacles yet they are defined in the obstacle file
Duplicated code		Dimx and dimy have been defined for each class and they are needed to be the same. Changing them in only one place will break the code.

BUGS

Bug number	Short description
1) FileNotFoundError: [Errno 2] No such file or directory: './drag'	A file named `drag is missing, without this we cannot run the code, so we made an empty file named drag .
2) Even after this he/she didn't implement a clear screen. (start.py)	Because of this, in each frame, the new screen matrix is printed below the screen matrix which was in the previous frame .
3) Line 56 mandorian.py	Defined an extra variable (unused)
4) Line 675 start.py	Unused variable
5) Obstacle.py line 187,188,198,199	Defined an extra variable (unused)

PS: We couldn’t run the functionality for bugs as the game itself wasn’t running.Screen wasn’t getting cleared and elements were scattered.