**DASS ASSIGNMENT 2**
**Code Review and Refactoring**

**Team – 43:**

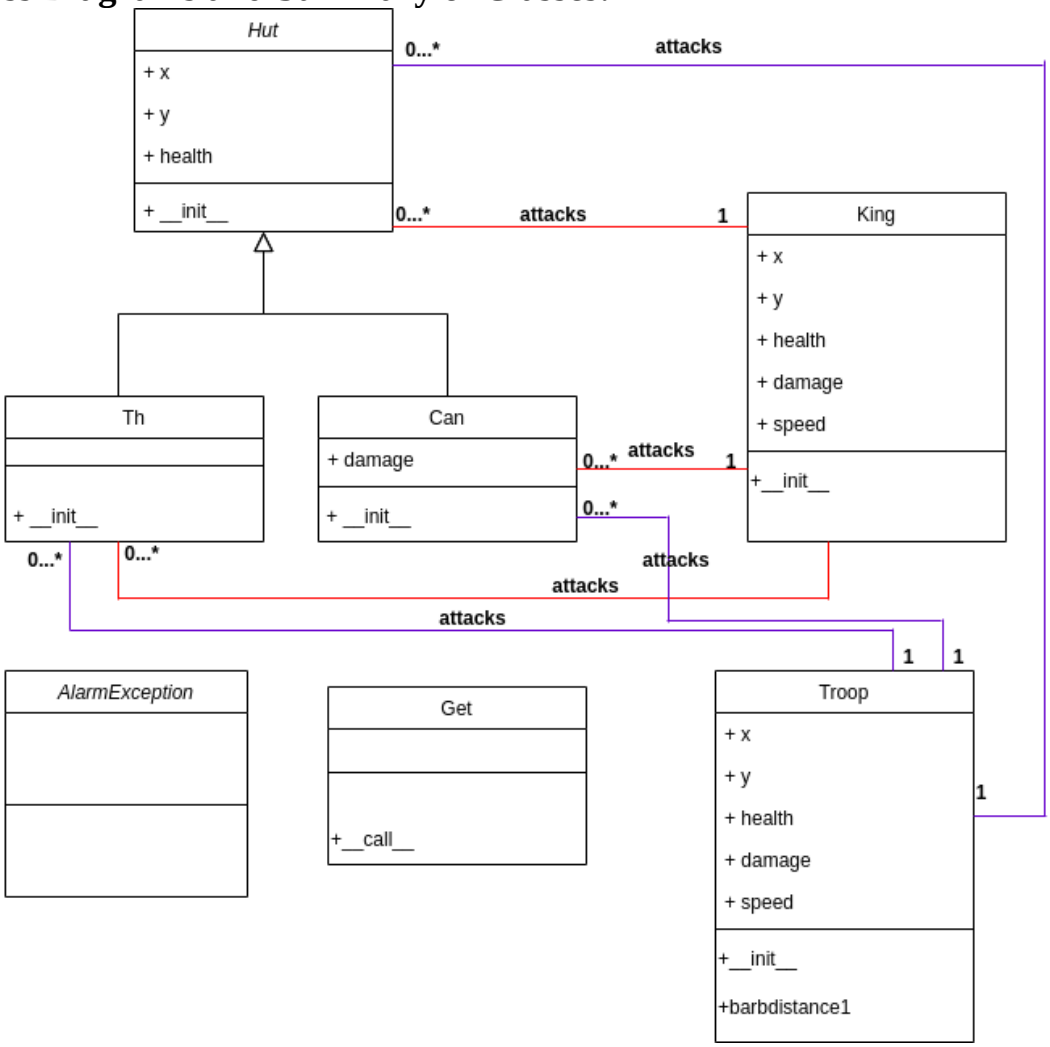
| **Name** | **Roll No** | **Task** |
|---|---|---|
| Bocha venkata krishna sai Anirudh | 2021101003 | Went through the assignment and explained the code and functionality of different components and found some code smells. |
| Darur Lakshmipathi Balaji | 2021114007 | Did an indepth study of code and found some codesmells and bugs. |
| Anush Anand | 2021101086 | Did UML diagrams by understanding code base and gave a clear summary of classes. |
| Mukta Chanda | 2021101071 | Did an indepth study of code and found some codesmells and bugs and helped in overview. |

**Overview:**

The software system we studied is a assignment done which is building clash of clans game with the specified basic features. The developer wrote program with less classes and many conditional statements which increased the length of code.

Developer organised classes like hut, townhall and cannon for buildings and troop and king for attacking titans. Health, damage co-ordinates of all the buildings and attacking titans are well described through code.

## UML Class Diagrams and Summary of Classes:




| Class | Responsibilities |
|---|---|
| Hut | Base class that represents a hut with coordinates and health |
| Th | Subclass of Hut that inherits its properties, with no additional responsibilities |
| Can | Subclass of Hut that inherits its properties and adds a damage attribute, representing a hut that can cause damage to other objects |

| Class | Responsibilities |
|---|---|
| Get | Class that defines a callable object to get user input from the terminal, using the termios and tty modules |
| AlarmException | An empty class |

| Class | Responsibilities |
|---|---|
| Troop | Represents a troop with x and y coordinates, health, damage, and speed attributes. It has a method `barbdistance1` that calculates the distance between the troop and other objects (h1, h2, h3, h4, h5, c1, c2, th1) and moves the troop towards the closest one based on a set of conditions. |

| Class | Responsibilities |
|---|---|
| King | Represents a character in a game with attributes such as position (x, y), health, damage, and speed. |

**Code Smells:**

| Code smell | Code snippet/File/Location | Description of Code Smell | Suggested refactoring |
|---|---|---|---|
| Combinatorial Explosion | 1) In barbdistance1 function in troop class and replay.py.<br><br>2)Function defense() in game.py and replay.py | 1)All the comparisions do the same thing – comparing for nearest hut or cannon or townhall.<br><br>2)Function defense() is repeating for c1 and c2 for all the army. | 1)Can create other function to find nearest hut or can or townhall than doing those many comparisions by passing things as parameters,<br><br>2)we can define a function in cannon which reduces health of king/troop by passing after checking distance. |
| Incomplete Library Class | The attack() function in game.py and replay.py | All the comparisions include king and other buildings. | It can be avoided by keeping the attack() function in class king |
| Long Method | The barbdistance1 function in troop class | It has repititive comparisions and the method is too long. | The method length can be reduced by dividing it into different individual functions. |
| Duplicated Code | 1) In game.py and | 1) For position shift | we can define a |

| | | | |
|---|---|---|---|
| | replay.py<br><br>2) In game.py and replay.py | of any troop or king<br><br>2) While printing the game on terminal a lot of lines of code is just duplicated. | function on passing the parameters which changes the value to 1 or 0 than repeating so many times.<br><br>2) It can be avoided by keeping multiple functions for similar type of printing. |
| Large Class | Class Troop in game.py and replay.py | It is due to the large method in class troop. | It can be optimized by re distributing the functionalities done in barbdistance1 function. |
| Comments (a.k.a. Deodorant) | In game.py and replay.py | The code snippets are commented which are used for debugging but not removed again and comments are brief. | More comments are to be added and the unrelated comments are to be removed. |
| Naming Inconsistency | 1) Class Th in hut.py<br><br>2) In game.py and replay.py. | 1) Th is townhall<br><br>2) The input is kept in a variable ‘p’ | 1) Rather than just Th townhall can be used as the class name.<br><br>2) For ease of reading the name can be changed to input...<br>Similarly many such naming inconcestencies are observed. |
| Data Clumps | In game.py and replay.py | The buildings and walls are lacking object orientation. | The walls and the town hall can be made into a class for better orientation and maintancace in future. |

| | | | |
|---|---|---|---|
| Using depreceated libraries | Import imp in game.py and replay.py | game.py:11: DeprecationWarning: the imp module is deprecated in favour of importlib; see the module's documentation for alternative uses<br>import imp | This can be avoided by choosing some alternative ways by using documentation. |
| Freeloader (a.k.a. Lazy Class) | 1) Class AlarmException in input.py<br><br>2) In game.py and replay.py | 1) It is unused as it just has pass<br><br>2) In this file is opened and closed with no reason | 1) It can be eliminated<br><br>2) It can be eliminated. |

**Bugs:**

| **Bug** | **Code snippet/File/Location** | **Description of Bug** | **Suggested refactoring** |
|---|---|---|---|
| In elif condition it should be >1 | In class troop in game.p y and replay.py | It has to be >1 as the above elif conditions which is leading to misfunctionality. | Should change the >2 to >1 in line 142 of game.py |
| Troop can also move in double speed along y axis. | In game.py and replay.py | The troop is not moving with double speed in y direction it is only possible in x-direction. | The g y direction should also be along y-direction. |
| The existence of x,y cell is not checked while moving troop | In game.py and replay.py | This may lead to segmetnation fault as the x,y cell may not exist when the troop moved. | Checking if x,y are in range is one possible solution. |
| The existence of x,y cell is not checked while moving king | In game.py and replay.py | This may lead to segmetnation fault as the x,y cell may not exist when the king moved. | Checking if x,y are in range is one possible solution. |

| | | | |
|---|---|---|---|
| On pressing ‘h’ – heal spell an extra condition is checked which is wrong according to the features asked. | In game.py and replay.py | According to this we cannot double the health of king or troops if it is not less than a particular value. | Removing the if conditon will do the job perfectly. |
| Colors of the buildings are not exactly matching with percentages of health. | In game.py and replay.py | The percentages for the buildings on which its colours are dependent are not followed as asked to. | This can be done by re evaluating the health values for displaying colors. |
| The defense function doesn’t work as expected | In game.py and replay.py | When simultaneously multiple troops are in the range of a cannon or cannons only one is effected and the order of the affect is also arbitary. | This can be fixed by removing elifs and only keeping if conditions. |

**Refactoring:**

- Incomplete Library class

Code snippets before refactoring:

```
def attack():

    if((abs(k.x - th1.x) <= 13) and (abs(k.y - th1.y) <= 6) and (th1.health > 0)):
        th1.health -= k.damage
    if(abs(c1.x - k.x) <= 10 and abs(k.y - c1.y) <= 5 and (c1.health > 0)):
        c1.health -= k.damage
    if(abs(c2.x - k.x) <= 10 and abs(k.y - c2.y) <= 5 and (c2.health > 0)):
        c2.health -= k.damage
    if(abs(h1.x - k.x) <= 10 and abs(k.y - h1.y) <= 5 and (h1.health > 0)):
        h1.health = h1.health - k.damage
    if(abs(h2.x - k.x) <= 10 and abs(k.y - h2.y) <= 5 and (h2.health > 0)):
        h2.health = h2.health - k.damage
    if(abs(h3.x - k.x) <= 10 and abs(k.y - h3.y) <= 5 and (h3.health > 0)):
        h3.health = h3.health - k.damage
    if(abs(h4.x - k.x) <= 10 and abs(k.y - h4.y) <= 5 and (h4.health > 0)):
        h4.health = h4.health - k.damage
    if(abs(h5.x - k.x) <= 10 and abs(k.y - h5.y) <= 5 and (h5.health > 0)):
        h5.health = h5.health - k.damage
```

```
def attack_1():
    k.attack(th1)
    k.attack(c1)
    k.attack(c2)
    k.attack(h1)
    k.attack(h2)
    k.attack(h3)
    k.attack(h4)
    k.attack(h5)
```

Code snippets after refactoring:

```
def attack(self, enemy):
    if((abs(self.x - enemy.x)<= 13 and (abs(self.y-enemy.y)<=6) and (enemy.health > 0))):
        enemy.health -= self.damage
        return True
    else :
        return False
```

- Combinatorial Explosion

Code snippets before refactoring:

```
def defense():
    if((abs(k.x - c1.x) <= 10) and (abs(k.y - c1.y) <= 5) and (k.health > 0)):
        k.health = k.health - c1.damage
    if((abs(k.x - c2.x) <= 10) and (abs(k.y - c2.y) <= 5) and (k.health > 0)):
        k.health = k.health - c2.damage
    elif((abs(b1.x - c1.x) <= 10) and (abs(b1.y - c1.y) <= 5) and (b1.health > 0)):
        b1.health = b1.health - c1.damage
    elif((abs(b3.x - c1.x) <= 10) and (abs(b3.y - c1.y) <= 5) and (b3.health > 0)):
        b3.health = b3.health - c1.damage
    elif((abs(b2.x - c1.x) <= 10) and (abs(b2.y - c1.y) <= 5) and (b2.health > 0)):
        b2.health = b2.health - c1.damage
    elif((abs(b1.x - c2.x) <= 10) and (abs(b1.y - c2.y) <= 5) and (b1.health > 0)):
        b1.health = b1.health - c2.damage
    elif((abs(b3.x - c2.x) <= 10) and (abs(b3.y - c2.y) <= 5) and (b3.health > 0)):
        b3.health = b3.health - c2.damage
    elif((abs(b2.x - c2.x) <= 10) and (abs(b2.y - c2.y) <= 5) and (b2.health > 0)):
        b2.health = b2.health - c2.damage
```

Code snippets after refactoring:

```
def defense():
    c1.defense(k)
    if(c2.defense(k)):
        pass
    elif(c1.defense(b1)):
        pass
    elif(c1.defense(b3)):
        pass
    elif(c1.defense(b2)):
        pass
    elif(c2.defense(b1)):
        pass
    elif(c2.defense(b3)):
        pass
    elif(c2.defense(b2)):
        pass
```

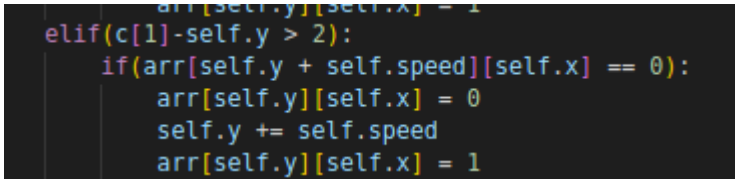
```
class Can(Hut):
    def __init__(self,x,y,health,damage):
        super().__init__(x,y,health)
        self.damage = damage

    def defense(self,enemy):
        if((abs(self.x - enemy.x)<= 10 and (abs(self.y-enemy.y)<=5) and (enemy.health > 0))):
            enemy.health -= self.damage
            return True
        else :
            return False
```

**Note:** The bug mentioned in the defence function is not fixed only the code smell is handled.

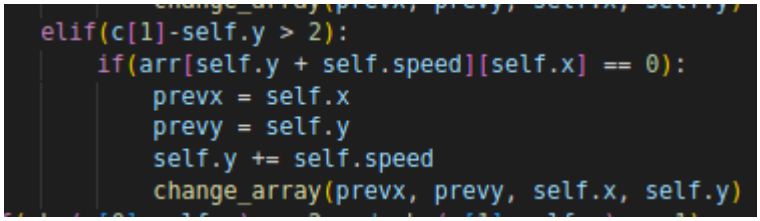
- Duplicated Code:

Code snippets before refactoring:



```
elif(c[1]-self.y > 2):
    if(arr[self.y + self.speed][self.x] == 0):
        arr[self.y][self.x] = 0
        self.y += self.speed
        arr[self.y][self.x] = 1
```

Code snippets after refactoring:



```
elif(c[1]-self.y > 2):
    if(arr[self.y + self.speed][self.x] == 0):
        prevx = self.x
        prevy = self.y
        self.y += self.speed
        change_array(prevx, prevy, self.x, self.y)
```