Poon King Hin,1155077526,khpoon6,khpoon6@cse.cuhk.edu.hk,csci3180 Assg2

Task 2: Demonstrating Advantages of Dynamic Typing

- For the first advantage, as some programming language using static typing, when passing parameter into function, it is necessary to initialize the type of the parameter:
  Eg: public Player(int healthCap, int mob, int posx, int posy, int index, SurvivalGame game)
  But for Dynamic Typing, it only care about the number of argument instead of the type, so the code will be more simple, and when there is some input that is unknown type, it is more easy to do the work with dynamic typing.
  E.g public Player(healthCap, mob, posx, posy, index, game)
  Moreover, it can use one function to handle different type of argument.
  Eg. Public handle_str_or_int(input):
  Try:
  Result=”the result is”+input
  Except:
  Result=”the result is”+ str(input)
  Return Result
- For the second advantage, when handling a huge set of data, it is possible to have different type of data, and it even have missing or incorrect data that in different type. If we using static typing, it is time consuming for checking the type of data. But in dynamic typing, it is easy to have a array have different type of value within different column, without a lot of checking and initialization. In python we can always use numpy or pandas to collect the mix type of data without a lots of code, and to do the machine learning.
  E.g
  df = pd.read_csv('imports-85.data',header=None,
  names=["symboling","normalized-losses","make","fuel-type","aspiration","num-of-doors","body-style","drive-wheels","engine-location","wheel-base","length","width","height","curb-weight","engine-type","num-of-cylinders","engine-size","fuel-system","bore","stroke","compression-ratio","horsepower","peak-rpm","city-mpg","highway-mpg","price"], na_values=("?"))
  df.info()

  Output: <class 'pandas.core.frame.DataFrame'>
  symboling 205 non-null int64
  …(different type of different features)
  dtypes: float64(11), int64(5), object(10)

So it is a pandas dataframe with different data type of features insides.

Task 3: Survival Game

```
Java:        public boolean positionOccupied(int randx, int randy) {
                 for (Object o : teleportObjects) {
                     if (o instanceof Player) {
                     Pos pos = ((Player) o).getPos();
                     if (pos.getX() == randx && pos.getY() == randy)
                         return true;
                 } else {    Pos pos = ((Obstacle) o).getPos();
```

```
                    if (pos.getX() == randx && pos.getY() == randy)
                        return true;} }
            return false;}
            for (Object obj : teleportObjects) {
                    if (obj instanceof Human)
                        ((Human) obj).teleport();
                    else if (obj instanceof Chark)
                        ((Chark) obj).teleport();
                    else if (obj instanceof Obstacle)
                        ((Obstacle) obj).teleport(); }
```

```
        for (Object obj : teleportObjects) {
                    if (obj instanceof Human)
                        ((Human) obj).teleport();
                    else if (obj instanceof Chark)
                        ((Chark) obj).teleport();
                    else if (obj instanceof Obstacle)
                        ((Obstacle) obj).teleport();
```

```
Python:     def positionOccupied(self,randx,randy):
                for item in self.teleportObjects:
                    pos=item.getPos()
                    if pos.getX() == randx and pos.getY() == randy:
                        return True
            return False
```

```
            for item in self.teleportObjects:
                item.teleport()
```

The above code are the two selected scenarios in which I think the Python implementation is better than the Java implementation.

The first one is the code of function **positionOccupied(),**as java doesn't have duck typing, so the three different object human, chark and obstacle have to set pos and check independently. But in python, it has duck typing, therefore they both can use setpos and teleport without stating the object type. Which make the code more easy to read and shorter.

The second one is also about duck typing, it is similar to the above one. Java need to check the type of the object and state the different type of the object one by one.

Task 4: For Glory!

As java doesn't have duck typing, as there are a new object Wand occurs, I have to include it in every function that use of it in human and chark:

```
if(index == game.getN()/2 – 1{      ((Wand)this.object).enhance();
Else                                ((Rifle)this.equipment).enhance();   }
```

Also as java does not have dynamic typing, it need to create 1 more variable now, which is wand:

protected Weapon equipment; protected Wand object;

But in python, it only need to include the first one in the initialize section, and doesn't need to create one more variable, it can be a equipment of wand, even though it is not a weapon.

So duck typing and dynamic typing reduce messy code with similar variable, and reduce complexity of the code have many different object that have a same object.