Keith Barry
Kevin Justich

1. Javadoc for classes
    i. Adventurer
        1. This class hold the methods that move the character around the house. It also holds methods that help the game interact with the player's inventory. This class also holds an ArrayList of Things that is the player's inventory.
            a. Inventory();
                i. /**
                *Description
                *This method prints out all the things in the user's
                *inventory.
                */
            b. Take();
                i. /**
                *@param takes in a thing to be added to the
                *inventory
                *Description
                *This method takes in a thing and adds it into the
                *player's inventory
                */
            c. Drop();
                i.
                /**
                *@param the string name of the item the
                * player wants to drop
                *Description
                *This method takes in a string and uses that to drop
                *thing out of the player's inventory
                */
            d. North();
                i.
                /**
                *Description
                *Moves the player north
                *
                */

e. East();
    i.
        /**
        *Description
        *Moves the player east
        *
        */
f.
g. South();
    i.
        /**
        *Description
        *Moves the player south
        */
h. West();
    i.
        /**
        *Description
        *Moves the player west
        */

ii. Rooms
    1. This class creates rooms that are populated with Things and given a descriptions. It holds an ArrayList of type Things which is the contents of a given room.

iii. TextAdventure
    1. This is the “runner” class of the text adventure game. It asks the player if they would like to play and if they do it runs the command method from the AdventureModel Class
        a. Main();
            i.
                /**
                *Description
                *This method asks the user if they would like to
                *play It also checks if the player won and if they can
                *move to the next level
                */

iv. Things
    1. The Things class holds all the item that the player can interact with in the game.. It also has a constructor that gives a Things a name description and a description when used
        a. Inspect();
            i. a

```
/**
*Description
*prints out the description of the item
*/
```

        b. Use();
            i.

```
/**
*Description
*This method prints out a little description of what
*happens when the item is used and set the used
*variable  to true to activate level completion or
* game completion
*/
```

v. AdventureModel
    1. This is the main logic class of the entire game. It interacts with all the other classes and basically created the game. The first thing it does it create the rooms and populate them with the Things that go in them. Then it takes commands from the user and uses that to call methods. This class also holds a 2d array that is the map of the game that the player can move around.
        a. roomDescription
            i.

```
/**
*Description
*This method prints out a description of the room
* that the player is in it uses the currentRoom
*variable to check what room they are in
*/
```

        b. Command

i.

/**
*Description
*This method is the bases of the whole game it
*allows players to input what they would like to do
* and then calls the methods that are pertinent
*/

c. Look();
   i.

/**
*Description
*This method gives the player a description of the
*room or an item based on their input
*/

d. ManipulateItem ();
   i.

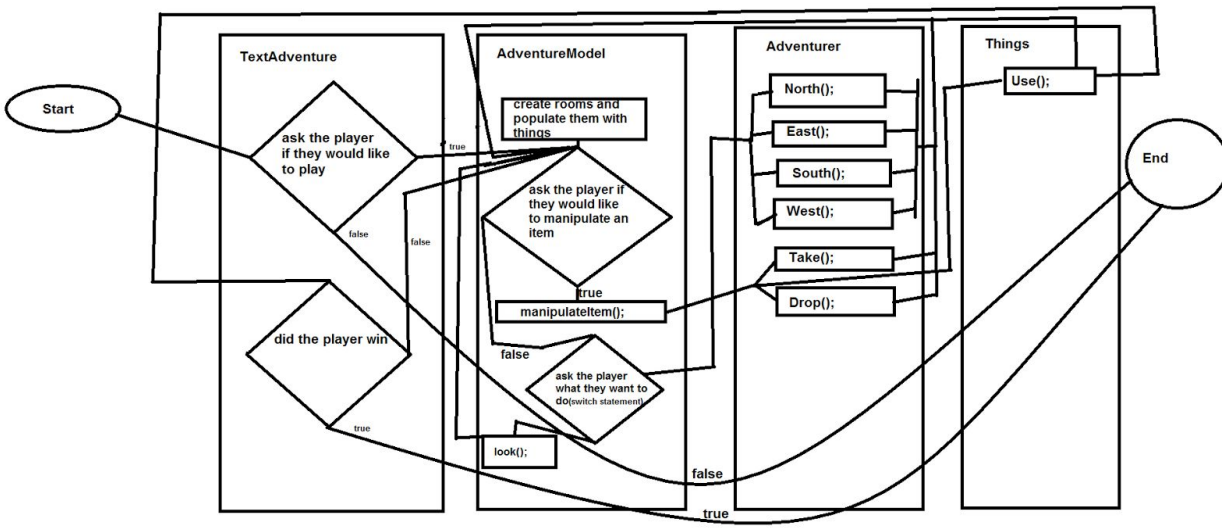
/**
*Description
*This method allows a player to add an item to their
*inventory, use an item, or drop an item out of their
*inventory.
*
*/

2. How classes interact
   i. TextAdventure
      1. AdventureModel
         a. The only other class that the textAdventure interacts with is the adventureModel class. The textAdventure class creates a textAdventure object and calls the command method.
      2.
   ii. AdventureModel
      1. Rooms
         a. AdventureModel interacts with room by creating a room object for each room and adding Things into it. AdventureModel also calls the description variable form the Rooms class when describing a room.
      2. Things
         a. AdventureModel interacts with the Things class by creating things objects and putting them in a ArrayList and putting

that into rooms. It also calls the use method in the Things class when the user wants to use an item.

3. Adventurer
    a. AdventureModel interacts with the Adventurer class by calling the methods to move the player around. The adventure model also calls methods from the Adventurer class when the user wants to interact with their inventory.

3. Control Flow



ROOM DESCRIPTION

Your cage- There is cage filling made from shredded paper. You catch a glimpse of one of the shreds and it is a shipping invoice for magic pellets (the food you grew up eating in Colombia). You make a decision to go after the food. You also see a laser pointer and toothpick on the floor of the cage. In the corner, there is generic hamster food and water.

Rabbit Cage- This room looks like an ordinary pet cage with filling, food, and water. There is an rabbit and a plastic bag lying on the floor. The rabbit wants to fight!

Cat Cage- The food and water bowls are empty. There is also a cat who must be very hungry. As a hamster, you have to be careful. There are also pet tranquilizer pills on the ground, as well as some puppy treats.
Puppy Pen- There are adorable puppies who want to play with you, because you look like a toy. There is a food bowl, water, and some animal droppings on the floor. You also notice a key in the corner.
Snapping Turtle Cage- This cage has rocks and shallow waters for the turtles. They immediately start snapping at you. You also notice some rat poison and an alka-seltzer tablet in the corner.
Gerbil Cartel Cage- Several of Andres' lackeys are waiting for you. They aren't going to let you get the magic pellets first. However, it looks like they are furiously eating. There is food, water, and something shiny on the floor. You also notice puppy treats.
Mouse Cage- There are several mice in this cage and a lot of mouse poop. There is an empty food bowl, and a water spout. The mice were promised a cut of Andres' loot. There is also a box cutter on the ground.
Guinea Pig Cage- The guinea pig cage is barren. No food, no water, and he looks a little unstable and unhinged. He hasn't had sleep in weeks. He looks like he wants to eat you. There is a plastic bag on the floor and some already-chewed gum.
Snake Cage- The snakes seem really interested in your adventure. They have food and water. There is a rocky terrain underneath a glow lamp. You also notice a toothpick on the ground. The snake doesn't want to hurt you, he just wants to get out of his cage, but can't because he doesn't have hands.
Checkout Counter- You have reached the magic pellets! However, Andres has been waiting for you. There is nothing else on the counter. Andres looks rather small- you can probably pick him up.

Commands-

a. Take
    i. This command adds the item that the player wants into their inventory
b. Look
    i. This command prints out the description of the room or the item that the player is interested in
c. Use
    i. Prints out what happens when the item is used and its effect on the room that it is in
d. Drop
    i. Removes the item that the player chooses from their inventory
e. Move north
    i. Moves the player one block north
f. Move south
    i. Moves the player one block south
g. Move east
    i. Moves the player one block east

h. Move west
    i. Moves the player one block west

ERROR HANDLING

Error Handling- There are several ways we will try to sanitize user input. At certain points, the game will only ask yes/no questions to limit the possibility of user error (ignore case). If anything else is entered, we will use try/catch to display an error message showing acceptable answers. If it is an open ended question, we will also use try catch to display appropriate answer. Another error we must be mindful of is using an item that does not defeat the animal you are facing. We will simply give an error message of how that item can't be used on this animal, in addition to a hint of what can be used to defeat the animal. Finally, if the use tries to walk off the map, we will have an error message saying they cannot walk in that direction, and display the next possible moves.

4. Miscellaneous
    a. Naming convention
        i. The main naming convention that we use for all our variables and methods is camel case. We try to make our methods and variables as descriptive as possible while still being brief so there is no misunderstanding in what their job is.
    b. Constructor for the Rooms class

```
public Rooms(String n, String d, ArrayList<Things> RoomContents)
{
    name=n;
    description=d;
    contents=RoomContents;
}
```