Home Tour Help:

Step 1: Read through the instructions carefully without trying to execute any of them.

Just try to grasp the overall format of the application and formalize any questions that you have.

Step 2: Create the classes.

So, you read the HomeTour app instructions and aren't sure where to go from here.

(If you have another idea that deviates from the instructions then reach out to your trainer. Creative approaches are encouraged, but you will want to make sure that your approach still covers the key concepts.)

If you are approaching the application in a way laid out by the instructions, but need some more assistance then proceed with this helper guide.

Create all of the classes from the initial instructions. (You don't need to fill them in just yet. We will go through them together.)

Fixture and ***Room*** in the ***fixtures*** package.
Main, ***Player*** and ***RoomManager*** in the ***game*** package.

Step 3: Understanding the structure.

Don't do anything just yet. Try to read through the description below in conjunction with the descriptions on the initial instructions and understand the flow of the application.

Main, Player and RoomManager are the classes that will allow you to set up and move through your game.

In Main, we will define our ***main method*** that starts the application. In this main method we will need to create a ***RoomManager object*** and a ***Player object***. We will use the *RoomManager to set up all of the Room objects in the game* and define the relationships between them (this means that *we will write the logic for* ***Room setup in the RoomManager*** *class* and then call upon it here in the main method).

We will also need a *game loop.* The loop should read in commands from the user and move the player through the house.

So, how do Room and Fixture fit in? Fixture is the general outline for all game items that can be interacted with (Room objects included). Thus, Room will extend Fixture. -- Room IS-A Fixture

Together the Room/Fixture classes will outline the characteristics that each Room/item should have. You will use this blueprint in your RoomManager class. (In RoomManager you will create the instances of each Room, i.e. a living room, foyer, dining room etc).

Step 4: Filling out the model.

Referring to the instructions, let's fill in Fixture (which should be an abstract class).

- Create an instance variable for each property.
- You don't need to assign them values just yet, but you should set up constructors and getters/setters that you can use later when you are creating objects that have an IS-A relationship with Fixture.

Then do the same for Room. This class should extend Fixture and it should set up appropriate constructors/getters and setters. Note- you don't need to populate the array of exits, but you should set up your array to hold your exits (In other words initialize Room[] exits with a new array of the appropriate size.)

- If you are unsure of what getters/setters and what constructors to include, think about how you will use this class.
    - Inside of the ***RoomManager*** you will create all the Room objects for your application. Saying something similar to this:

```
public void init(){
      ...
Room start = new Room("Foyer", "Grand entryway", "The entryway of a once grand home"
            + " that is well past its prime. Covered in cobwebs and dust, it is clear that "
            + "this entrance has not been used in a long time. Having once been lavish,"
            + "with a golden chandelier and marble floors, the room's chandelier now hangs"
            + "barely supported over the cracked and greying floor.");
      ...
}
```

    - Thus, it will be helpful to have a constructor that assigns the values for name, shortDescription, and long Description to the appropriate instance variables.

Note: if you write the constructor outlined in the instructions this constructor calls ***super with arguments. => You will need to have defined the corresponding constructor in Fixture.***

(`super(name, shortDescription, longDescription);`)

So in Fixture you would need:

```
public Fixture( String name, String shortDescription, String longDescription){
      this.name = name;
      this.shortDescription = shortDescription;
      this.longDescription = longDescription;
}
```

Now thinking about what getters and setters to define- you have options, but let's think again about ***how you will use these methods***.

In Main, you will have a method that will parse commands (turn commands into actions for your program) and let's say that one of the commands your program wants to handle is "go north".

Now we will assume the program has accessed the room your player is currently in and refers to it through a variable ***currentRoom***. Let's also say that your program has determined the user command wants to move your player to the next room that is north of currentRoom.

Then you may want your program to use a command like the following to actually move your player:

```
...
     Room nextRoom = currentRoom.getExit(direction);//direction holds "north" from the user
     player.setCurrentRoom(nextRoom);
...
```

This means that we need to write out the logic for getExit(direction) in the Room class.

```
public Room getExit(String direction) {
//use the direction to return a Room from the exits array
/*For example if I always save the south exit at position 0 in my exits array when I define my code in the first
place, then I could write out logic like the following.*/
      if(direction.equals("south")){
            return exits[0];
      }
/*If you decide to include a getter like this then you should write out the logic for the other directions you
want to support.*/
}
```

Step 5: Filling out the setup.

Open up RoomManager. Create instance variables for each property. Make sure to initialize your rooms array with a new array that is the length of the number of rooms in your application.

Then write out the logic of your init method. Inside this method, you need to make sure to create all of your Room objects and add them to each other's lists of exits and to the rooms array.

Step 6: Fill out Player.

The player class should include at least the single instance variable currentRoom.

Step 7: Main

Now you will outline the logic of your actual game. The first thing that your main method should do is something like this:

```
public static void main(String[] args) {
	/*Set up rooms*/
	RoomManager roomManager = new RoomManager();
	roomManager.init();
	/*Get starting Room from room manager*/
	Room startingRoom = roomManager.getStartingRoom(); //roomManager.startingRoom
	/*Create player and set up their currentRoom to be the startingRoom*/
	Player player = new Player(startingRoom);

	/*Game loop
		print Room
		get user command and translate it
		move Player/execute command
		repeat until quit */
}
```