ZorkGame Dokumentation M320

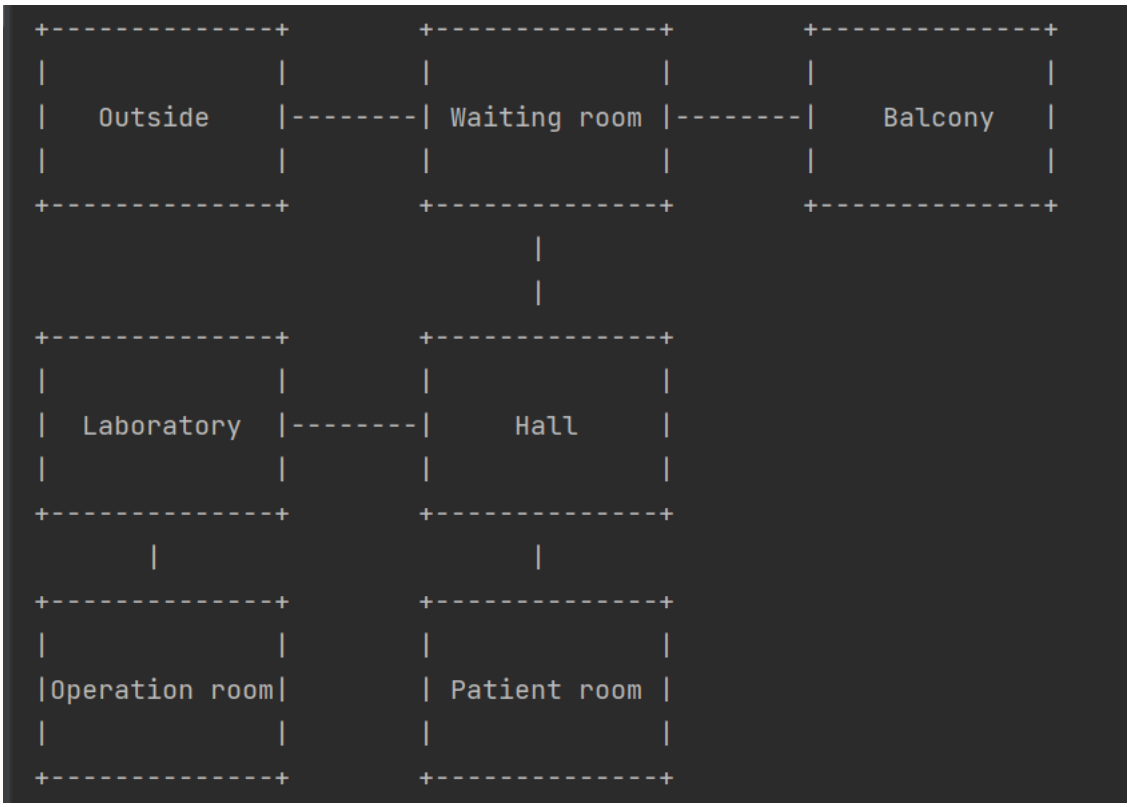
Inhalt

Spielidee

- In diesem Spiel ist man in einem Krankenhaus gefangen, wo man in jedem Raum Gegenstände finden kann um sich weiterzubringen
- Dabei geht es darum zum letzten Level anzulangen, um den finalen Boss zu erledigen
- Der Spieler startet Outside und kann mit den Befehlen «go quit help inv get say map» zu Recht finden

Spielmap

Das Krankenhaus hat sieben Räume, diese Räume sind mit Türen verbunden. In einigen von diesen Räumen findet man Gegenstände, welche von wichtiger Bedeutung sind.



Wie wird das Inventar ausgegeben?

Die Methode «printInventory()» zeigt dem Spieler alle Gegenstände an, die sich in seinem Inventar befinden. Wenn das Inventar leer ist, wird dem Spieler mitgeteilt, dass es keine Gegenstände gibt. Sie greift auf das Spielerobjekt zu und ruft die Methode «getInventory()» auf, um die aktuellen Gegenstände zu erhalten. Anschliessend wird über die Liste der Gegenstände iteriert, und jeder Gegenstand wird mit seinem Namen aufgelistet. Das Ergebnis wird auf der Konsole ausgegeben, um dem Spieler eine klare Übersicht über seinen Besitz zu bieten.

```
public void generateRandomItem() {
    String[] itemNames = {"Needle", "Scalpel", "Syringe", "Stethoscope",
"Tweezers", "Forceps", "Medical Hammer", "Blood Pressure Cuff",
"Thermometer"};

    String name = itemNames[random.nextInt(itemNames.length)];
    String description = "Description of " + name;
    double damage = random.nextDouble() * 10; // Adjust the damage range as
needed
    int durability = random.nextInt(10) + 1;
    double weight = random.nextDouble() * 2; // Adjust the weight range as
needed
    String[] perks = null;
    int hp = random.nextInt(10) + 1;

    // Create a new Item object
    Item randomItem = new Item(name, description, damage, durability,
weight, perks, hp);

    // Add the Item object directly to the player's inventory
    player.addToInventory(randomItem);

    System.out.println("You found a " + name + "!");
}
```

Wie wird ein Item erstellt?

Ich habe die Methode generateRandomItem() geschrieben, mit dem Gewissen einen zufälligen Gegenstand erstellen zu lassen. Zu diesem Zweck habe ich eine Liste von Gegenstandsamen erstellt, die in itemNames gespeichert sind.

Dann wähle ich einen zufälligen Namen aus dieser Liste aus, indem ich random.nextInt(itemNames.length) verwende. Dieser ausgewählte Name wird als name gespeichert, und ich erstelle eine Beschreibung für diesen Gegenstand, indem ich "Description of" an den Namen anhänge.

Um den Schaden (damage), die Haltbarkeit (durability) und das Gewicht (weight) des Gegenstands festzulegen, verwende ich zufällige Werte. Diese Werte werden zufällig generiert, um die Vielfalt der Gegenstände im Spiel sicherzustellen.Der Gegenstand hat keine speziellen Fähigkeiten (perks), daher setze ich diesen Wert auf null. Schliesslich bestimme ich

auch die Gesundheitspunkte (hp) des Gegenstands zufällig. Nachdem ich alle Eigenschaften des Gegenstands festgelegt habe, erstelle ich ein neues Item-Objekt namens randomItem, indem ich diese Eigenschaften übergebe. Schliesslich füge ich diesen Gegenstand direkt zum Inventar des Spielers hinzu, indem ich die Methode player.addToInventory(randomItem)

```
public void addToInventory(Item item) {
    if (inventory == null) {
        inventory = new Item[]{item};
    } else {
        // Check if the item is already in the inventory
        if (!Arrays.asList(inventory).contains(item)) {
            // Create a new array with one more slot for the new item
            Item[] newInventory = Arrays.copyOf(inventory, inventory.length
+ 1);
            newInventory[inventory.length] = item;
            inventory = newInventory;
        }
    }
}
```

aufrufe. Dies bedeutet, dass der Spieler diesen zufälligen Gegenstand erhalten wird. Abschliessend gebe ich eine Nachricht aus, die dem Spieler mitteilt, dass er einen Gegenstand gefunden hat. Diese Nachricht enthält den Namen des gefundenen Gegenstands.

```
public void generateRandomItem() {
   String[] itemNames = {"Needle", "Scalpel", "Syringe", "Stethoscope",
"Tweezers", "Forceps", "Medical Hammer", "Blood Pressure Cuff",
"Thermometer"};

   String name = itemNames[random.nextInt(itemNames.length)];
   String description = "Description of " + name;
   double damage = random.nextDouble() * 10; // Adjust the damage range as
needed
   int durability = random.nextInt(10) + 1;
   double weight = random.nextDouble() * 2; // Adjust the weight range as
needed
   String[] perks = null;
   int hp = random.nextInt(10) + 1;

   // Create a new Item object
   Item randomItem = new Item(name, description, damage, durability,
weight, perks, hp);

   // Add the Item object directly to the player's inventory
   player.addToInventory(randomItem);

   System.out.println("You found a " + name + "!");
}
```

Room

Meine Klasse Room sieht folgendermassen aus:

```
private void goRoom(Command command) {
   if (!command.hasSecondWord()) {
      System.out.println("Go where?");
   } else {
      String direction = command.getSecondWord();
      // Try to leave current room.
      Room nextRoom = currentRoom.nextRoom(direction);

      if (nextRoom == null)
         System.out.println("There is no door!");
      else {
         currentRoom = nextRoom;
         if (currentRoom.shortDescription() == "outside G block on
Peninsula campus") {
            System.out.println("Das Fenster ist offen, brrrrrrr");
         }
         System.out.println(currentRoom.longDescription());
      }
   }
}
```

Die Methode goRoom(Command command) ermöglicht es dem Spieler, sich im Spiel zwischen Räumen zu bewegen. Wenn der Spieler keinen zweiten Befehlsteil angibt, wird ihm gesagt, dass er angeben muss, wohin er gehen möchte.

Wenn der Spieler jedoch eine Richtung angibt, wird versucht, den aktuellen Raum zu verlassen und in den nächsten Raum in der angegebenen Richtung zu wechseln. Wenn es keinen gültigen Ausgang in die angegebene Richtung gibt, wird dem Spieler mitgeteilt, dass es keine Tür gibt.

Falls ein gültiger Ausgang gefunden wird, wechselt der Spieler in den nächsten Raum, und es wird eine spezielle Nachricht angezeigt, wenn der neue Raum "outside G block on Peninsula campus" ist. Anschließend wird die Beschreibung des neuen Raums ausgegeben, um dem Spieler die Umgebung im neuen Raum zu zeigen.

Diese Methode ermöglicht es dem Spieler, die Spielwelt zu erkunden und zwischen den Räumen zu navigieren.

Wie trägt der Spieler einen Gegenstand bei sich?

Ich habe die Methode addToInventory entwickelt, um die Fähigkeit zur Hinzufügung eines neuen Items zu einem bereits vorhandenen Inventar zu ermöglichen. Die Funktionsweise dieser Methode ist recht einfach:

Zunächst wird überprüft, ob das Inventar bereits existiert. Sollte es noch nicht initialisiert worden sein, erfolgt die Erstellung eines neuen Inventars, in dem das übergebene Item abgelegt wird.

Im Falle eines bereits vorhandenen Inventars, wird zunächst überprüft, ob das zu übergebene Item bereits in diesem Inventar enthalten ist. Sofern es noch nicht vorhanden ist, durchlaufe ich die folgenden Schritte:

Ein neues Inventar wird erzeugt, das um eine Position grösser ist als das bisherige Inventar.

Danach werden alle bereits vorhandenen Items aus dem aktuellen Inventar in das neu erstellte Inventar kopiert.

Schliesslich platziere ich das neue Item am Ende des neu erstellten Inventars.

```
public void addToInventory(Item item) {
    if (inventory == null) {
        inventory = new Item[]{item};
    } else {
        // Check if the item is already in the inventory
        if (!Arrays.asList(inventory).contains(item)) {
            // Create a new array with one more slot for the new item
            Item[] newInventory = Arrays.copyOf(inventory, newLength: inventory.length + 1);
            newInventory[inventory.length] = item;
            inventory = newInventory;
        }
    }
}
```

Item ablegen

Sobald der Spieler «X» sucht, suchen wir im Inventar nach diesem Gegenstand und im Fall, dass es das im Inventar gibt löschen wir es anhand von der Methode «removeFromInventory()»

```
public void removeFromInventory(Item item) {
    if (inventory != null) {
        for (int i = 0; i < inventory.length; i++) {
            if (inventory[i] == item) {
                // Create a new array with one less slot to remove the item
                Item[] newInventory = new Item[inventory.length - 1];
                for (int j = 0, k = 0; j < inventory.length; j++) {
                    if (j != i) {
                        newInventory[k] = inventory[j];
                        k++;
                    }
                }
                inventory = newInventory;
                break; // Item found and removed, exit the loop
            }
        }
    }
}
```

```
    case "drop":
        if (command.hasSecondWord()) {
            String indexOrName = command.getSecondWord();
            if (isNumeric(indexOrName)) {
                // If it's a number, try to drop the item by index
                int index = Integer.parseInt(indexOrName) - 1; // Adjust index to start from 0
                if (index >= 0 && index < player.getInventory().length) {
                    Item itemToDrop = player.findItemByIndex(index);
                    player.removeFromInventory(itemToDrop);
                    System.out.println("You dropped the " + itemToDrop.getName());
                } else {
                    System.out.println("Invalid item index. Check your inventory and try again.");
                }
```