Feature: Countable Items

Most items (at least the ones that you can pick up) in a game are unique. You might have one candle, one sword, and one gold key. Sometimes there are two or three of an item, and they can work the same way. But a *countable item* is a type of item that has many copies in the game, like money. If the player is carrying around 27 dollars, when we print their inventory we do not want to see "dollar" listed 27 times. Instead, 27 dollars will be stored as a single object that represents all 27 of them.

Implementation

1. Create a `CountableItem` class that extends `Item`, with the following features:
   (a) A field to store how many of the item are in this group of them.
   (b) Appropriate constructors, getters, and setters.
   (c) Any methods from `Item` that should be overridden.
   (d) A method with the following signature:

```
1  /**
2   * Splits a CountableItem into two groups.
3   *
4   * @param amountToTake How many to move into the second group.
5   * @return A new CountableItem containing some of the items that
        were in this.
6   *
7   * For example, suppose that we had a CountableItem containing 5
        dollars and we called this method with the parameter 3.
8   * The amount for this CountableItem would be reduced to 2 and it
        would return a new CountableItem with the same name/
        description/etc. and an amount of 3.
9   */
10  public CountableItem split(int amountToTake)
```

2. Update the `addItem` methods in `Room`, `Player`, and `Container`. When adding a normal `Item`, these methods should work the same way they do now. When adding a `CountableItem`, they should check whether or not another `CountableItem` with the same name exists. If so, instead of adding the new item it should combine the two by adding the amount of the new item to the existing item.

3. Be sure to test your project thoroughly and check your changes into Github.