**OOADJ Lab 5 Hackerrank challenge**

1. **Method Overloading:**

Write a program where users can perform addition operation on the given input. The program should be able to handle different types of data types. Use method overloading and static binding concepts in Java to implement the addition operation for the program. When users input two values, the application determines their data types and performs addition accordingly.

**Input Format**

The input consists of two lines with either an integer, decimal, or string.

**Constraints**

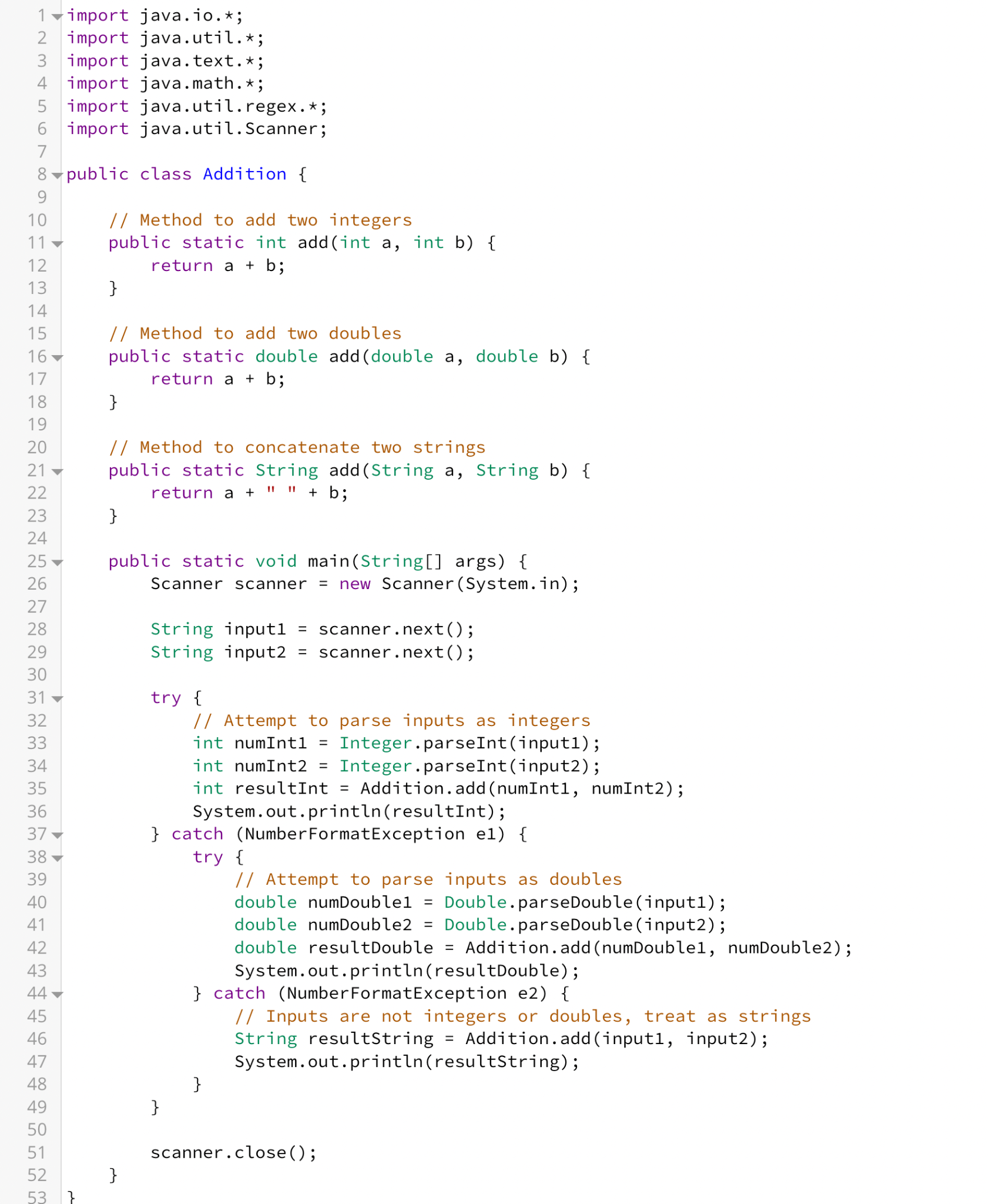
a < 10^9

b < 10^9

**Output Format**

Print the output in a single line.

**Code:**



**Test Cases:**

Test cases test the code on different data types, an intermixing of data types and null values. Students must handle addition in case of int and float and concatenation in case of strings.



1. **Constructor overloading**

Problem Statement: You are tasked with developing a character creation system for a video game. The system should allow players to create their in-game characters with different attributes. Each character is defined by their name, class, and level. Develop a Java program that demonstrates the use of constructor overloading to create character objects with different sets of attributes.

Requirements:

Implement a Character class with the following attributes: name (String): representing the name of the character. characterClass (String): representing the class or profession of the character (e.g., warrior, mage, archer). level (int): representing the level of the character. Implement constructor overloading to initialize character objects according to their respective inputs. Create a default constructor class where characters without roles default to "Unknown" and level defaults to 0.

**Input Format**

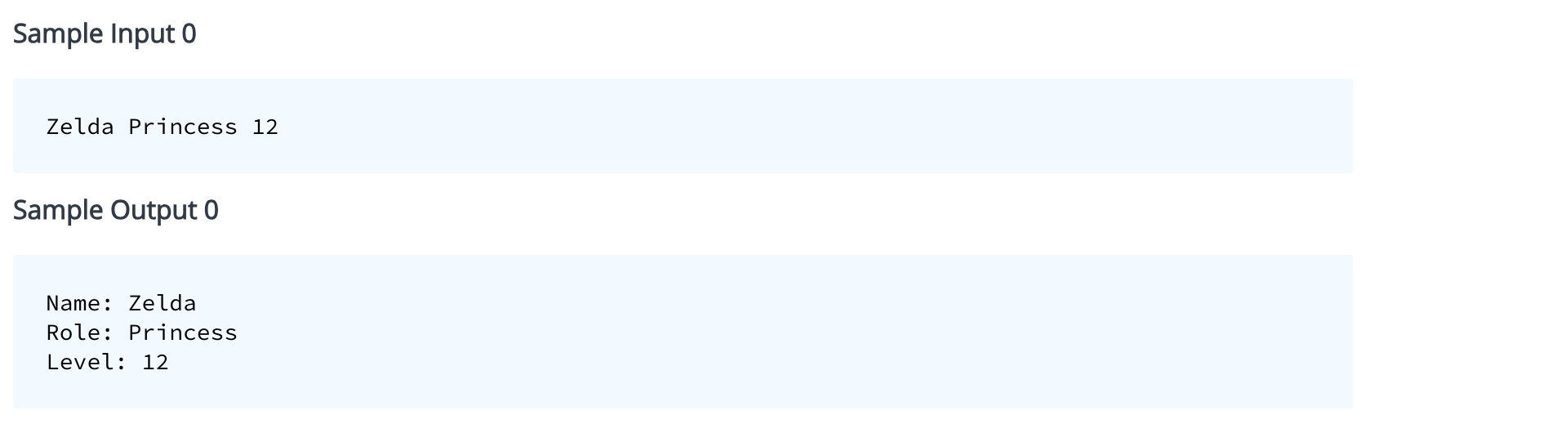
The input consists of a single line in which the name, role and level are given separated by whitespace.

**Constraints**

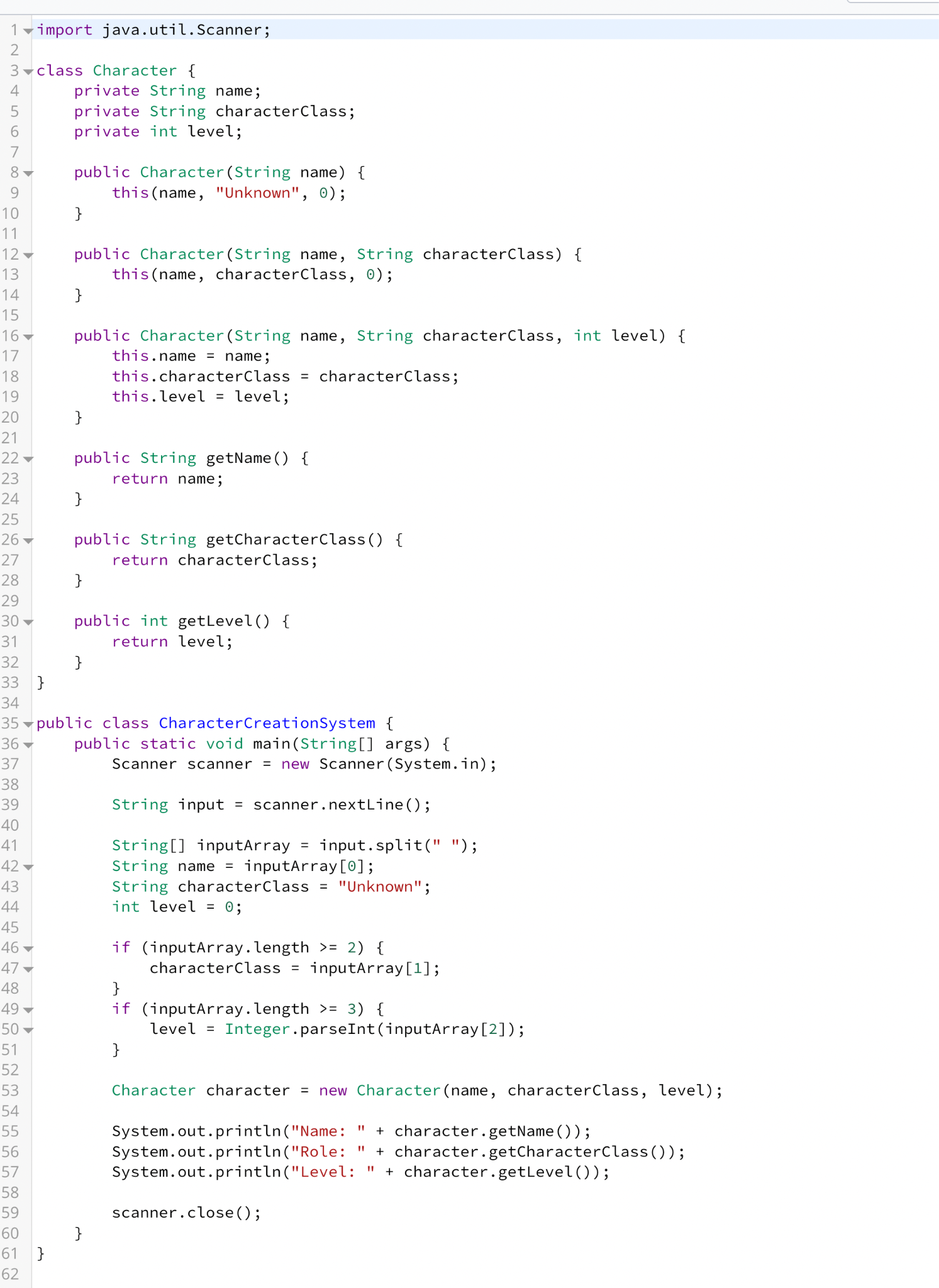
Make use of constructor overloading. Follow the specific output formatting to clear the test cases.

**Output Format**

The output consists of three lines that display the given information accordingly.

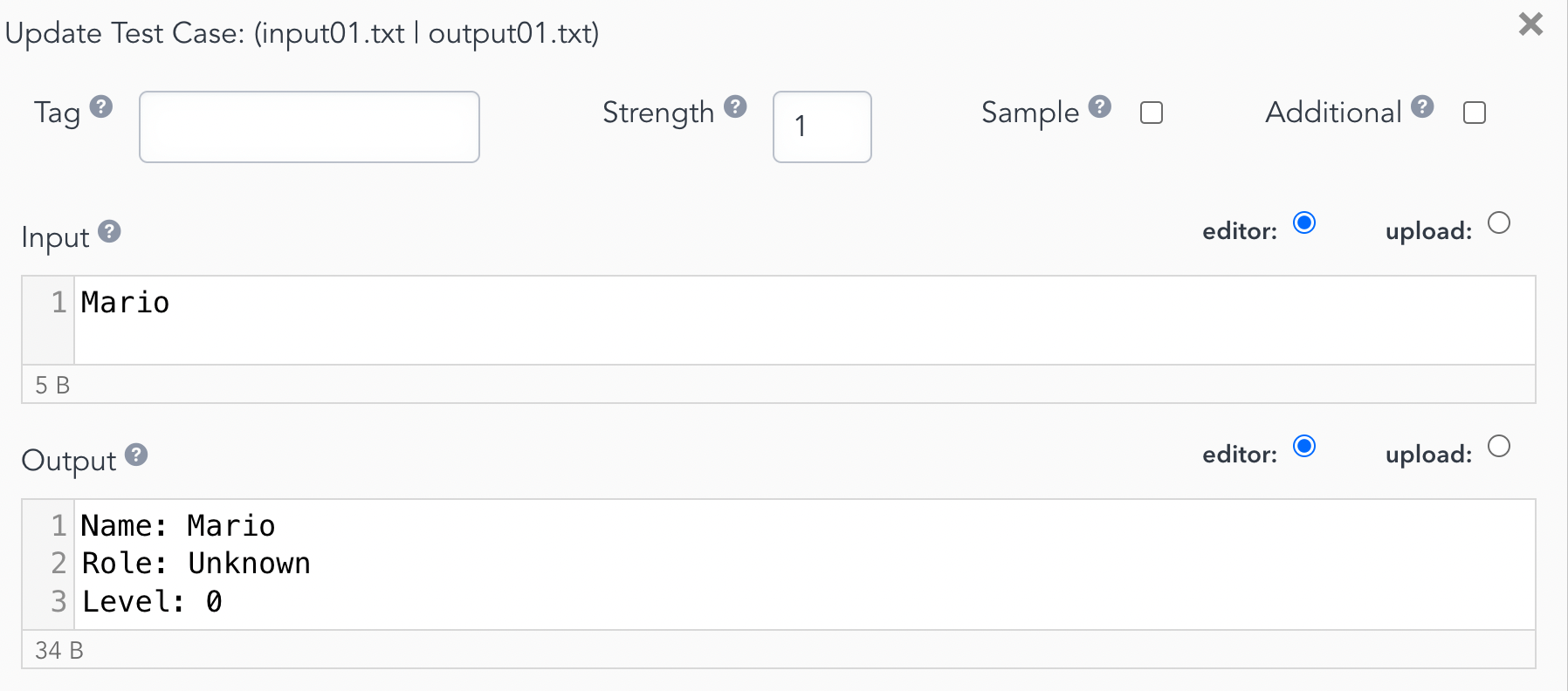


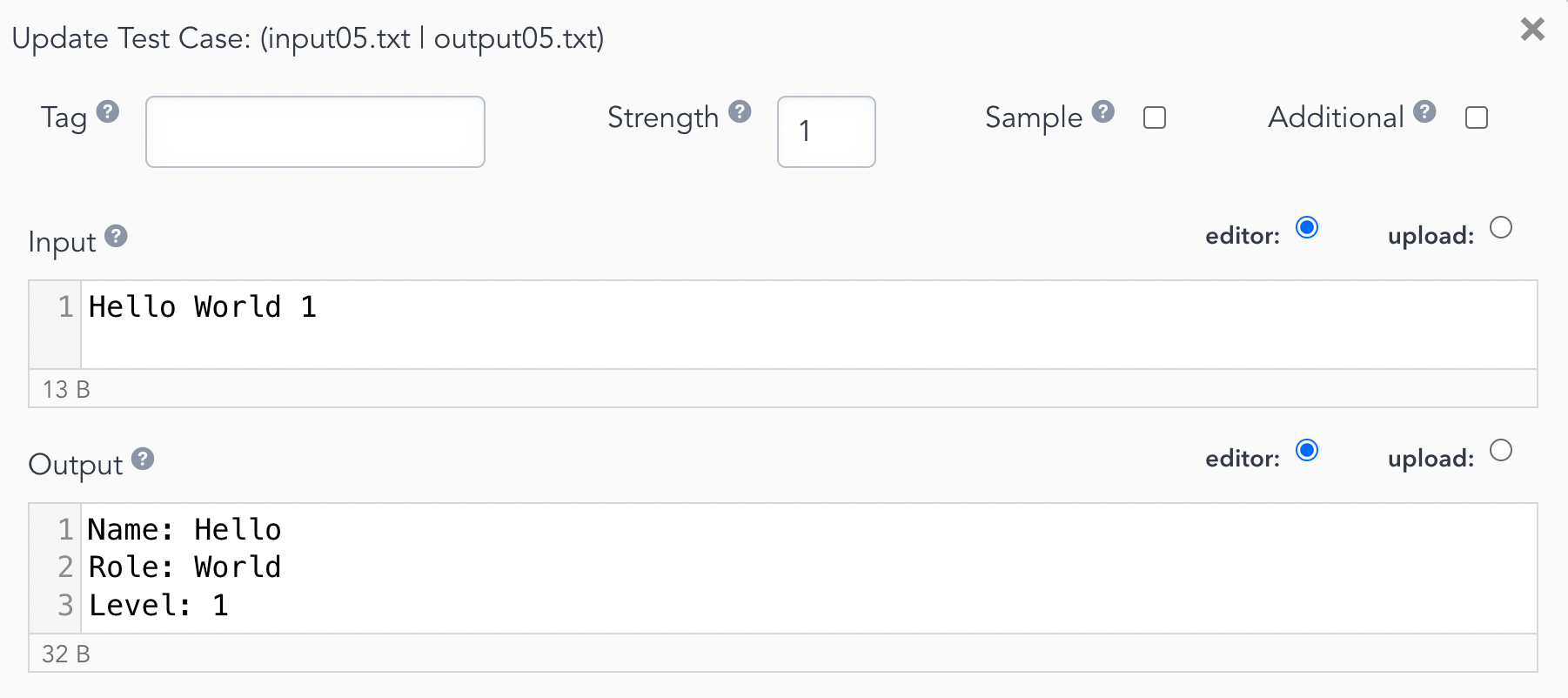
**Code**:



**Test cases:**

The program is tested for varying numbers of parameters, as the concept is constructor overloading on the basis of parameters, as well as null values.

****

****

1. **Composition Aggregation**

Design a simple game system in Java that allows players to pick up and manage items in their inventory. The system should demonstrate the use of object-oriented principles such as composition and aggregation to model the relationships between the player character, inventory, and items.

The Item class represents individual items in the game world. These items are aggregated into the player's inventory, as the inventory contains a collection of Item objects.

The Inventory class contains a collection of Item objects, representing the items that the player possesses. This demonstrates aggregation, as the inventory "has a" collection of items.

The Player class represents the main character in the game. It contains attributes such as health points and an inventory using composition.

**Input Format**

The input is in the form of a single line with two strings separated by whitespace, the first string represents the item type, the second represents the item name.

**Constraints**

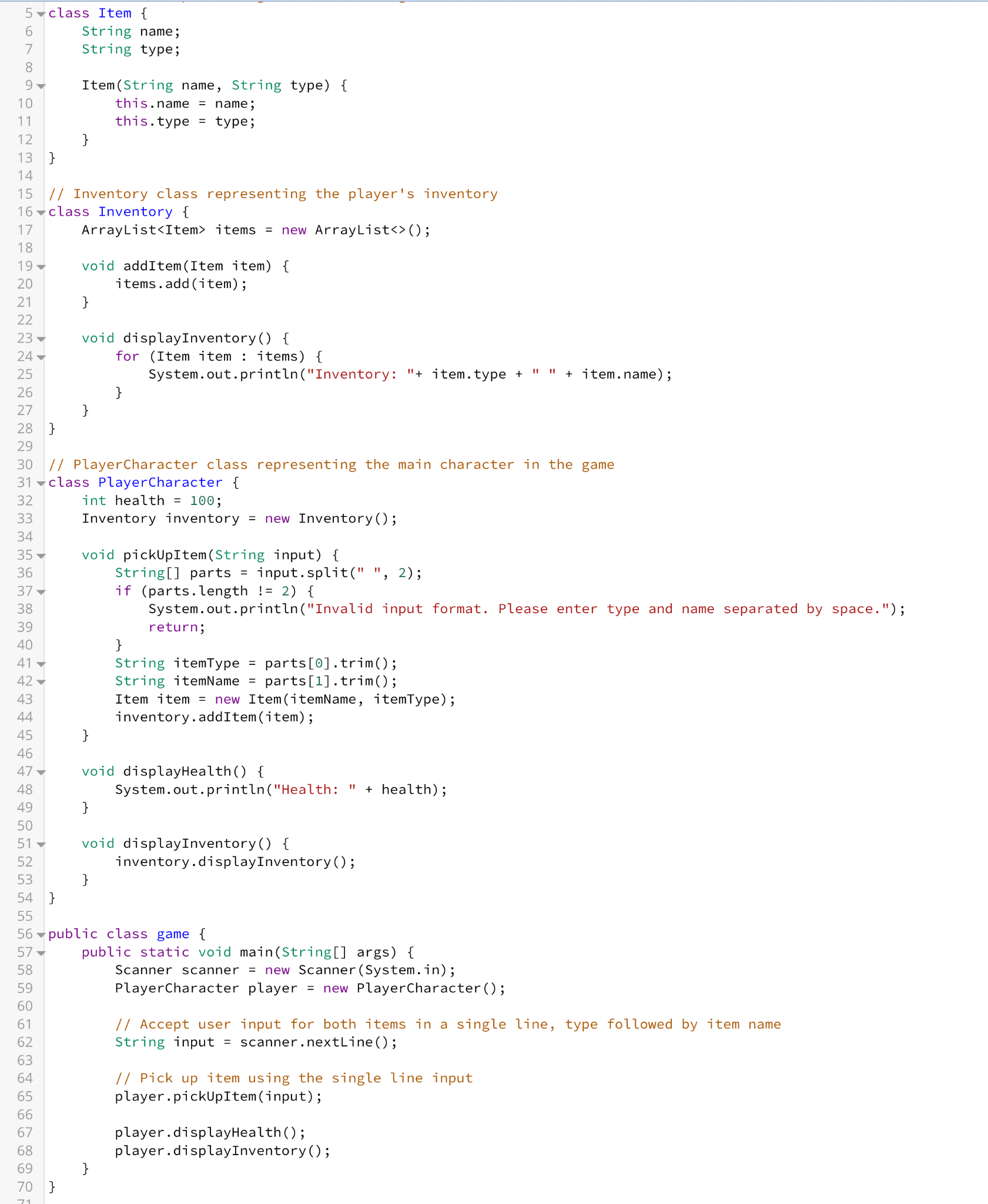
* Make use of composition and aggregation concepts while writing the program.
* Set Health: 100 within the program.
* Input is taken in the format (item type) (item name).
* Follow the specific output formatting to clear the test cases.

**Output Format**

The output should be in the form of two lines. The first line displays the health of characters which should be set to 100. The second line displays the inventory in the order (item type) (item name)



**Code**:



**Test cases:**

The program is tested for inputs of different data types and null values and invalid input types. 