Documentation: Implementation of the "Prototype" Pattern

1. Project Structure

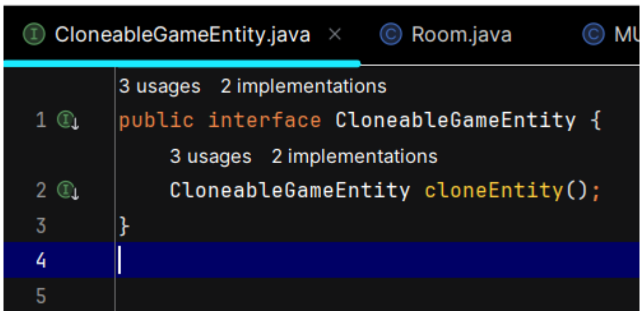
The project includes the following files:

- **CloneableGameEntity.java** – an interface for cloning objects.
- **Room.java** – a room class implementing the Prototype pattern.
- **NPC.java** – a non-playable character (NPC) class implementing the Prototype pattern.
- **MUDPrototypeDemo.java** – a demo class demonstrating the Prototype pattern in action.

2. Implementation of the "Prototype" Pattern in Code

2.1. CloneableGameEntity Interface

This interface defines the cloneEntity() method, which will be implemented in the Room and NPC classes.

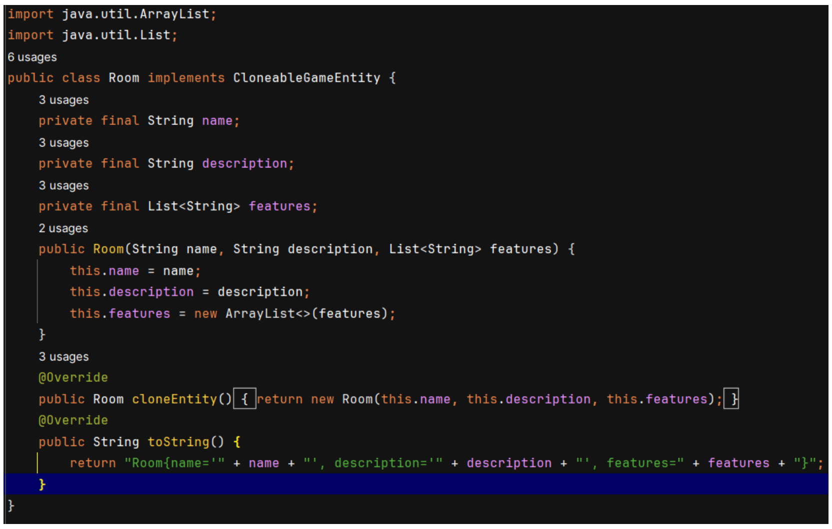


```
CloneableGameEntity.java ×    Room.java    MU
3 usages  2 implementations
public interface CloneableGameEntity {
    3 usages  2 implementations
    CloneableGameEntity cloneEntity();
}

```

2.2. Room Class

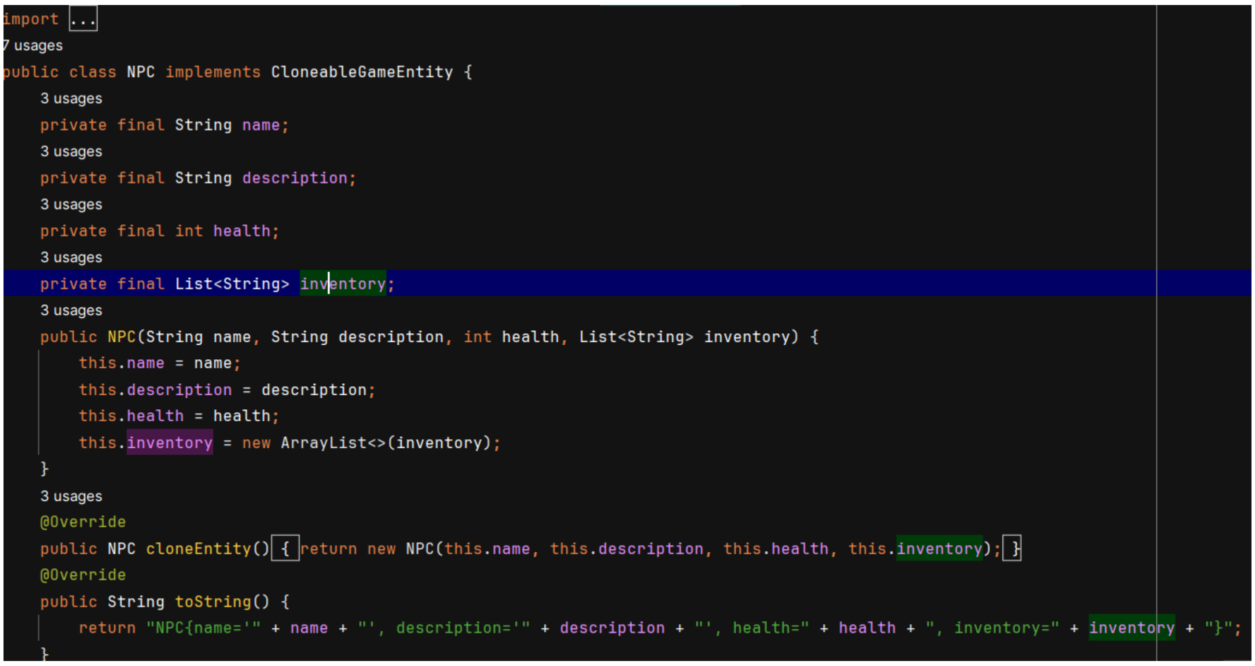
This class represents a game room. It contains a list of features, which is deep-copied.



```
import java.util.ArrayList;
import java.util.List;
6 usages
public class Room implements CloneableGameEntity {
    3 usages
    private final String name;
    3 usages
    private final String description;
    3 usages
    private final List<String> features;
    2 usages
    public Room(String name, String description, List<String> features) {
        this.name = name;
        this.description = description;
        this.features = new ArrayList<>(features);
    }
    3 usages
    @Override
    public Room cloneEntity() { return new Room(this.name, this.description, this.features); }
    @Override
    public String toString() {
        return "Room{name='" + name + "', description='" + description + "', features=" + features + "}";
    }
}
```

2.3. NPC Class

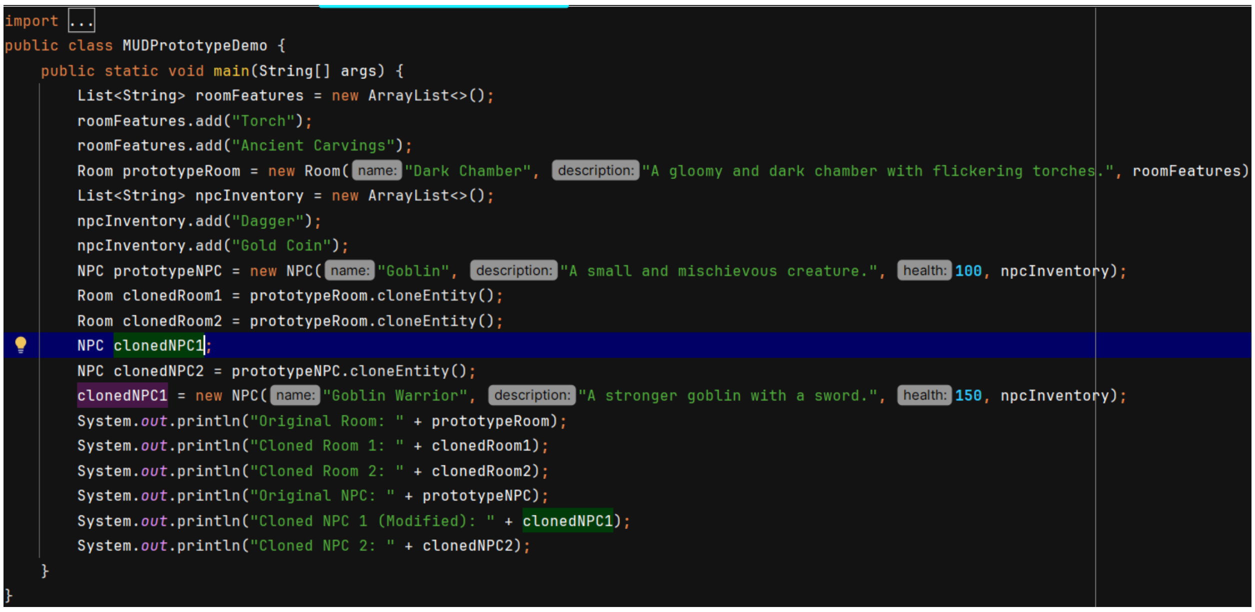
This class represents a game character. It contains health and an inventory, which are also deep-copied.



```java
import ...
7 usages
public class NPC implements CloneableGameEntity {
    3 usages
    private final String name;
    3 usages
    private final String description;
    3 usages
    private final int health;
    3 usages
    private final List<String> inventory;
    3 usages
    public NPC(String name, String description, int health, List<String> inventory) {
        this.name = name;
        this.description = description;
        this.health = health;
        this.inventory = new ArrayList<>(inventory);
    }
    3 usages
    @Override
    public NPC cloneEntity() { return new NPC(this.name, this.description, this.health, this.inventory); }
    @Override
    public String toString() {
        return "NPC{name='" + name + "', description='" + description + "', health=" + health + ", inventory=" + inventory + "}";
    }
```

2.4. Demonstration Class: MUDPrototypeDemo

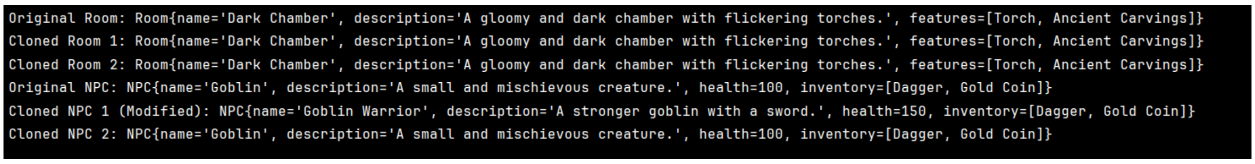
In this class, prototypes of a room and an NPC are created, cloned multiple times, and one of the clones is modified to confirm the independence of objects.



```java
import ...
public class MUDPrototypeDemo {
    public static void main(String[] args) {
        List<String> roomFeatures = new ArrayList<>();
        roomFeatures.add("Torch");
        roomFeatures.add("Ancient Carvings");
        Room prototypeRoom = new Room( name: "Dark Chamber",  description: "A gloomy and dark chamber with flickering torches.", roomFeatures)
        List<String> npcInventory = new ArrayList<>();
        npcInventory.add("Dagger");
        npcInventory.add("Gold Coin");
        NPC prototypeNPC = new NPC( name: "Goblin",  description: "A small and mischievous creature.",  health: 100, npcInventory);
        Room clonedRoom1 = prototypeRoom.cloneEntity();
        Room clonedRoom2 = prototypeRoom.cloneEntity();
        NPC clonedNPC1;
        NPC clonedNPC2 = prototypeNPC.cloneEntity();
        clonedNPC1 = new NPC( name: "Goblin Warrior",  description: "A stronger goblin with a sword.",  health: 150, npcInventory);
        System.out.println("Original Room: " + prototypeRoom);
        System.out.println("Cloned Room 1: " + clonedRoom1);
        System.out.println("Cloned Room 2: " + clonedRoom2);
        System.out.println("Original NPC: " + prototypeNPC);
        System.out.println("Cloned NPC 1 (Modified): " + clonedNPC1);
        System.out.println("Cloned NPC 2: " + clonedNPC2);
    }
}
```

3. Program Output

Console output:



```
Original Room: Room{name='Dark Chamber', description='A gloomy and dark chamber with flickering torches.', features=[Torch, Ancient Carvings]}
Cloned Room 1: Room{name='Dark Chamber', description='A gloomy and dark chamber with flickering torches.', features=[Torch, Ancient Carvings]}
Cloned Room 2: Room{name='Dark Chamber', description='A gloomy and dark chamber with flickering torches.', features=[Torch, Ancient Carvings]}
Original NPC: NPC{name='Goblin', description='A small and mischievous creature.', health=100, inventory=[Dagger, Gold Coin]}
Cloned NPC 1 (Modified): NPC{name='Goblin Warrior', description='A stronger goblin with a sword.', health=150, inventory=[Dagger, Gold Coin]}
Cloned NPC 2: NPC{name='Goblin', description='A small and mischievous creature.', health=100, inventory=[Dagger, Gold Coin]}
```