**Documentation**

1. Assignment 1: Implement a Builder for a Complex Object

* create a Builder interface with needed methods including build():

public interface IDungeonBuilder {  
 IDungeonBuilder setName(String name);  
 IDungeonBuilder addRoom(Room room);  
 IDungeonBuilder addNPC(NPC npc);  
 Dungeon build();  
}

* create a Builder class that implements our interface. Add atributes and constructor:

public class SimpleDungeonBuilder implements IDungeonBuilder {  
 private Dungeon dungeon;  
 private List<Room> rooms = new ArrayList<>();  
 private List<NPC> npcs = new ArrayList<>();

start with blank Dungeon

public SimpleDungeonBuilder() {  
 this.dungeon = new Dungeon();  
 }

* override all methods:

public IDungeonBuilder setName(String name) {  
 dungeon.setName(name);  
 return this;  
}

this method calls setName for dungeon object and assigns the provided name to it. Then returns current instance of IdungeonBuilder (this) which allows method chaining

public IDungeonBuilder addRoom(Room room) {  
 dungeon.addRooms(room);  
 return this;  
}  
public IDungeonBuilder addNPC(NPC npc) {  
 dungeon.addNpcs(npc);  
 return this;  
}

same with these methods

public Dungeon build() {  
 return dungeon;  
}

this method returns our built Dungeon

* then create a demo and a bulder in it:

public class MUDBuilderDemo {  
 public static void main(String[] args) {  
 IDungeonBuilder builder = new SimpleDungeonBuilder();

* create a room and a NPC:

Room room1 = new Room("room","some random room");  
NPC npc = new NPC("npc");

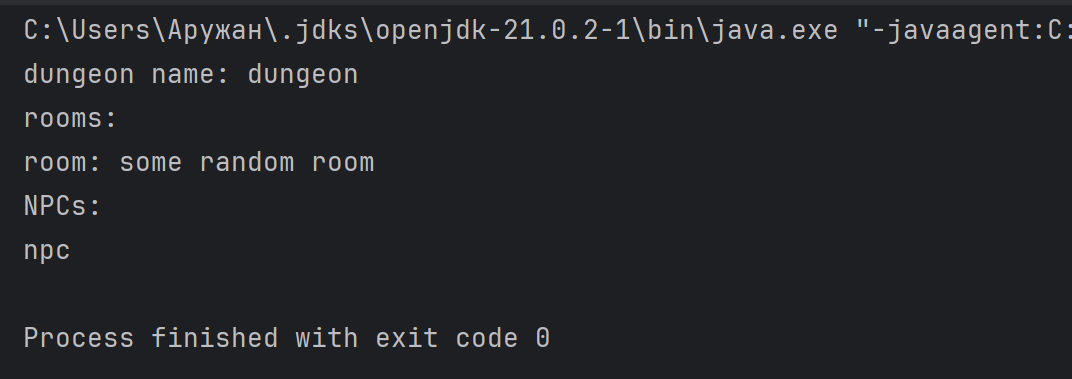
* configure all methods:

Dungeon dungeon = builder.setName("dungeon").addRoom(room1).addNPC(npc).build();

* show the result:

dungeon.show();

* run the code:



2. Assignment 2: Implement a Prototype for Cloning Objects

* first create a cloneable interface

public interface CloneableGameEntity {  
 CloneableGameEntity cloneEntity();  
 void show();  
}

* then create entities implementing CloneableGameEntity

public class Room implements CloneableGameEntity {  
 private String name;  
 private String description;  
  
 public Room(String name, String description) {  
 this.name = name;  
 this.description = description;  
 }

* getters:

public String getName() {  
 return name;  
}  
  
public String getDescription() {  
 return description;  
}

* method for cloning. It returns new Room with exact same attributes

@Override  
public Room cloneEntity() {  
 return new Room(this.name,this.description);  
}

* method to show info

@Override  
public void show() {  
 System.*out*.println(getName() + ": " + getDescription());  
}

* do same with NPC class

public class NPC implements CloneableGameEntity {  
 private String name;  
 private String description;  
 private int health;  
  
 public NPC(String name, String description, int health) {  
 this.name = name;  
 this.description = description;  
 this.health = health;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public String getDescription() {  
 return description;  
 }  
  
 public int getHealth() {  
 return health;  
 }  
  
 @Override  
 public NPC cloneEntity() {  
 return new NPC(this.name,this.description,this.health);  
 }  
  
 @Override  
 public void show() {  
 System.*out*.println(getName() + ": " + getDescription() + "\thealth: " + getHealth());  
 }  
}

* in Demo class create prototrypes

Room room = new Room("prototype","all rooms are same");  
NPC npc = new NPC("prototype","all npcs are same",100);

* clone them

Room room1 = room.cloneEntity();  
Room room2 = room.cloneEntity();

NPC npc1 = npc.cloneEntity();  
NPC npc2 = npc.cloneEntity();

* modify clones

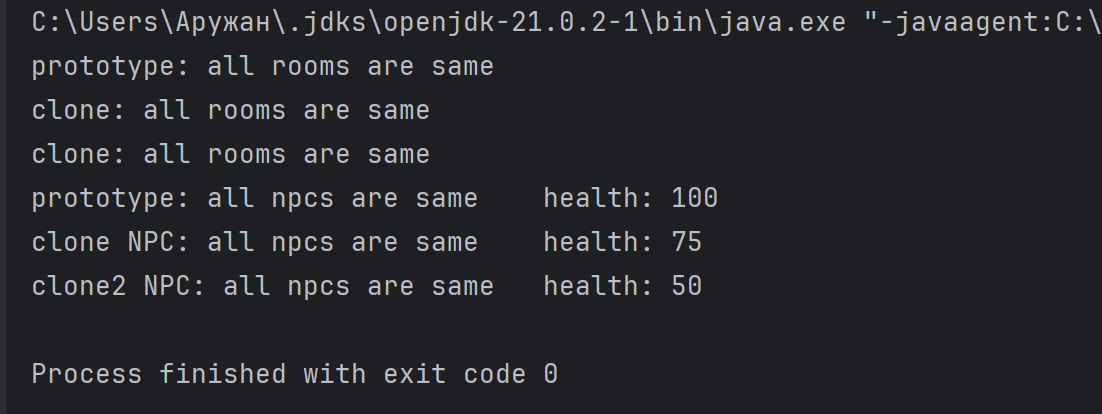
room1 = new Room("clone",room1.getDescription());  
room2 = new Room(room1.getName(),room2.getDescription());

npc1 = new NPC("clone NPC",npc1.getDescription(),75);  
npc2 = new NPC("clone2 NPC",npc2.getDescription(),50);

* just show our prototypes and clones we created

room.show();  
room1.show();  
room2.show();  
  
npc.show();  
npc1.show();  
npc2.show();

* run the code:



3. Assignment 3: Combine Builder and Prototype (Advanced)

We’ll use our previous classes: Dungeon, Room

* Add a Clone method to Room:

public Room cloneEntity() {  
 return new Room(this.name,this.description);  
}

* In Demo create a builder

IDungeonBuilder builder = new SimpleDungeonBuilder();

* Then prototype object:

Room room = new Room("prototype","blablabla");

* Configure the Builder and add Prototype room:

Dungeon dungeon = builder.setName("builder").addRoom(room).build();

* Clone a Room 3 times

Room room1 = room.cloneEntity(); //clones 'room'  
Room room2 = room.cloneEntity(); //clones 'room'  
Room room3 = room1.cloneEntity(); //clones 'room1'

* Modify them

room1 = new Room("room1","clones the prototype");  
room2 = new Room("room2",room2.getDescription()); //clones  
room3 = new Room("room3",room1.getDescription());

* Add them to a builder

dungeon.addRooms(room1);  
dungeon.addRooms(room2);  
dungeon.addRooms(room3);

* And show our Dungeon with similar rooms:

dungeon.show();

* Run the code

