SOLID

Open Close Principle

```java
public class Entity {
    protected boolean collisionAllowed;
    public Entity(boolean c) {
        collisionAllowed = c;
    }
}
```

```java
public class Door extends Entity {
    private String direction;
    public Door(String direction, boolean c) {
        super(c);
        this.direction = direction;
    }
    public String returnDirection() { return direction; }
}
```

This example came from our source code to introduce entities into our rooms. We will extend the Entity class to create classes that will have a collision relationship with the player. We add functionalities by writing it into the new subclass. As seen in the images, the Door constructor will be based on the Entity constructor, but it will also have a new attribute, direction, to allow the player to enter and exit the room. This is a good example of the Open Close Principle since the Entity class is being extended, but it is not being modified.

Single Responsibility Principle

```java
public class MapGenerator {
    private Room[][] roomMap;
    private Room start = new Room();
    private Room end = new Room();
    private Room currentRoom = new Room();
```

```java
public class MapController extends Application {
    private static Stage stage;
    private MapGenerator dungeonMap = new MapGenerator();
    private Room currentRoom = new Room();
    private static Player playerCharacter;

    MapController(Player playerPass) { playerCharacter = playerPass; }
```

```java
public class Room {
    private Room left;
    private Room right;
    private Room up;
    private Room down;
    private RoomTile[][] tiles;
    private boolean isRoomExit = false;
```

The map is based on different types of rooms. For the player to have the ability to move, search, and attack, many different aspects need to be created and implemented. Due to this fact, we created many classes that will be responsible for one single aspect. For example, MapGenerator was made for the sole purpose of generating a map with rooms, based on the Room class, while the MapController is used to set the player's movement and actions in the map. The Single Responsibility Principle supports this motion. We built our game with numerous small classes with specific names to be responsible for one single responsibility.