Code Review

Team 1 – T.A.P. (The Art of Programming)

Integrants:

Mateo Aymacaña

Josue Carbajal

Emily Calle

Samantha Bonilla

NRC: 28434

Lack of Clean Code

The main problem identified was the complete lack of application of clean code principles. The code was crammed together without a clear structure, with methods bunched together and lacking the logical separation necessary for smooth reading. There were no spaces between blocks of functionality, lines were excessively long, and there was no visual grouping to allow for quick identification of the different responsibilities within each class.

This lack of organization not only made the code difficult to read and understand but also significantly increased the likelihood of introducing errors during maintenance. The absence of separation between attributes, constructors, and different methods made navigating the code a tedious and confusing task.

```
public String getBlockName() {
    return blockName;}
public void setBlockName(String blockName) {
    this.blockName = blockName;}
public String getBlockCode() {
    return blockCode;}
public void setBlockCode(String blockCode) {
    this.blockCode = blockCode;}
public List<ParkingZone> getSections() {
    return sections;}
```

Excessive Coupling

The system exhibited excessive coupling between components, with circular dependencies that made it difficult to modify one part of the system without affecting the others. This coupling hampered unit testing and made the system fragile in the face of changes.

Responsibilities were mixed in such a way that a class intended for entity management also handled technical aspects such as JSON serialization and format validation.

Poor Error Handling

Error handling was basic and uninformative. Generic exceptions were used, which did not provide enough context to diagnose problems. Instead of using specific exception types that indicated the nature of the error, the Exception superclass was caught, thus losing valuable information about the specific type of failure.

Lack of consistency in conventions

The code exhibited inconsistencies in naming conventions, formatting, and structure. Some classes used one naming style while others used another, some methods followed different verbal patterns, and the internal organization of classes varied significantly across different parts of the system.

This lack of consistency made the code unpredictable and difficult to navigate, as developers could not rely on established patterns to quickly find the functionality they needed.

```
    String action = scanner.nextLine();
    EntryExitRecord record = new EntryExitRecord(name, action);
    // You can later save it with JsonDataManager or ParkingControlSystem
    System.out.println("Record saved successfully.");
}

case 2 -> {
    System.out.println("\n--- Track parking space status ---");
    // Example: ParkingSpace or ParkingLot
    ParkingSpace space = new ParkingSpace("A1", true);
    System.out.println("Space " + space.getId() + " is currently " +
            (space.isAvailable() ? "Available" : "Occupied"));
```