Refactoring Log

Code Restructuring Phase – SG Technologies POS System

Last Updated: 2025–11–28

This document tracks all 10 refactorings performed during the code restructuring phase. Each includes before/after code, rationale, and quality impact.

Refactoring #1: Extract Constants Class

Date: 2025–11–28 **Type:** Extract Constant **Files Changed:** 4

Before

```
public double tax=1.06;
private static float discount = 0.90f;
public static String couponNumber = "Database/couponNumber.txt";
if (length != 16) // credit card validation
```

After

```
public static final double DEFAULT_TAX_RATE = 1.06;
public static final float COUPON_DISCOUNT = 0.90f;
public static final String COUPON_FILE = DATABASE_DIR + "couponNumber.txt";
public static final int CREDIT_CARD_LENGTH = 16;

// Usage
public double tax=Constants.DEFAULT_TAX_RATE;
if (length != Constants.CREDIT_CARD_LENGTH)
```

Rationale

- Eliminates magic numbers/strings (code smell: Magic Number)
- Centralizes configuration values
- Single source of truth for future changes

Quality Impact

- **Maintainability:** ↑ High improvement
- **Readability:** ↑ Medium improvement
- **Testability:** → No change
- **Risk:** ✅ Low – compile-time safe

Refactoring #2: Extract System Utilities

Date: 2025–11–28 **Type:** Extract Method → Utility Class **Files Changed:** 4

Before

```
if (System.getProperty("os.name").startsWith("W") ||
    System.getProperty("os.name").startsWith("w")) { ... }
bw.write(System.getProperty("line.separator"));
```

After

```
public static boolean isWindows() {
    String osName = System.getProperty("os.name");
    return osName != null && (osName.startsWith("W") ||
        ↪ osName.startsWith("w"));
}
public static String getLineSeparator() {
    return System.getProperty("line.separator");
}
```

Quality Impact

- **Maintainability:** ↑ High improvement
- **DRY Principle:** ↑ High improvement
- **Testability:** ↑ Medium improvement
- **Risk:** ✅ Low – pure functions

Refactorings #3 to #9: Apply Constants & SystemUtils

Applied to: `PointOfSale.java`, `POSSystem.java`, `POS.java`, `POR.java`, `POH.java`, `Management.java`, `EmployeeManagement.java`

Summary

- All magic strings/numbers replaced
- All OS detection and line separator calls centralized
- 80 lines of duplication eliminated
- All changes: ✅ Very Low

Refactoring #10: Extract Duplicate deleteTempItem Method

Date: 2025–11–28 **Type:** Extract Method + Template Method **Files Changed:** 4

Before – Nearly identical 30–35 line methods in POS, POR, POH

After – Shared helper in PointOfSale.java

```
protected void deleteTempItemHelper(int id, boolean hasPhoneNumber) {
    try{
      String temp = Constants.NEW_TEMP_FILE;
      // ... consolidated file I/O logic ...
      if (hasPhoneNumber) {
        String phone = reader.readLine();
```

```
        writer.write(phone);
        writer.write(SystemUtils.getLineSeparator());
      }
      // ... write remaining items ...
    } catch(...) { ... }
}
```

After – Calling methods (3 lines each)

```
public void deleteTempItem(int id){
    deleteTempItemHelper(id, false);  // POS
    // or true for POR/POH
}
```

Rationale

- Eliminates 3 nearly identical implementations
- Single source of truth for temp file deletion
- Template Method handles phone number variation

Quality Impact

- **Maintainability:** ↑ High Very High
- **DRY:** ↑ High Very High (90 → 49 lines)
- **Lines Reduced:** 50 net
- **Risk:** ✅ Low

Final Summary of All Refactorings

#	Type	Description	Risk	Impact
1	Extract Constant	Created `Constants.java`	Low	High
2	Extract Utility	Created `SystemUtils.java`	Low	High
3–9	Apply Constants/Utils	7 classes updated	Very Low	Medium–High
10	Extract Method	`deleteTempItem()` → shared helper	Low	Very High

Overall Impact (Final)

- **Code Smells Eliminated:** Magic Numbers, Duplicate Code, Data Clumps (partial)
- **Lines Reduced:** **130 lines** of duplication removed
- **Files Improved:** 10 Java classes
- **Maintainability: Significantly improved**
- **Tests:** All characterization tests pass — no behavior change

Next Planned Refactorings

1. Extract file I/O into Repository pattern
2. Replace `float` with `BigDecimal` for currency
3. Extract date formatting into utility
4. Refactor long methods in `Management.java`
5. Add comprehensive unit test coverage

Document Version	1.2
Last Updated	2025–11–28
Status	10 Refactorings Complete – Phase 4 (60%) Done