AJ Wint
Task 3: Part D

1. Test Plan for Unit Testing

The objective of the unit tests was to ensure that individual components of the software, particularly the core entities (`Part`, `Product`, `OutsourcedPart`, and `InhousePart`), function correctly and meet the validation criteria. The focus was on testing critical fields such as name, price, inventory, date added, and other constraints that ensure business logic and data integrity are upheld.

Test Cases Covered:

- **Field Validations:** Ensured fields like `name`, `price`, `inventory`, and `dateAdded` adhere to the specified validation rules.
- **Boundary Tests:** Test fields such as `price` and `inventory` for both upper and lower boundary conditions.
- **Null Tests:** Validate that fields requiring non-null values throw appropriate errors when set to null.
- **Business Logic Validation:** Ensure that business constraints, like inventory staying within a specified range, are met.

(Screenshots Provided in ‘DOCUMENTATION’ Folder)

2. Unit Test Scripts(stored in the ‘domain’ directory in the ‘target’ package, as well as in the ‘DOCUMENTATION’ folder in “Test Plan Screenshots”)

The unit tests were implemented using JUnit 5 and validated using Java Bean Validation (JSR 380). Key validators for fields like `name` and `price` were annotated in the respective entity classes, ensuring that constraints such as non-null, minimum, and maximum values were enforced.

Examples of Unit Tests:

- **Product Tests:**
  - `testNullName()`: Verifies that a `null` name for the product triggers a validation error.
  - `testInventoryOutsideRange()`: Ensures inventory values that exceed the allowed maximum trigger a validation error.
  - `testNullDateAdded()`: Tests that the `dateAdded` field cannot be null.
- **Part Tests:**
  - `testNullName()`: Checks that setting the name to null causes a validation failure.

    - `testInvalidPrice()`: Ensures that negative values for price are disallowed.
- **OutsourcedPart and InhousePart Tests:**
    - `testNullCompanyName()` for `OutsourcedPart`: Ensures that a company name must be provided.
    - `testInvalidInventory()` for `InhousePart`: Confirms that negative inventory values are correctly flagged as invalid.

3. Results of Unit Tests (Screenshots in 'DOCUMENTATION' folder within "Unit Test Results Screenshots")

After running the tests, all validation-related errors were successfully captured and handled by the appropriate annotations in the entity classes. Below is a summary of the results:

- **Successful Tests:**
    - Field validations for `price`, `inventory`, `name`, and `dateAdded` worked as expected, with invalid inputs being rejected by the validators.
    - Tests such as `testInvalidPrice()` in the `Part` class and `testInventoryOutsideRange()` in the `Product` class confirmed that boundary conditions were appropriately managed.
- **Failed Tests & Fixes:**
    - **Initial Failures:**
        - Some initial test cases, such as `testNullName()` and `testInvalidPrice()` in `PartTest`, failed due to improper validation messages. These were corrected by aligning the test assertions with the validation messages specified in the entity class annotations.
        - The `testNullDateAdded()` in `ProductTest` initially failed due to the automatic population of the `dateAdded` field via the `@PrePersist` annotation. This was addressed by explicitly allowing the field to be validated without triggering the `@PrePersist` behavior in tests.
    - After resolving these issues, all tests passed, confirming the robustness of the implemented validation logic.

4. Summaries of Changes Resulting from Completed Tests

The unit tests brought about several code revisions to enhance validation coverage:

- **Validation Message Alignment:** The expected validation messages were corrected in the tests to ensure they matched the actual validation constraints in the entity classes.
- **Boundary Condition Handling:** Adjustments were made to ensure that fields like `price` and `inventory` correctly handle both minimum and maximum values, particularly in the `Product` and `Part` classes.

- **Date Handling:** For the `Product` entity, the `dateAdded` field’s handling was fine-tuned to ensure it is correctly set automatically and validated properly during testing.