**Part 1: Code Review and Bug Identification**

**a. Key Characteristics of Each Level of Testing**

### 1. Unit Testing

**Test Objectives:**

* Verify the functionality of individual components or methods within the code.
* Ensure that each unit functions correctly under specified conditions.
* Detect bugs in the smallest parts of code before they integrate into larger systems.

**Test Basis:**

* Source code of individual functions (e.g., ProcessClaim, calculatePayout).
* Specifications of the methods—input parameters and expected outputs.
* Use cases for various scenarios including edge cases.

### 2. Integration Testing

**Test Objectives:**

* Test the interaction between integrated units or modules.
* Identify issues that arise from combining different modules, such as communication failures.
* Ensure that integrated components work together as intended.

**Test Basis:**

* Interface designs and APIs between various modules.
* Combined functional and non-functional requirements.
* Interaction protocols—how two or more units interact (e.g., how ProcessClaim connects with other components).

### 3. System Testing

**Test Objectives:**

* Validate the complete and integrated software product.
* Ensure that the system meets the specified requirements.
* Test for overall functionality, performance, security, and compliance.

**Test Basis:**

* Complete system specifications, including functional and non-functional requirements.
* Use case scenarios covering typical user profiles.
* Regulatory and compliance standards that the system must adhere to.

### 4. Acceptance Testing

**Test Objectives:**

* Confirm that the software meets business requirements and is acceptable for delivery.
* Validate the system from an end-user perspective focusing on usability and functionality.
* Identify any discrepancies in requirements from the user's point of view.

**Test Basis:**

* Business requirements and end-user expectations.
* Acceptance criteria defined by stakeholders.
* Regulatory guidelines that must be fulfilled before the system goes live.

**b. Defects or Failures at Each Level and Examples**

|  |  |  |
| --- | --- | --- |
| **Level** | **Common Defects / Failures** | **Example** |
| Unit Testing | Logic Errors | If claimStatus == "Pending" should account for other states |
|  | Type Mismatches | Passing a string instead of a double to updateClaimAmount |
|  | Null Pointer Exceptions | Accessing properties of uninitialized objects, e.g., claimStatus |
| Integration Testing | Interface Mismatches | Mismatched method signatures between modules |
|  | Data Flow Issues | Updating amount in one place not reflecting in calculatePayout |
|  | State Management Problems | Claim status not updating properly in shared states |
| System Testing | Functionality Defects | calculatePayout returns wrong amount due to miscalculation logic |
|  | Performance Issues | Long processing time for multiple claims |
|  | Security Vulnerabilities | Allowing unauthorized users to change claim status directly |
| Acceptance Testing | Usability Issues | Clauses or status updates not clearly labeled in the UI |
|  | Compliance Failures | Insufficient checks for valid claim statuses in processing |
|  | User Acceptance Failures | Users unable to update claim amount, returning false without feedback |

# **2. Code Review – List of Errors and Fixes**

Question: Identify 6 errors in the code above. Explain each error and suggest fixes.

  
List of Errors and Fixes:

|  |  |  |
| --- | --- | --- |
| Error | Description | Fix |
| 1 | Variable names do not follow Java naming conventions. | Use `claimId` instead of `ClaimId`. Variable names should start with a lowercase letter (camelCase). |
| 2 | Class attributes are missing access modifiers (should be encapsulated). | Declare all class attributes as `private` for encapsulation (e.g., `private String claimId;`). |
| 3 | Constructor parameter not explicitly tied to class member. | Use `this.ClaimId = id;` or better, rename to `this.claimId = id;` for clarity. |
| 4 | String comparison uses `==` instead of `.equals()` in ProcessClaim. | Change `if (claimStatus == "Pending")` to `if (claimStatus.equals("Pending"))`. |
| 5 | Method and class members missing access modifiers. | Specify `public` or `private` for methods and fields for clear visibility. |
| 6 | Magic number used in calculation (`0.85`). | Replace `0.85` with a named constant, e.g., `private static final double PAYOUT\_PERCENTAGE = 0.85;`. |

Explanation:   
1. Java variables use camelCase.   
2. Use `private` for class members.   
3. Use `this.` for class member assignment in constructors.   
4. String comparison in Java must use `.equals()`.   
5. Specify access modifiers.   
6. Avoid magic numbers; use named constants.