Mutation Testing

Core Concepts of Mutation Testing with PITest

PITest Maven Dependency

```
<!-- PITest Maven Plugin -->
<build>
    <plugins>
        <plugin>
            <groupId>org.pitest</groupId>
            <artifactId>pitest-maven</artifactId>
            <version>1.15.3</version>
            <configuration>
                <!-- Configuration options here -->
            </configuration>
        </plugin>
    </plugins>
</build>
```

Running PITest with Maven

```
mvn org.pitest:pitest-maven:mutationCoverage
```

Conditional Boundary Mutation

```
public class Calculator {
    public boolean isEligibleForDiscount(int age) {
        return age > 18;
    }
}

// Test Class
public class CalculatorTest {
    @Test
    public void testIsEligibleForDiscount() {
        Calculator calculator = new Calculator();
        assertTrue(calculator.isEligibleForDiscount(19));
    }
}
```

Creating and Testing Mutants

```java
// Mutated Method in Calculator class
public boolean isEligibleForDiscount(int age) {
    return age >= 18; // Mutation: '>' changed to '>='
}
```

Test Results Interpretation

```
// Test Result Interpretation
Test Result: Passed
Mutation Details: Conditional Boundary (age > 18 to age >= 18)
Mutant Status: Survived
```

Advanced Features of PITest

1. Incremental Analysis
2. Excluded Classes and Methods
3. Custom Mutation Operators

Best Practices in Mutation Testing

1. Targeted Testing
2. Balancing Mutation Score and Practicality
3. Integrate PITest into your CI pipeline
4. Incremental Analysis
5. Complementary Testing

Common Pitfalls in Mutation Testing

1. Overemphasis on Mutation Score
2. Ignoring Equivalent Mutants
3. Neglecting Test Quality
4. Underestimating the Learning Curve
5. Misinterpreting Results