Assignment One

Test Plan

Odinakachukwu Nzekwe

Athabasca University

**Computer Science 308:**

Java for Programmers

COMP 308

**Test Plan for problem\_1 Package**

1. Introduction

This test plan aims to ensure the functionality and correctness of the problem\_1 package, specifically focusing on the Circle class. The Circle class represents a circle with its attributes and associated methods for calculations and operations.

2. Scope

The test plan covers the following aspects:

Correctness of circle attribute initialization.

Accuracy of circumference and area calculations.

Proper functioning of methods to set radius, print attributes, check if a point is inside the circle, and move the origin.

Validity of test cases provided in the main method.

3. Test Cases

3.1. Constructor

Test Case 1: Verify that the default constructor initializes the circle with the correct default values (coordinates at origin, radius 1.0).

Test Case 2: Test the parameterized constructor by initializing a circle with specific coordinates and radius, ensuring the values are correctly set.

3.2. Circumference and Area Calculation

Test Case 3: Validate the circumference() method by comparing the result with a manually calculated circumference.

Test Case 4: Confirm the area() method by comparing the result with a manually calculated area.

3.3. Set Radius Method

Test Case 5: Test the setRadius(double r) method with a radius within the maximum limit.

Test Case 6: Test the setRadius(double r) method with a radius exceeding the maximum limit, ensuring it sets the radius to the maximum value.

3.4. Print Attributes Method

Test Case 7: Verify that the printAttributes() method prints the correct values of coordinates, radius, circumference, and area to the console.

3.5. Is Inside Method

Test Case 8: Test the isInside(double xPoint, double yPoint) method with points inside and outside the circle's boundary, ensuring correct boolean results.

3.6. Move Method

Test Case 9: Validate the move(int xMove, int yMove) method by moving the origin to a new position and checking if the coordinates are updated accordingly.

3.7. Test Cases in Main Method

Test Case 10: Verify the output of the main method by comparing the printed attributes and results of isInside() after calling move().

4. Test Execution

Tests will be executed manually by invoking the methods and analyzing the output.

Each test case will be documented with its expected behavior.

Results will be compared against expected outcomes to determine success or failure.

5. Test Reporting

Test results, including passed and failed cases, will be documented.

Detailed information regarding any failed tests, including deviations from expected behavior, will be provided for debugging.

6. Conclusion

This test plan aims to ensure the correctness and functionality of the Circle class within the problem\_1 package. By following this plan, the class can be thoroughly tested to ensure its reliability and adherence to requirements.

**Test Plan for problem\_2**

1. Introduction

This test plan aims to ensure the functionality and correctness of the problem\_2 package, which includes classes for managing mailing addresses and shipping labels. The package consists of the FullName, MailingAddress, and ShippingLabel classes.

2. Scope

The test plan covers the following aspects:

Correct initialization of objects.

Proper formatting of names and addresses.

Validity of the toString() methods in the FullName and MailingAddress classes.

Accuracy of the label printing in the ShippingLabel class.

3. Test Cases

3.1. FullName Class

Test Case 1: Verify the construction of a FullName object with all parameters provided.

Test Case 2: Check the toString() method of the FullName class by creating an object and comparing the formatted name with the expected result.

3.2. MailingAddress Class

Test Case 3: Ensure the construction of a MailingAddress object with all parameters provided.

Test Case 4: Validate the toString() method of the MailingAddress class by creating an object and comparing the formatted address with the expected result.

3.3. ShippingLabel Class

Test Case 5: Verify the construction of a ShippingLabel object with valid ship-from and ship-to addresses.

Test Case 6: Test the printLabel() method of the ShippingLabel class and ensure the printed label matches the expected format.

3.4. Main Class

Test Case 7: Execute the main method and check for any runtime errors or exceptions.

4. Test Execution

Tests will be executed manually by invoking methods and analyzing outputs.

Each test case will be documented with its expected behavior.

Results will be compared against expected outcomes to determine success or failure.

5. Test Reporting

Test results, including passed and failed cases, will be documented.

Detailed information regarding any failed tests, including deviations from expected behavior, will be provided for debugging.

6. Conclusion

This test plan aims to ensure the correctness and functionality of the classes within the problem\_2 package. By following this plan, the package can be thoroughly tested to ensure its reliability and adherence to requirements.

**Test Plan for Problem\_3**

1. Introduction

This test plan aims to ensure the functionality and correctness of the Main class, which demonstrates polymorphism with cycles. The class includes instances of Unicycle, Bicycle, and Tricycle objects, showcasing the concept of upcasting and method overriding.

2. Scope

The test plan covers the following aspects:

Correct instantiation and upcasting of Unicycle, Bicycle, and Tricycle objects.

Proper invocation of the ride() method to display the appropriate message based on the number of wheels.

Verification of method overriding and polymorphic behavior.

3. Test Cases

3.1. Unicycle Class

Test Case 1: Verify that the Unicycle class correctly overrides the wheels() method to return 1.

Test Case 2: Ensure that invoking the ride() method on a Unicycle object prints the message "Riding a cycle with 1 wheel."

3.2. Bicycle Class

Test Case 3: Verify that the Bicycle class correctly overrides the wheels() method to return 2.

Test Case 4: Ensure that invoking the ride() method on a Bicycle object prints the message "Riding a cycle with 2 wheels."

3.3. Tricycle Class

Test Case 5: Verify that the Tricycle class correctly overrides the wheels() method to return 3.

Test Case 6: Ensure that invoking the ride() method on a Tricycle object prints the message "Riding a cycle with 3 wheels."

3.4. Main Class

Test Case 7: Execute the main method and confirm that each object's ride() method is correctly called based on their respective wheel counts.

4. Test Execution

Tests will be executed by running the Main class.

Each test case will be documented with its expected behavior.

Results will be compared against expected outcomes to determine success or failure.

5. Test Reporting

Test results, including passed and failed cases, will be documented.

Detailed information regarding any failed tests, including deviations from expected behavior, will be provided for debugging.

6. Conclusion

This test plan aims to ensure the correctness and functionality of the Main class, demonstrating polymorphism with cycles. By following this plan, the class can be thoroughly tested to ensure its reliability and adherence to requirements.

**Test Plan for problem\_4**

1. Introduction

This test plan aims to ensure the functionality and correctness of the problem\_4 package. The package includes classes and interfaces related to musical instruments and their playability.

2. Scope

The test plan covers the following aspects:

Correctness of instrument behavior.

Compliance with interface implementations.

Invocation and behavior of methods defined in classes.

Proper functionality of the Music5 class methods.

3. Test Cases

3.1. Instrument Class

Test Case 1: Ensure play() method of Instrument class prints "Instrument playing" to the console.

Test Case 2: Verify adjust() method of Instrument class prints "Adjusting Instrument" to the console.

3.2. Wind Class

Test Case 3: Confirm play() method of Wind class prints "Wind playing" to the console.

Test Case 4: Check clearSpitValve() method of Wind class prints "Clearing Spit Valve" to the console.

3.3. Percussion Class

Test Case 5: Ensure play() method of Percussion class prints "Percussion playing" to the console.

Test Case 6: Verify adjust() method of Percussion class prints "Adjusting Percussion" to the console.

3.4. Stringed Class

Test Case 7: Confirm play() method of Stringed class prints "Stringed playing" to the console.

Test Case 8: Check tune() method of Stringed class prints "Tuning Stringed" to the console.

3.5. Brass Class

Test Case 9: Ensure play() method of Brass class prints "Brass playing" to the console.

Test Case 10: Verify adjust() method of Brass class prints "Adjusting Brass" to the console.

3.6. Woodwind Class

Test Case 11: Confirm play() method of Woodwind class prints "Woodwind playing" to the console.

Test Case 12: Check adjust() method of Woodwind class prints "Adjusting Woodwind" to the console.

3.7. Music5 Class

Test Case 13: Ensure tune(Playable) method in Music5 class invokes the play() method of the passed Playable object.

Test Case 14: Verify tuneAll(Playable[]) method in Music5 class correctly invokes tune(Playable) for each element in the array.

4. Test Execution

Tests will be executed using JUnit or a similar testing framework.

Each test case will be documented with its expected behavior.

Tests will validate the expected output by comparing it with the actual output.

Tests will cover both positive and negative scenarios to ensure robustness.

5. Test Reporting

Test results, including passed and failed cases, will be documented.

Detailed information regarding any failed tests, including stack traces, will be provided for debugging.

Test coverage metrics may be generated to assess the completeness of testing.

6. Conclusion

This test plan aims to ensure the correctness and functionality of the problem\_4 package, covering various scenarios and edge cases. By following this plan, the package can be thoroughly tested to ensure its reliability and adherence to requirements.