System Requirements Specification Index

For

Shipping Charges-Junit

Version 1.0

TABLE OF CONTENTS

Shipping Charges

System Requirements Specification

1 PROJECT ABSTRACT

The **Java-Shipping Charges** project presents developers with a vital task: to design and implement a comprehensive set of test cases using JUnit to validate the functionality of the shipping charge calculation.

Your task is to develop a robust suite of test cases that thoroughly evaluate the shipping charge calculation system under various scenarios, ensuring accurate results and error-free performance.

The **Java-Shipping Charges** test suite aims to ensure the accuracy and reliability of the shipping charge calculation system, providing confidence in its performance and enhancing customer satisfaction.

2 CODE STRUCTURE

2.1 PACKAGE: COM.SHIPPING.SERVICE

Resources

Class/Interface	Description	Status
ShippingService(class)	• This class represents a service for calculating shipping costs based on the weight of the package and the distance it needs to be shipped. • It takes the weight and distance as input parameters and calculates the shipping cost according to predefined rates. • The billing logic is structured such that different rates are applied based on different weight ranges of the package.	Already implemented.

	• Don't modify any in this class as this is already implemented.	

2.2 PACKAGE: COM.SHIPPING.TEST

Resources

Class/Interface	Description	Status
ShippingTest(class)	• This class contains JUnit test cases to verify the correctness of the calculateShippingCost() method in the ShippingService class. • These test cases ensure that the shipping cost calculation implemented in the ShippingService class produces accurate results for different scenarios of package weight and shipping distance. • Make sure the test cases you write achieves 100% code coverage. • Make sure the test cases you write achieves 100% code coverage. • Make sure you write the test case for Invalid Data type for weight and distance • Make sure your test class has @Parameters method that should return test data. The test data must contain weight, distance (Values used to initialize ShippingService object) and expected value. Use the same in your test methods. • "Before" Lifecycle method must create and initialize the ShippingService object using weight and expected values. • "After" lifecycle method must make ShippingService class object as null to release the resources. Make sure you create "BeforeClass" and" AfterClass", lifecycle methods and print a log message in them.	To be implemented.

3 EXECUTION STEPS TO FOLLOW

1. **All actions like build, compile, running application, running test cases will be through Command Terminal.**
2. **To open the command terminal the test takers, need to go to Application menu (Three horizontal lines at left top) Terminal New Terminal.**
3. **This editor Auto Saves the code.**
4. **These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.**
5. **To execute and run test cases:**
   mvn clean install exec:java -Dexec.mainClass="mainapp.MyApp" -DskipTests=true