To conduct Boundary Value Analysis (BVA) in Eclipse Oxygen IDE using JUnit for the provided requirements, here's a step-by-step guide based on the document content:

Step-by-Step Demonstration

1. **Set up Eclipse Oxygen IDE**:
   - Open Eclipse Oxygen and create a new Java project (`File` > `New` > `Java Project`).
   - Name your project (e.g., "BankingAppBVA").
2. **Add JUnit Library**:
   - Right-click on your project > `Properties`.
   - Navigate to `Java Build Path` > `Libraries` tab > `Add Library...` > select `JUnit` (choose JUnit 5 for this example) > `Finish`.
3. **Create a New Java Class**:
   - Right-click on `src` > `New` > `Class`.
   - Name the class (e.g., `BankingApplicationTest`).
   - Ensure the `public static void main(String[] args)` box is unchecked.
4. **Implement the Validation Logic**:

```
import static org.junit.jupiter.api.Assertions.*;

import org.junit.jupiter.api.Test;


public class BankingApplicationTest {


   // Helper method to simulate validation (for demonstration)

   public boolean validateInput(String areaCode, String prefix, String suffix, String password, String command) {

      return (areaCode.matches("\\d{3}") || areaCode.isEmpty()) &&

            prefix.matches("\\d{3}") && Integer.parseInt(prefix) >= 200 && Integer.parseInt(prefix) <= 999 &&

            suffix.matches("\\d{4}") && Integer.parseInt(suffix) >= 1000 && Integer.parseInt(suffix) <= 9999 &&

            password.matches("\\w{6}") &&

            (command.equals("Check status") || command.equals("Deposit") || command.equals("Withdrawal"));

   }


   // Example Positive Test Cases

   @Test
```

```
    public void testValidMinAreaCode() {
        assertTrue(validateInput("000", "200", "1000", "abc123", "Check status"));
    }

    @Test
    public void testValidMaxAreaCode() {
        assertTrue(validateInput("999", "999", "9999", "xyz456", "Deposit"));
    }

    // Add remaining positive and negative test cases here following the format.
}
```