```java
package fr.eni.vault;

import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertThrows;
import org.junit.jupiter.api.Test;

import com.eni.vault.bll.BLLException;
import com.eni.vault.bll.Vault;

public class TestVaultClose {

    Vault vault = new Vault();

    @Test
    public void testFermetureCode() throws BLLException {
        //Arrange - préparerletest


        //Act -Vérifierlaméthode
        boolean canLock = vault.lock("1234");
        //Assert -Comparer le resultat renvoyé avec le résultat attendu
        assertEquals(canLock, true, "Echec fonction lock");


        //Act -Vérifierlaméthode
        boolean isLocked = vault.isLocked();
        //Assert isClosed
        assertEquals(isLocked, true, "Echec fonction est ferme");

        //Act / Assert
        assertThrows(BLLException.class,() -> vault.lock("1234"));
}

public class TestOpenVault {

    @Test
    public void testOpenVaultPasswordOk() throws BLLException {

        //Arrange
        vault.lock("1234");

        //Act open with code
        vault.unlock("1234");
        String password = vault.getPassword();
        //Assert
        assertEquals(password, "1234", "Echec fonction unLock");
    }

    @Test
    public void testOpenVaultIsOpened() throws BLLException {

        //Arrange
        vault.lock("1234");

        //Act open with code
        vault.unlock("1234");
        boolean isLocked = vault.isLocked();
        //Assert
        assertEquals(isLocked, false, "Echec fonction unLock");
    }

    @Test
    public void testWrongPassword() throws BLLException {
        //Arrange - préparerletest
        Vault vault = new Vault();

        //Act -lock the vault
        vault.lock("1234");

        //Act open with code with wrong password
        vault.unlock("1235");
        boolean isLocked = vault.isLocked();
        //Assert
        assertEquals(isLocked, true, "Echec fonction unLock - mauvais password");
    }

    @Test
    public void testDejaOuvert() throws BLLException {
        //Arrange - préparerletest
        Vault vault = new Vault();

        //Act / Assert
        assertThrows(BLLException.class,() -> vault.unlock("1235"));
    }

public class TestAjouterObjet {
    Vault vault = new Vault();

    @Test
    public void testAddObject() throws BLLException {
        //Arrange
        Storable gold = new Gold(20);

        //Act add Object in the vault
        int listObjDansCoffre = vault.addObject(gold);

        //Assert
        assertEquals(listObjDansCoffre, 1, "Echec fonction addObject");
    }

    @Test
    public void testItemIsInTheVault() throws BLLException {
        //Arrange
        Storable gold = new Gold(20);
        ArrayList<Storable> listObjects = vault.getObjectsInside();

        //Act add Object in the vault
        vault.addObject(gold);

        //Assert
        assertEquals(listObjects.get(0), gold,"Echec fonction addObject");
    }
}

public class TestRemoveObject {

    Vault vault = new Vault();

    @Test
    public void testRemoveObject() throws BLLException{
        //Arrange
        Storable gold = new Gold(20);
        vault.addObject(gold);

        //Act remove Object in the vault
        int listObjectSize = vault.removeObject(gold);

        //Assert
        assertEquals(listObjectSize, 0, "Echec fonction removeObject");

    }

    @Test
    public void testRemoveObjectVaultClosed()  throws BLLException {
        //Arrange
        Storable gold = new Gold(20);
        vault.addObject(gold);
        vault.lock("1234");

        //Assert and act
        assertThrows(BLLException.class,() -> vault.removeObject(gold));
    }

}
```