```java
import lt.vtvpmc.java.postoffice.AbstractPostOfficeTest;
import lt.vtvpmc.java.postoffice.PostOffice;

public class PostOfficeTestImpl extends AbstractPostOfficeTest {
    @Override
    protected PostOffice getPostOffice() {
        return new PostOfficeImpl();
    }
}

```

```java
import lt.vtvpmc.java.postoffice.IllegalPackageException;
import lt.vtvpmc.java.postoffice.Package;
import lt.vtvpmc.java.postoffice.PostOffice;

import java.util.ArrayList;
import java.util.List;

public class PostOfficeImpl implements PostOffice {
    private List<Package> packages = new ArrayList<>();//reikia sukurti sarasa kad butu deti paketus

    @Override //sio metodo adoc'e parasyta kad "Not having package code or owner is having those values set to null or an empty string", todel reikia patikrinti ar jie ne null bei ne empty
    public void postPackage(Package aPackage) throws IllegalPackageException {
    if (aPackage.getOwner() == null || aPackage.getPackageCode() == null || aPackage.getOwner() == "" || aPackage.getPackageCode() == "") {
        throw new IllegalPackageException();
    } else {
        packages.add(aPackage);
    }
    }

    @Override
    public long numberOfPackagesForOwner(String owner) {
        long count = 0; //pradinis kiekis yra nulis, long ne tai parasyta adoc (ir metode)
        for (Package aPackage : packages) { //iteruojam  per paketus ir jei paketo savininkas sutampa tai sumuojam
            if (aPackage.getOwner().equals(owner)) {
                count++;
            }
        }
        return count;
    }

//    @Override //paketus atiduodam pagal koda (zr. adoc)
```

```
32 //    public Package retrievePackage(String packageCode) {
33 //        for (int i = 0; i < packages.size(); i++) {
34 //            if (packages.get(i).getPackageCode().equals(packageCode)) {
35 //                Package pack = packages.get(i);
36 //                packages.remove(i);
37 //                return pack;
38 //            }
39 //        }
40 //        return null;
41 //    }
42     //alternatyva
43     @Override //paketus atiduodam pagal koda (zr. adoc)
44     public Package retrievePackage(String packageCode) {
45         for (Package aPackage : packages) {
46             if (aPackage.getPackageCode().equals(packageCode)) {
47                 packages.remove(aPackage);
48                 return aPackage;
49             }
50         }
51         return null;
52     }
53 }
54
```

```java
import lt.itakademija.exam.Bank;
import lt.itakademija.exam.CurrencyConverter;
import lt.itakademija.exam.CurrencyRatesProvider;
import lt.itakademija.exam.test.BaseTest;

public class MyTestSolution extends BaseTest {
    @Override
    protected Bank createBank(CurrencyConverter currencyConverter) { //eilute generuojasi pati po metodu implementavimo
        return new BankImpl(currencyConverter); //pakoreguota pagal poreiki
    }

    @Override
    protected CurrencyConverter createCurrencyConverter(CurrencyRatesProvider currencyRatesProvider) {
        return new CurrencyConverterImpl(currencyRatesProvider) {
        };
    }
}

```

```java
import lt.itakademija.exam.*;

public class CurrencyConverterImpl implements CurrencyConverter {

    private final CurrencyRatesProvider currencyRatesProvider; //kad keisti pradziai reikia zinot keitimo santyki

    public CurrencyConverterImpl(CurrencyRatesProvider currencyRatesProvider) {
        this.currencyRatesProvider = currencyRatesProvider;
    }

    @Override
    public Money convert(Currency from, Currency to, Money amount) { //eilute generuojasi pati

        Money rate = currencyRatesProvider.getRate(from, to); //kad keisti pradziai reikia zinot keitimo santyki

        if (rate == null) { //patikrinimas ar keitimo santykis turi duomenis (nera null)
            throw new CurrencyConversionException("Could  not convert currency ... ");
        }

        return amount.multiply(rate);
    }
}

```

```java
import lt.itakademija.exam.*;

import java.util.ArrayList;
import java.util.List;

public class BankImpl implements Bank {

    private final CurrencyConverter currencyConverter; //deklaruojam valiutos keitykla
    private final List<Customer> customers; //deklaruojam lista kad butu kur deti vartotojus, sukuriamus pagal metoda 'createCustomer'

    //trys generatoriai eiliskumui-unikalumui:
    private SequenceGenerator customerSequence = new SequenceGenerator(); //sukuriam generatoriu kad vartotojas turetu eiliskuma
    private SequenceGenerator accountSequence = new SequenceGenerator(); //sukuriam generatoriu saskaitai kita valiuta
    private SequenceGenerator operationSequence = new SequenceGenerator(); //sukuriam generatoriu operaciju eiliskumui


    public BankImpl(CurrencyConverter currencyConverter) { //'Generate --> Construktor'
    this.currencyConverter = currencyConverter;
    this.customers = new ArrayList<>(); //rankom (buvo this.customers = customers)
    }

    @Override // visi override generuojami, o paskui koreguojami rankomis
    public Customer createCustomer(PersonCode personCode, PersonName personName) {
        if (personCode == null || personName == null) { //cia tik patikrinimas ar vartotojas turi duomenis (koda ir varda)
            throw new NullPointerException();
        }

        for (Customer customer : customers) { //cia tik patikrinimas ar vartotojas nesikartoja (ar toks jau yra ar nera)
```

```
29              if (customer.getPersonCode().equals(personCode)) {  //tikrinam pagal koda (nes jis unikalus)
30                  throw new CustomerCreateException("Customer already exist");
31              }
32          }
33
34          Customer customer = new Customer (customerSequence.getNext(), personCode, personName); //sukuriam
    vartotoja + eiliskumas
35          customers.add(customer); //pridedam vartotoja i lista
36          return customer;
37      }
38
39      @Override
40      public Account createAccount(Customer customer, Currency currency) {
41          if (customer == null || currency == null) { //patikrinimas ar saskaita turi duomenis (vartotoja ir
    valiuta)
42              throw  new NullPointerException();
43          }
44
45          if (!customers.contains(customer) ) { //patikrinimas ar vartotojas jau turi saskaita kokia nors kita
     valiuta (pvz. EUR)
46              throw new AccountCreateException("Customer does not exits ... ");
47          }
48
49          Account account = new Account(accountSequence.getNext(), customer, currency, new Money(0.0d));
50          customer.addAccount(account); //pridedam saskaita vartotojui
51          return account;
52      }
53
54      @Override
55      public Operation transferMoney(Account debitAccount, Account creditAcount, Money debitAmount) {
56          if (debitAccount.getBalance().isLessThan(debitAmount)) { //palyginam ar saskaita nera mazesne uz
    paimama suma ("isLessThan' yra is jar failo)
57              throw new InsufficientFundsException("You are bad customer ... ");
58          }
```

```
59
60          debitAccount.setBalance(debitAccount.getBalance().substract(debitAmount)); //is saskaitos atimam
    paimamsuma (substract reiskia atimti)
61          //negalim tiesiog prideti debitAmount, nes gali buti operacija tarp skirtingu valiutu, todel
    pradzioj reikia konvertuoti:
62          Money creditAmount = currencyConverter.convert(debitAccount.getCurrency(), creditAcount.getCurrency(
    ), debitAmount );
63          creditAcount.setBalance(creditAcount.getBalance().add(creditAmount));
64          return new Operation(operationSequence.getNext(), debitAccount, creditAcount, debitAmount);
65      }
66
67      @Override
68      public Money getBalance(Currency currency) {
69          Money balance = new Money (0.0d); //sukuriam balansa su nuliu valiutos
70
71          //kadangi reikia balanso ir visu saskaitu ir tik eurais, tai iteruojam per visus vartotojus ir
    iteruojam per visas vartotojo saskaitas:
72          //jei visu saskaitu valiuta Eur, tai pridedam iskart, jei ne - tai pirma konvertuojam ir tada
    pridedam
73          for(Customer customer : customers) {
74              for (Account account : customer.getAccounts()) {
75                  if (account.getCurrency().equals(currency)) {
76                      balance = balance.add(account.getBalance());
77                  } else {
78                      balance = balance.add(currencyConverter.convert(account.getCurrency(), currency, account
    .getBalance()));
79                  }
80              }
81          }
82          return balance;
83      }
84  }
85
```

```java
package lt.vtvpmc.ems.aidas;
/*
    log4j --> log4j2, j.u.l., logback, slf4j
    log4j: ALL, TRACE, DEBUG, INFO, WARN, ERROR, FATAL, OFF; // cia yra lygiai

 */

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.io.File;
import java.io.IOException;

public class JavaLogging {

    //'prisukam' logeri:
    private final static Logger logger = LoggerFactory.getLogger(JavaLogging.class);

    public static void main(String[] args) throws IOException{

        logger.trace("Message from TRACE level");
        logger.info("Message from INFO level");
        logger.error("Message from ERROR level");

        createNewFile();
        createNewDirectory();
        createNewDirectories();
        createTempFile();
        renameFile();
        deleteFile();
    }

    private static void createNewFile() throws IOException {
        try {
```

```
35              File file = new File("test.txt"); //naujo objekto sukurimas
36                                                  // veliau ivedam nesama direktorija homeris
   ir meta 'No such file or directory' - isisaiskinti kodel ?
37              boolean bool = file.createNewFile();    //naujo failo sukurimas
38              System.out.println("File created: " + bool);
39          } catch (NullPointerException e) {
40              //System.out.println("null" + e); //parodo kuris exception meta klaida
41              e.printStackTrace(); //parodo klaidos atsiradimo kelia
42          } catch (IOException e) {
43              //System.out.println("io " + e); // parodo kuris exception meta klaida
44              e.printStackTrace(); // versija su starcTrace
45              logger.error("Sukuriant faila iskilo problemu" + e); //versija su logging
46          }
47      }
48
49      private static void createNewDirectory() throws IOException {
50          File file = new File("Java"); //naujo objekto sukurimas
51          boolean bool = file.mkdirs(); //naujos direktorijos sukurimas
52          System.out.println("Directory created: " + bool);
53      }
54
55      private static void createNewDirectories() throws IOException {
56          File file = new File("Java/Collections/List"); //naujo objekto sukurimas
57          boolean bool = file.mkdir(); //nauju direktoriju sukurimas
58          System.out.println("Directories created: " + bool);
59      }
60
61      private static void createTempFile() throws IOException {
62          File tempFile = File.createTempFile("tmp", ".tmp");
63          System.out.println("File path: " + tempFile.getAbsolutePath());
64          tempFile = File.createTempFile("tmp", null); //pakartota iliustravimui kad jei null vistiek sukuria
   tmp (zr. kode)
65          System.out.println("File path: " + tempFile.getAbsolutePath());
66      }
```

```
67
68      private static void renameFile() throws IOException {
69          File selectedFile = new File("test.txt");
70          selectedFile.renameTo(new File ("new_test.txt"));
71      }
72
73      private static void deleteFile() throws IOException {
74          File selectedFile = new File("new_test.txt");
75          selectedFile.delete();
76          System.out.println("File deleted: ");
77      }
78
79  }
80
```

```java
package lt.vtvpmc.ems.aidas;
/*
    log4j --> log4j2, j.u.l., logback, slf4j
    log4j: ALL, TRACE, DEBUG, INFO, WARN, ERROR, FATAL, OFF; // cia yra lygiai

 */

import org.apache.logging.log4j.LogManager; //naujas
import org.apache.logging.log4j.Logger;
//import org.slf4j.LoggerFactory; //senas

import java.io.File;
import java.io.IOException;

public class JavaLogging2 {

    //'prisukam' logeri:
    //private final static Logger logger = LoggerFactory.getLogger(JavaLogging.class); //senas
    private static final Logger logger = LogManager.getLogger(JavaLogging2.class.getName()); //naujas

    public static void main(String[] args) throws IOException{

        logger.trace("Message from TRACE level");
        logger.info("Message from INFO level");
        logger.error("Message from ERROR level");
        logger.info("Message from INFO level");

        createNewFile();
        createNewDirectory();
        createNewDirectories();
        createTempFile();
        renameFile();
        deleteFile();
    }
```

```
35
36      private static void createNewFile() throws IOException {
37          try {
38              File file = new File("/abc/test.txt"); //naujo objekto sukurimas
39                                                                  // veliau ivedam nesama direktorija homeris
    ir meta 'No such file or directory' - isisaiskinti kodel ?
40              boolean bool = file.createNewFile();    //naujo failo sukurimas
41              System.out.println("File created: " + bool);
42          } catch (NullPointerException e) {
43              //System.out.println("null" + e); //parodo kuris exception meta klaida
44              e.printStackTrace(); //parodo klaidos atsiradimo kelia
45          } catch (IOException e) {
46              //System.out.println("io " + e); // parodo kuris exception meta klaida
47              e.printStackTrace(); // versija su starcTrace
48              logger.error("Sukuriant faila iskilo problemu" + e); //versija su logging
49          }
50      }
51
52      private static void createNewDirectory() throws IOException {
53          File file = new File("Java"); //naujo objekto sukurimas
54          boolean bool = file.mkdirs(); //naujos direktorijos sukurimas
55          System.out.println("Directory created: " + bool);
56          logger.info("Direktorija sukurta" + bool);
57      }
58
59      private static void createNewDirectories() throws IOException {
60          File file = new File("Java/Collections/List"); //naujo objekto sukurimas
61          boolean bool = file.mkdir(); //nauju direktoriju sukurimas
62          System.out.println("Directories created: " + bool);
63      }
64
65      private static void createTempFile() throws IOException {
66          File tempFile = File.createTempFile("tmp", ".tmp");
67          System.out.println("File path: " + tempFile.getAbsolutePath());
```

```
68          tempFile = File.createTempFile("tmp", null); //pakartota iliustravimui kad jei null vistiek sukuria
    tmp (zr. kode)
69          System.out.println("File path: " + tempFile.getAbsolutePath());
70      }
71
72      private static void renameFile() throws IOException {
73          File selectedFile = new File("test.txt");
74          selectedFile.renameTo(new File ("new_test.txt"));
75      }
76
77      private static void deleteFile() throws IOException {
78          File selectedFile = new File("new_test.txt");
79          selectedFile.delete();
80          System.out.println("File deleted: ");
81          //logger.all();
82      }
83
84  }
85
```

```java
package acart;

import ashop.Item;
import ashop.Medicine;
import ashop.Product;
import ashop.Wine;
import org.junit.jupiter.api.Test;

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

class ShoppingCartTest {

    @Test
    public void medicineCanBeAddedToCart() {
        ShoppingCart shoppingCart = new ShoppingCart();

        shoppingCart.add(new Medicine("Test", 5.0));

        assertEquals(1, shoppingCart.getItemCount());
    }

    @Test
    public void wineCanBeAddedToCart() {
        ShoppingCart shoppingCart = new ShoppingCart();

        shoppingCart.add(new Wine("Test", 5.0, 1.0));

        assertEquals(1, shoppingCart.getItemCount());
    }

    @Test
    public void shouldReportFinalPrice() {
        ShoppingCart shoppingCart = new ShoppingCart();

```

```
35          shoppingCart.add(new Medicine("Ibuprofen", 1.0));
36          shoppingCart.add(new Medicine("Aspirin", 2.0));
37          shoppingCart.add(new Wine("Voruta", 5.0, 1.0));
38
39          assertEquals(11.2, shoppingCart.getTotalPrice(), 0.01);
40      }
41
42      @Test
43      public void shouldApplyDiscount() {
44          ShoppingCart shoppingCart = new ShoppingCart(
45                  new TestDiscountStrategy()
46          );
47
48          shoppingCart.add(new TestProduct("", 0.0));
49
50          assertEquals(0.5, shoppingCart.getTotalPrice());
51      }
52
53      class TestProduct extends Item {
54
55          public TestProduct(String name, double basePrice) {
56              super(name, basePrice);
57          }
58
59          public double getTotalPrice() {
60              return 1;
61          }
62      }
63
64      class TestDiscountStrategy implements DiscountStrategy {
65
66          public double applyDiscount(double totalPrice) {
67              return totalPrice * 0.5;
68          }
```

```java
package lt.vtvpmc.printer;

import lt.vtvpmc.printer.shapes.Circle;
import lt.vtvpmc.printer.shapes.Rectangle;
import org.junit.jupiter.api.Test;

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

public class InkCalculatorTest {

    @Test
    public void shouldCalculateInkUsageForSingleShape() {
        InkCalculator inkCalculator = new InkCalculator();

        inkCalculator.add(new TestShape());     //TestShape yra figura is pabaigoje esancio kodo

        assertEquals(0.1, inkCalculator.getRequiredInkInMl());
    }

    @Test
    public void shouldCalculateInkUsageForMultipleShapes() {
        InkCalculator inkCalculator = new InkCalculator();

        inkCalculator.add(new TestShape());
        inkCalculator.add(new TestShape());

        assertEquals(0.4, inkCalculator.getRequiredInkInMl());
    }

    //nebutinas, sukurtas tik tam kad neprisiristi + kad grazintu zinoma reiksme (siuo atveju 2)
    class TestShape implements Shape {
        public double getArea() {
            return 2;
        }
```

```java
package lt.vtvpmc.printer;

import lt.vtvpmc.printer.shapes.Rectangle;
import org.junit.jupiter.api.Test;

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

public class RectangleTest {

    @Test
    public void shouldCalculateAreaCorrectly() {
        Rectangle rectangle = new Rectangle(2.0, 3.0);

        assertEquals(6.0, rectangle.getArea());
    }

}

```