MODEL TEST 2

1. Fă o clasă de bază Animal cu o metodă sunet(), o clasă Caine care moștenește Animal, un thread care scrie sunetul câinelui într-un fișier text și o excepție custom aruncată dacă apare o eroare la scrierea în fișier.
2. Fă o clasă de bază Vehicul cu o metodă vitezaMaxima(), o clasă Masina care moștenește Vehicul, un thread care citește dintr-un fișier un singur cuvânt („MASINA” sau altceva) și afișează viteza, iar dacă tipul citit nu este „MASINA” să se arunce o excepție proprie.
3. Fă o clasă de bază Produs cu preț și o metodă pretFinal(), o clasă ProdusAlimentar care moștenește Produs și scade o reducere de 10%, un thread care scrie prețul final într-un fișier, iar dacă prețul inițial e negativ să arunce o excepție proprie.

```java
package test2;
import java.io.FileWriter;
import java.io.IOException;
class Animal {
    public String sunet() {
        return "sunet generic";}}
class Caine extends Animal {
    @Override
    public String sunet() {
        return "Ham ham";}}
class FileWriteException extends Exception {
    public FileWriteException(String msg) {
        super(msg);}}
class AnimalThread extends Thread {
    private Animal animal;
    private String fileName;

    public AnimalThread(Animal animal, String fileName) {
        this.animal = animal;
        this.fileName = fileName; }
    @Override
    public void run() {
        try (FileWriter fw = new FileWriter(fileName)) {
            fw.write("Sunet: " + animal.sunet());
        } catch (IOException e) {
            try {
                throw new FileWriteException("Nu pot scrie in fisier");
            } catch (FileWriteException ex) {
                System.out.println(ex.getMessage());} } }}
public class Problema1 {
    public static void main(String[] args) {
        Animal a = new Caine();
        AnimalThread t = new AnimalThread(a, "animal.txt");
        t.start(); }}
```

MODEL TEST 2

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
class Vehicul {
    public int vitezaMaxima() {
        return 0; }}
class Masina extends Vehicul {
    @Override
    public int vitezaMaxima() {
        return 180; }}
class TipVehiculInvalidException extends Exception {
    public TipVehiculInvalidException(String msg) {
        super(msg);}}
class VehiculThread extends Thread {
    private String fileName;
    public VehiculThread(String fileName) {
        this.fileName = fileName;}
    @Override
    public void run() {
        try (BufferedReader br = new BufferedReader(new FileReader(fileName))) {
            String tip = br.readLine();
            if ("MASINA".equalsIgnoreCase(tip)) {
                Vehicul v = new Masina();
                System.out.println("Viteza maxima: " + v.vitezaMaxima());
            } else {
                throw new TipVehiculInvalidException("Tip invalid: " + tip);}
        } catch (IOException e) {
            System.out.println("Eroare la citire fisier");
        } catch (TipVehiculInvalidException e) {
            System.out.println(e.getMessage()); } }}
class Problema2 {
    public static void main(String[] args) {
        VehiculThread t = new VehiculThread("vehicul.txt");
        t.start(); }}
```

MODEL TEST 2

```java
package test2;
import java.io.FileWriter;
import java.io.IOException;
class Produs {
    protected double pret;

    public Produs(double pret) {
        this.pret = pret;
    }

    public double pretFinal() {
        return pret;
    }
}

class ProdusAlimentar extends Produs {
    public ProdusAlimentar(double pret) {
        super(pret);
    }

    @Override
    public double pretFinal() {
        return pret * 0.9; // reducere 10%
    }
}

class PretInvalidException extends Exception {
    public PretInvalidException(String msg) {
        super(msg);
    }
}
```

```java
class ProdusThread extends Thread {
    private Produs produs;
    private String fileName;
    public ProdusThread(Produs produs, String fileName) {
        this.produs = produs;
        this.fileName = fileName;
    }

    @Override
    public void run() {
        try {
            if (produs.pret < 0) {
                throw new PretInvalidException("Pret negativ!");
            }
            try (FileWriter fw = new FileWriter(fileName)) {
                fw.write("Pret final: " + produs.pretFinal());
            }
        } catch (PretInvalidException e) {
            System.out.println(e.getMessage());
        } catch (IOException e) {
            System.out.println("Eroare la scriere fisier");
        }
    }
}
```

```
59 class Problema3 {
60     public static void main(String[] args) {
61         Produs p = new ProdusAlimentar(100); // încearcă si cu -50 să vezi exceptia
62         ProdusThread t = new ProdusThread(p, "produs.txt");
63         t.start();
64     }
65 }
66
```