1)Klasa Item

```java
package pl.edu.agh.dronka.shop.model;

import pl.edu.agh.dronka.shop.model.items.MainItem;

public class Item {

   private String name;

   private MainItem mainItem;

   private int price;

   private int quantity;

   private boolean secondhand;

   private boolean polish;

   public Item(String name, MainItem mainItem, int price, int quantity) {
      this.name = name;
      this.mainItem = mainItem;
      this.price = price;
      this.quantity = quantity;
   }

   public Item() {
   }
```

1) Klasa Food

```java
package pl.edu.agh.dronka.shop.model.ItemTypes;

import pl.edu.agh.dronka.shop.model.Category;

import java.util.Date;

public class Food extends Item {

    private Date date;

    public Food(String name, Category category, int price, int quantity,  Date date) {
        super(name, category, price, quantity);
        this.date=date;
    }

    public Food(){}

    public Date date() {
```

```
        return date;
    }

    public void setDate(Date date) {
        this.date = date;
    }
}
package pl.edu.agh.dronka.shop.model.ItemTypes;

import pl.edu.agh.dronka.shop.model.Category;
import pl.edu.agh.dronka.shop.model.ItemTypes.utils.Genre;
```

2) Klasa Books

```
package pl.edu.agh.dronka.shop.model.ItemTypes;

import pl.edu.agh.dronka.shop.model.Category;

public class Books extends Item {
    private int pages;
    private boolean isHardcover;

    public Books(String name, Category category, int price, int quantity, int pages, boolean
isHardcover){
        super(name, category, price, quantity);
        this.pages=pages;
        this.isHardcover=isHardcover;
    }
    public Books(){}

    public int getPages() {
        return pages;
    }

    public void setPages(int pages) {
        this.pages = pages;
    }

    public boolean isHardcover() {
        return isHardcover;
    }

    public void setHardcover(boolean hardcover) {
        isHardcover = hardcover;
    }
}
```

3) Klasa Music

```java
public class Music extends Item {

    private Genre genre;
    private boolean hasVideo;

    public Music(String name, Category category, int price, int quantity, String genre,
boolean hasVideo) {
        super(name, category, price, quantity);
        this.genre = Genre.parser(genre);
        this.hasVideo = hasVideo;
    }

    public Music(){}

    public Genre getGenre() {
        return genre;
    }

    public void setGenre(Genre genre) {
        this.genre = genre;
    }

    public boolean isHasVideo() {
        return hasVideo;
    }

    public void setHasVideo(boolean hasVideo) {
        this.hasVideo = hasVideo;
    }
}
```

```java
package pl.edu.agh.dronka.shop.model.ItemTypes.utils;

public enum Genre {
    ROCK("ROCK"), JAZZ("JAZZ"), DUBSTEP("DUBSTEP"), TECHNO("TECHNO"), BLUES("BLUES"), COUNTRY("COUNTRY"),
POP("POP");

    private String displayName;

    Genre(String genreName) {
        this.displayName = genreName;
    }

    public static Genre parser(String genreName){
        switch(genreName.toUpperCase()){
            case "ROCK":
                return Genre.ROCK;
            case "JAZZ":
                return Genre.JAZZ;
           case "TECHNO":
                return Genre.TECHNO;
            case "BLUES":
                return Genre.BLUES;
            case "COUNTRY":
                return Genre.COUNTRY;
            case "POP":
                return Genre.POP;
            default:
                throw new IllegalArgumentException("No such genre: "+  genreName.toUpperCase());
```

```
        }
    }

    public String getDisplayName() {
        return displayName;
    }
}
```

4)Klasa Electronics

```
private static List<Item> readItems(CSVReader reader, Category category) {
        List<Item> items = new ArrayList<>();

        try {
            reader.parse();
            List<String[]> data = reader.getData();

            for (String[] dataLine : data) {
package pl.edu.agh.dronka.shop.model.ItemTypes;

import pl.edu.agh.dronka.shop.model.Category;

public class Electronics extends Item {
    private boolean isMobile;
    private boolean hasGuarantee;

    public Electronics(String name, Category category, int price, int quantity, boolean isMobile, boolean
hasGuarantee) {
        super(name, category, price, quantity);
        this.isMobile = isMobile;
        this.hasGuarantee = hasGuarantee;
    }

    public Electronics(){}

   public boolean isHasGuarantee() {
        return hasGuarantee;
    }

    public void setHasGuarantee(boolean hasGuarantee) {
        this.hasGuarantee = hasGuarantee;
    }


    public boolean isMobile() {
        return isMobile;
    }

    public void setMobile(boolean mobile) {
        isMobile = mobile;
    }


}
```

5) Klasa Sport

```java
package pl.edu.agh.dronka.shop.model.ItemTypes;

import pl.edu.agh.dronka.shop.model.Category;

public class Sport extends Item {

    public Sport(String name, Category category, int price, int quantity) {
        super(name, category, price, quantity);
    }
    public Sport(){}
}
```

6) Klasa PropertiesHelper

```java
                String name = reader.getValue(dataLine, "Nazwa");
                int price = Integer.parseInt(reader.getValue(dataLine, "Cena"));
                int quantity = Integer.parseInt(reader.getValue(dataLine,
                        "Ilość"));

                boolean isPolish = Boolean.parseBoolean(reader.getValue(
                        dataLine, "Tanie bo polskie"));
                boolean isSecondhand = Boolean.parseBoolean(reader.getValue(
                        dataLine, "Używany"));

                Item item;
                switch (category) {
                    case BOOKS:
                        item = new Books(name, category, price, quantity, (int)
                                Integer.parseInt(reader.getValue(dataLine, "Liczba stron")),

package pl.edu.agh.dronka.shop.model.util;

import java.util.LinkedHashMap;
import java.util.Map;

import pl.edu.agh.dronka.shop.model.ItemTypes.*;

public class PropertiesHelper {

          public static Map<String, Object> getPropertiesMap(Item item) {
                    Map<String, Object> propertiesMap = new LinkedHashMap<>();

                    propertiesMap.put("Nazwa", item.getName());
                    propertiesMap.put("Cena", item.getPrice());
                    propertiesMap.put("Kategoria", item.getCategory().getDisplayName());
                    propertiesMap.put("Ilość", Integer.toString(item.getQuantity()));
                    propertiesMap.put("Tanie bo polskie", item.isPolish());
                    propertiesMap.put("Używany", item.isSecondhand());

                    if (item instanceof Books) {
                              propertiesMap.put("Liczba stron", ((Books)item).getPages());
                              propertiesMap.put("Twarda oprawa", ((Books)item).isHardcover());
                    } else if (item instanceof Electronics){
```

```
                        propertiesMap.put("Mobilny", ((Electronics) item).isMobile());
                        propertiesMap.put("Gwarancja",
((Electronics)item).isHasGuarantee());
                } else if(item instanceof Food){
                        propertiesMap.put("Data przydatności", ((Food)
item).getExpirationDate());
                } else if (item instanceof Music){
                        propertiesMap.put("Gatunek muzyczny", ((Music) item).getGenre());
                        propertiesMap.put("Załączone wideo", ((Music) item).isHasVideo());
                }

                return propertiesMap;
        }
}




            Boolean.parseBoolean(reader.getValue(dataLine, "Twarda oprawa")));
                    break;
                case ELECTRONICS:
                    item = new Electronics(name, category, price, quantity,
                            Boolean.parseBoolean(reader.getValue(dataLine, "Mobilny")),
                            Boolean.parseBoolean(reader.getValue(dataLine, "Gwarancja")));
                    break;
                case FOOD:
                    String dateString = reader.getValue(dataLine, "Data przydatności");
                    SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");
                    Date date = formatter.parse(dateString);
                    item = new Food(name, category, price, quantity, date);
                    break;
                case MUSIC:
                                                    item = new Music(name, category, price,
quantity,

        reader.getValue(dataLine, "Gatunek muzyczny"),

        Boolean.parseBoolean(reader.getValue(dataLine, "Załączone wideo")));
                    break;
                                            case SPORT:
                                                    item = new Sport(name, category, price,
quantity);
                    break;

                                            default:
                                                    throw new
IllegalArgumentException("Error parsing data from csv");
                }


                item.setPolish(isPolish);
                item.setSecondhand(isSecondhand);

                items.add(item);

            }

        } catch (IOException | ParseException e) {
            e.printStackTrace();
        }
```

```
        return items;
    }
```

7) Klasa PropertiesPanel

```
package pl.edu.agh.dronka.shop.view;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.text.SimpleDateFormat;
import java.util.Date;

import javax.swing.BoxLayout;
import javax.swing.JCheckBox;
import javax.swing.JPanel;

import pl.edu.agh.dronka.shop.controller.ShopController;


import pl.edu.agh.dronka.shop.model.ItemTypes.*;
import pl.edu.agh.dronka.shop.model.filter.ItemFilter;

public class PropertiesPanel extends JPanel {

    private static final long serialVersionUID = -2804446079853846996L;
    private ShopController shopController;

    private ItemFilter filter ;

    public PropertiesPanel(ShopController shopController) {
        this.shopController = shopController;
        setLayout(new BoxLayout(this, BoxLayout.PAGE_AXIS));
    }

    public void fillProperties() {
        removeAll();
        this.filter = new ItemFilter( shopController.getCurrentCategory());

        filter.getItemSpec().setCategory(shopController.getCurrentCategory());
        add(createPropertyCheckbox("Tanie bo polskie", new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent event) {
                filter.getItemSpec().setPolish(
                        ((JCheckBox) event.getSource()).isSelected());
                shopController.filterItems(filter);
            }
        }));

        add(createPropertyCheckbox("Używany", new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent event) {
                filter.getItemSpec().setSecondhand(
                        ((JCheckBox) event.getSource()).isSelected());
```

```
                shopController.filterItems(filter);
            }
        }));

        switch (filter.getItemSpec().getCategory()) {
            case BOOKS:
                add(createPropertyCheckbox("Twarda oprawa", new ActionListener() {

                    @Override
                    public void actionPerformed(ActionEvent event) {
                        if (!(filter.getItemSpec() instanceof Books)) return;
                        ((Books) filter.getItemSpec()).setHard(
                                ((JCheckBox) event.getSource()).isSelected());
                        shopController.filterItems(filter);
                    }
                }));
                break;
            case ELECTRONICS:
                add(createPropertyCheckbox("Mobilny", new ActionListener() {

                    @Override
                    public void actionPerformed(ActionEvent event) {
                        if (!(filter.getItemSpec() instanceof Electronics)) return;
                        ((Electronics) filter.getItemSpec()).setMobile(
                                ((JCheckBox) event.getSource()).isSelected());
                        shopController.filterItems(filter);
                    }
                }));

                add(createPropertyCheckbox("Gwarancja", new ActionListener() {

                    @Override
                    public void actionPerformed(ActionEvent event) {
                        if (!(filter.getItemSpec() instanceof Electronics)) return;
                        ((Electronics) filter.getItemSpec()).setHasGuarantee(
                                ((JCheckBox) event.getSource()).isSelected());
                        shopController.filterItems(filter);
                    }
                }));
                break;
            case MUSIC:
                add(createPropertyCheckbox("Czy ma wideo", new ActionListener() {

                    @Override
                    public void actionPerformed(ActionEvent event) {
                        if (!(filter.getItemSpec() instanceof Music)) return;
                        ((Music) filter.getItemSpec()).setHasVideo(
                                ((JCheckBox) event.getSource()).isSelected());
                        shopController.filterItems(filter);
                    }
                }));
                break;
        }

    }

    private JCheckBox createPropertyCheckbox(String propertyName,
                                             ActionListener actionListener) {

        JCheckBox checkBox = new JCheckBox(propertyName);
        checkBox.setSelected(false);
```

```
        checkBox.addActionListener(actionListener);

        return checkBox;
    }

}
```

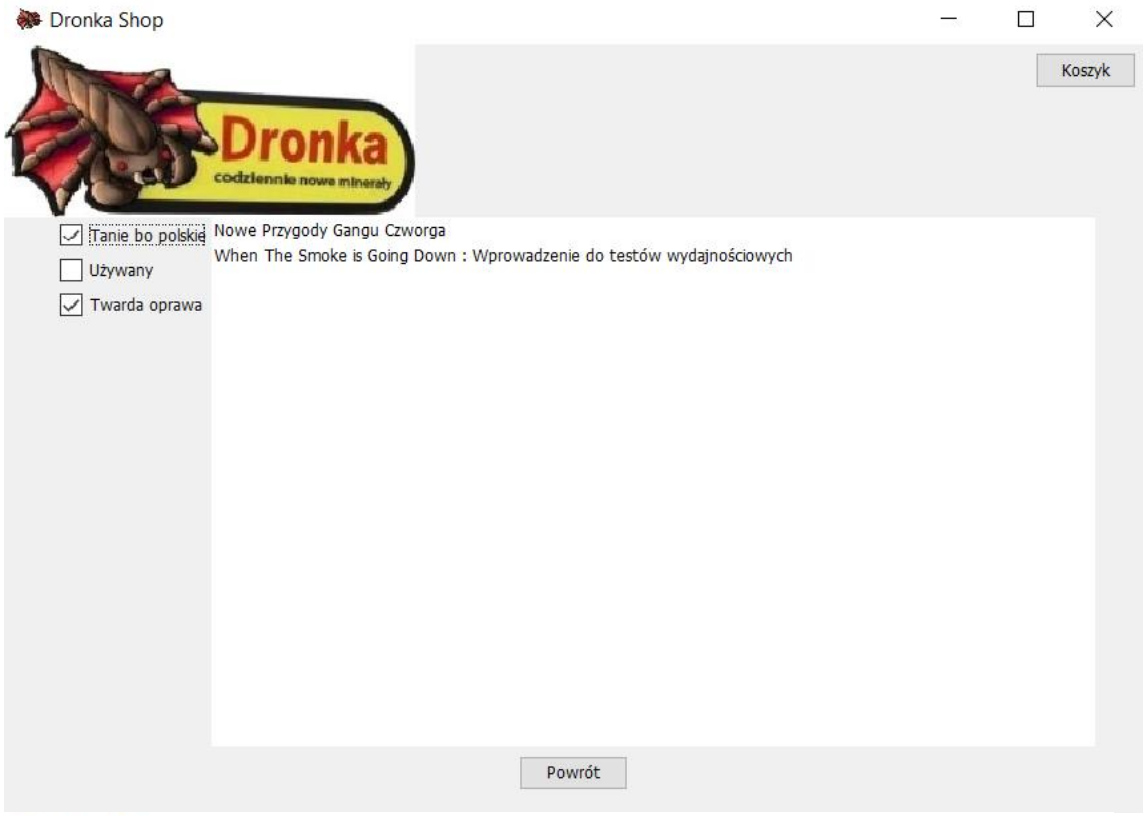
8) Klasa ShopProvider

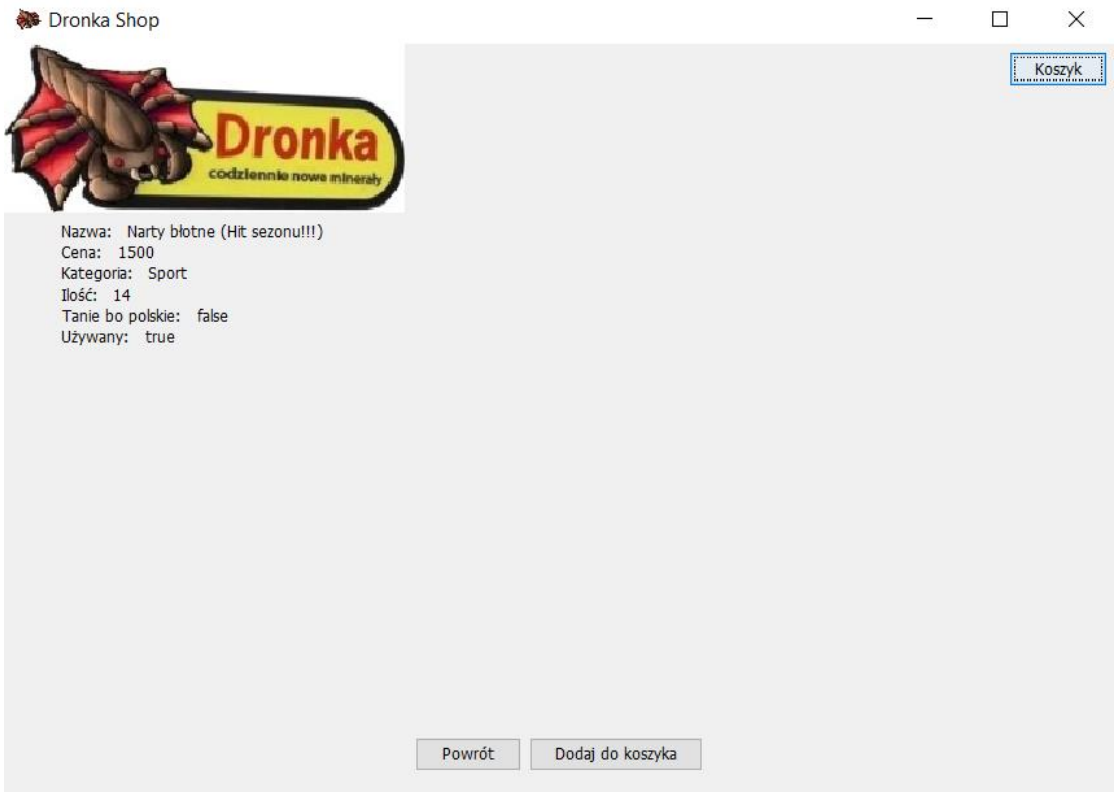
```
private static List<Item> readItems(CSVReader reader, Category category) {
        List<Item> items = new ArrayList<>();

        try {
            reader.parse();
            List<String[]> data = reader.getData();

            for (String[] dataLine : data) {

                String name = reader.getValue(dataLine, "Nazwa");
                int price = Integer.parseInt(reader.getValue(dataLine, "Cena"));
                int quantity = Integer.parseInt(reader.getValue(dataLine,
                        "Ilość"));

                boolean isPolish = Boolean.parseBoolean(reader.getValue(
                        dataLine, "Tanie bo polskie"));
                boolean isSecondhand = Boolean.parseBoolean(reader.getValue(
                        dataLine, "Używany"));

                Item item;
                switch (category) {
                    case BOOKS:
                        item = new Books(name, category, price, quantity, (int)
                                Integer.parseInt(reader.getValue(dataLine, "Liczba stron")),
                                Boolean.parseBoolean(reader.getValue(dataLine, "Twarda
oprawa")));
                        break;
                    case ELECTRONICS:
                        item = new Electronics(name, category, price, quantity,
                                Boolean.parseBoolean(reader.getValue(dataLine, "Mobilny")),
                                Boolean.parseBoolean(reader.getValue(dataLine, "Gwarancja")));
                        break;
                    case FOOD:
                        String dateString = reader.getValue(dataLine, "Data przydatności");
                        SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");
                        Date date = formatter.parse(dateString);
                        item = new Food(name, category, price, quantity, date);
                        break;
                                        case MUSIC:
                                                item = new Music(name, category, price,
quantity,

         reader.getValue(dataLine, "Gatunek muzyczny"),

         Boolean.parseBoolean(reader.getValue(dataLine, "Czy ma wideo")));
                        break;
                                        case SPORT:
```

```
                                                    item = new Sport(name, category, price,
quantity);
                        break;

                                                default:
                                                    throw new
IllegalArgumentException("Error parsing data from csv");
                }


                item.setPolish(isPolish);
                item.setSecondhand(isSecondhand);

                items.add(item);

            }

        } catch (IOException | ParseException e) {
            e.printStackTrace();
        }

        return items;
    }
```


Dronka Shop
Koszyk
Dronka
codziennie nowe minerały
Tanie bo polskie
Używany
Czy ma wideo
Ciley Myrus - Big Ball of Mud
Powrót

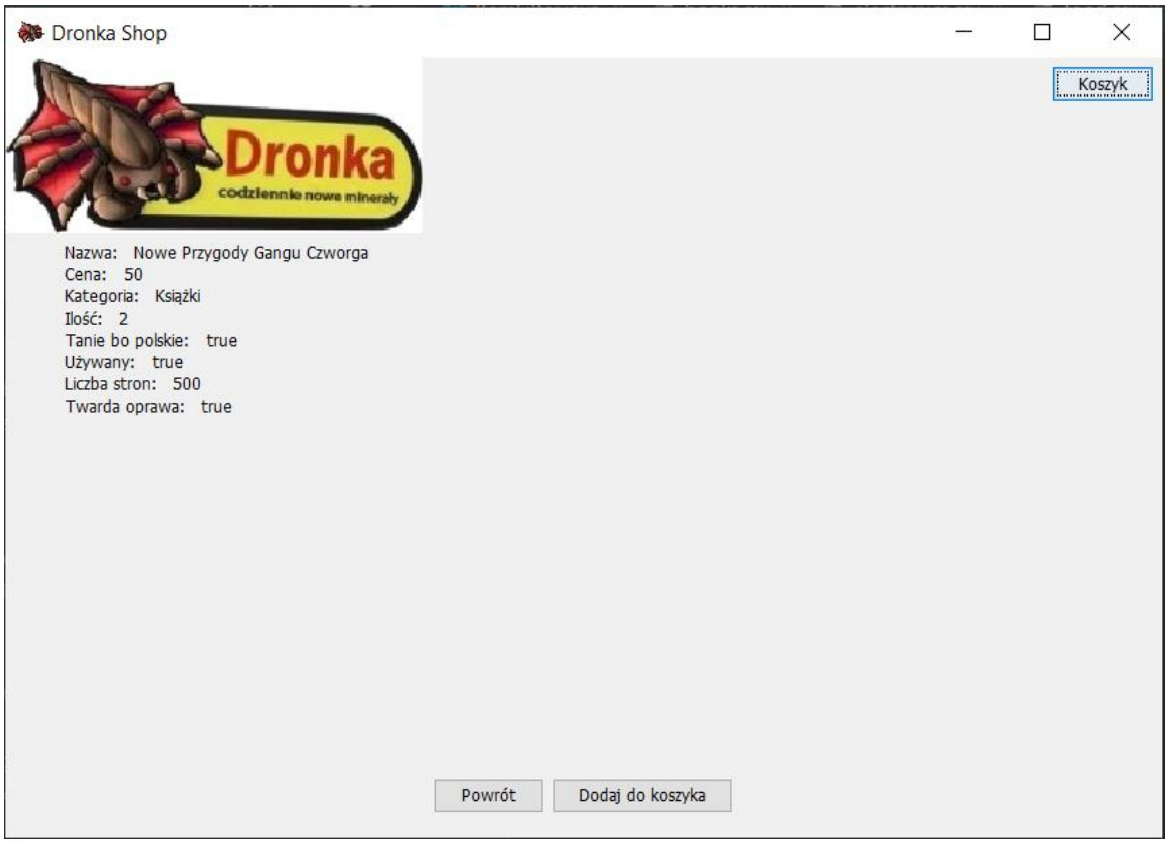

Dronka Shop
Koszyk
Dronka
codziennie nowe minerały
Tanie bo polskie
Używany
Mobilny
Gwarancja
Legendarny Bulbulator
Powrót

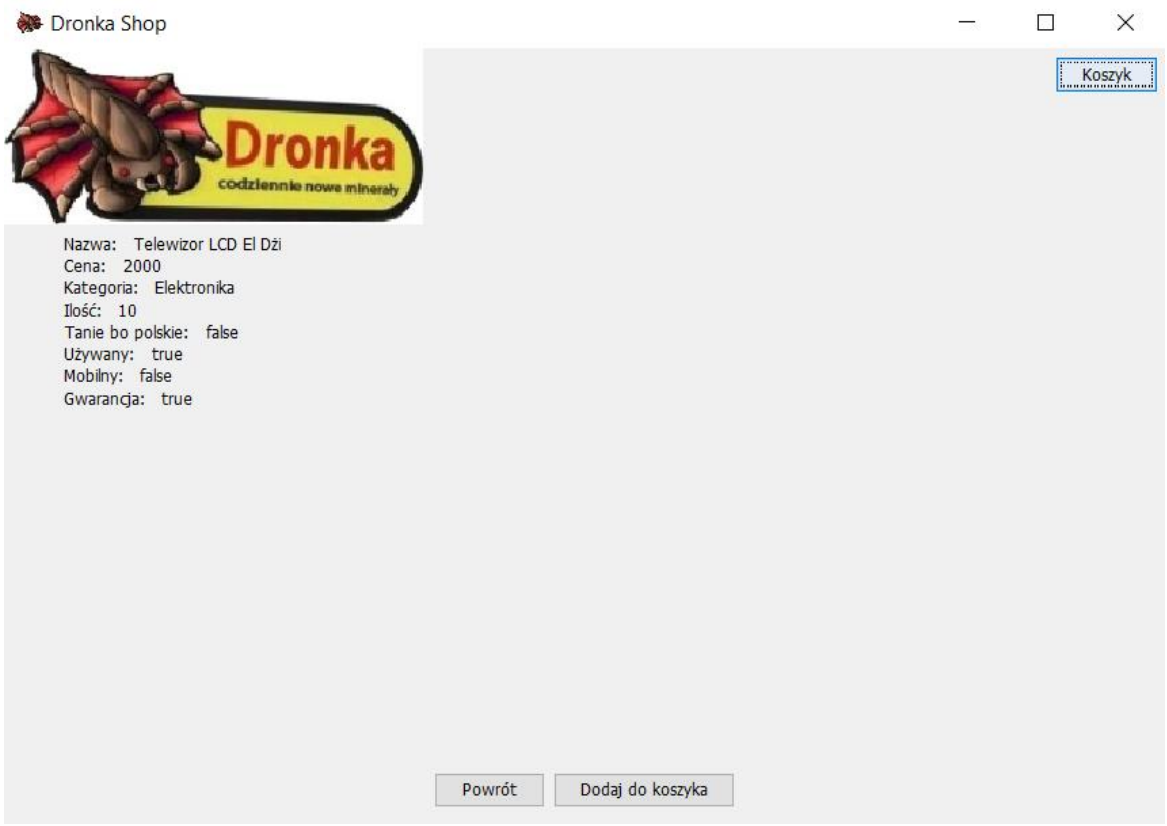

Dronka Shop
Koszyk
Dronka
codziennie nowe minerały
Tanie bo polskie
Używany
Twarda oprawa
Nowe Przygody Gangu Czworga
When The Smoke is Going Down : Wprowadzenie do testów wydajnościowych
Powrót


Dronka Shop
Koszyk
Dronka
codziennie nowe minerały
Nazwa: Narty błotne (Hit sezonu!!!)
Cena: 1500
Kategoria: Sport
Ilość: 14
Tanie bo polskie: false
Używany: true
Powrót
Dodaj do koszyka


Dronka Shop
Koszyk
Dronka
codziennie nowe minerały
Nazwa: Zupa Studencka Instant
Cena: 2
Kategoria: Żywność
Ilość: 100
Tanie bo polskie: true
Używany: false
Data przydatności: Sun May 10 00:00:00 CEST 2020
Powrót
Dodaj do koszyka


Dronka Shop
Koszyk
Dronka
codziennie nowe minerały
Nazwa: Ciley Myrus - Big Ball of Mud
Cena: 60
Kategoria: Muzyka
Ilość: 20
Tanie bo polskie: true
Używany: false
Gatunek muzyczny: POP
Czy ma wideo: true
Powrót
Dodaj do koszyka


Dronka Shop
Koszyk
Dronka
codziennie nowe minerały
Nazwa: Nowe Przygody Gangu Czworga
Cena: 50
Kategoria: Książki
Ilość: 2
Tanie bo polskie: true
Używany: true
Liczba stron: 500
Twarda oprawa: true
Powrót
Dodaj do koszyka


Dronka Shop
Koszyk
Dronka
codziennie nowe minerały
Nazwa: Telewizor LCD El Dżi
Cena: 2000
Kategoria: Elektronika
Ilość: 10
Tanie bo polskie: false
Używany: true
Mobilny: false
Gwarancja: true
Powrót
Dodaj do koszyka