Laboratorium 2 Wprowadzanie zmian w istniejącej aplikacji

Jakub Myśliwiec

Zadanie 2.1

Żeby rozszerzyć produkty o dodatkowe właściwości postanowiłem stworzyć klasę, dla każdej z kategorii, która miała posiadać dodatkowe właściwości. Klasy te dziedziczą po klasie Item, i posiadają atrybuty odpowiadające dodatkowym właściwościom.

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

public class Book extends Item {
    private int numberOfPages;

    private boolean hardcover;

    public Book(String name, Category category, int price, int quantity, int numberOfPages, boolean hardcover){
        super(name, category, price, quantity);
        this.numberOfPages = numberOfPages;
        this.hardcover = hardcover;
    }

    public int getNumberOfPages() { return numberOfPages; }

    public void setNumberOfPages(int numberOfPages) { this.numberOfPages = numberOfPages; }

    public boolean isHardcover() { return hardcover; }

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

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

public class Electronics extends Item {
    private boolean mobile;

    private boolean guarantee;

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

    public boolean isMobile() { return mobile; }

    public void setMobile(boolean mobile) { this.mobile = mobile; }

    public boolean isGuarantee() { return guarantee; }

    public void setGuarantee(boolean guarantee) { this.guarantee = guarantee; }
}
```

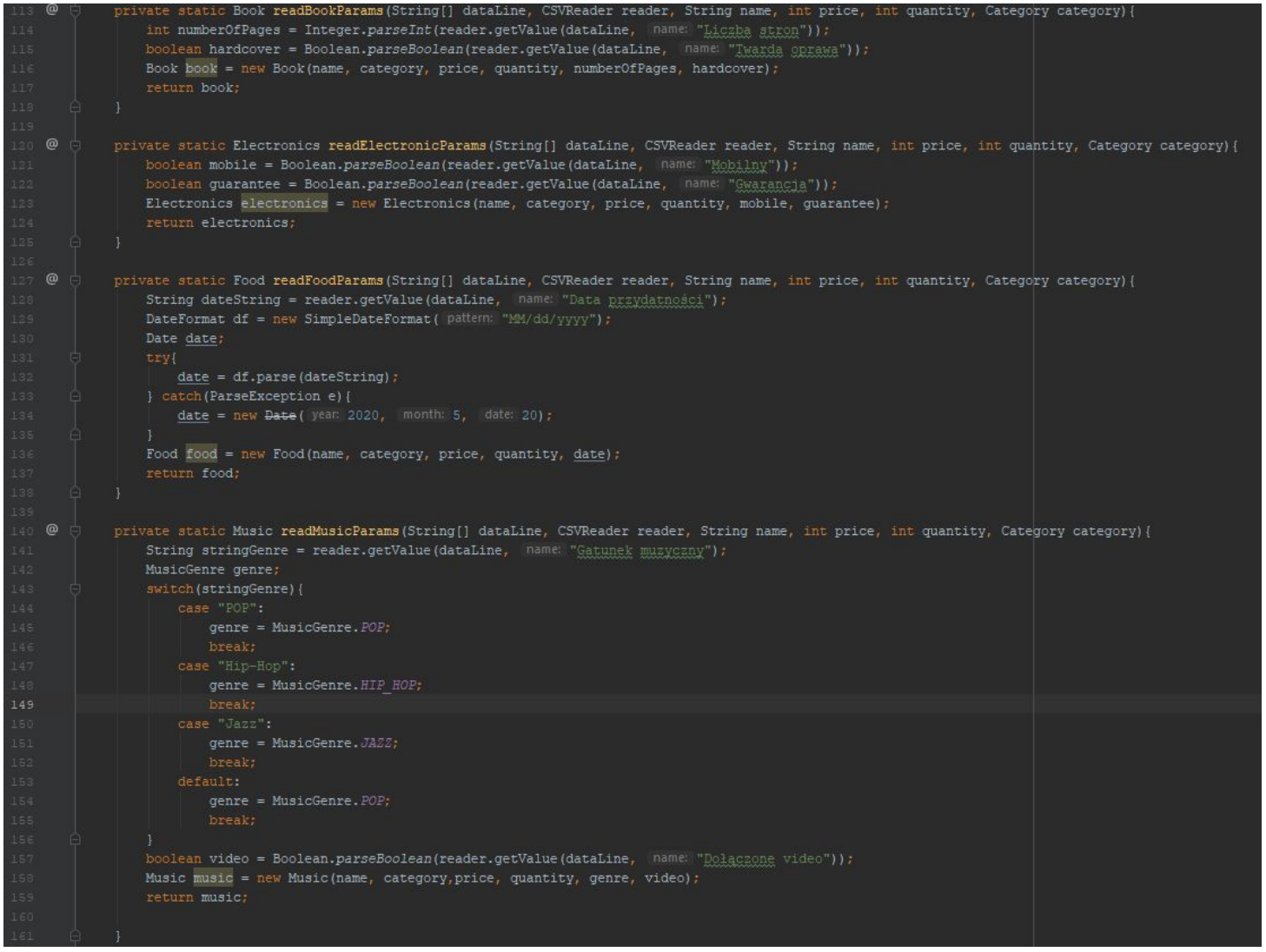
```java
package pl.edu.agh.dronka.shop.model;

import java.util.Date;

public class Food extends Item {
    private Date dateToEat;

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

    public Date getDateToEat() { return dateToEat; }

    public void setDateToEat(Date dateToEat) { this.dateToEat = dateToEat; }
}

```

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

public enum MusicGenre {
    HIP_HOP( displayName: "Hip-hop"), POP( displayName: "POP"), JAZZ( displayName: "Jazz");

    private String displayName;

    public String getDisplayName() { return displayName; }

    private MusicGenre(String displayName) { this.displayName = displayName; }
}

```

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

public class Music extends Item {

    private MusicGenre genre;
    private boolean video;

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

    public MusicGenre getGenre() { return genre; }

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

    public boolean isVideo() { return video; }

    public void setVideo(boolean video) { this.video = video; }
}

```

Żeby produkty odpowiednio wczytywać z pliku, dodałem zmiany w klasie ShopProvider, gdzie dodałem metody które wczytują dodatkowe właściwości w zależności od kategorii produktu, a także zmodyfikowałem metodę readItems.

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

            boolean isPolish = Boolean.parseBoolean(reader.getValue(
                    dataLine, name: "Tanie bo polskie"));
            boolean isSecondhand = Boolean.parseBoolean(reader.getValue(
                    dataLine, name: "Używany"));
            Item item;
            switch(category){
                case BOOKS:
                    item = readBookParams(dataLine, reader, name, price, quantity, category);
                    break;
                case ELECTRONICS:
                    item = readElectronicParams(dataLine, reader, name, price, quantity,  category);
                    break;
                case FOOD:
                    item = readFoodParams(dataLine, reader, name, price, quantity, category);
                    break;
                case MUSIC:
                    item = readMusicParams(dataLine, reader, name, price, quantity, category);
                    break;
                default:
                    item = new Item(name, category, price, quantity);
                    break;
            }
            item.setPolish(isPolish);
            item.setSecondhand(isSecondhand);

            items.add(item);

        }

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

    return items;
```

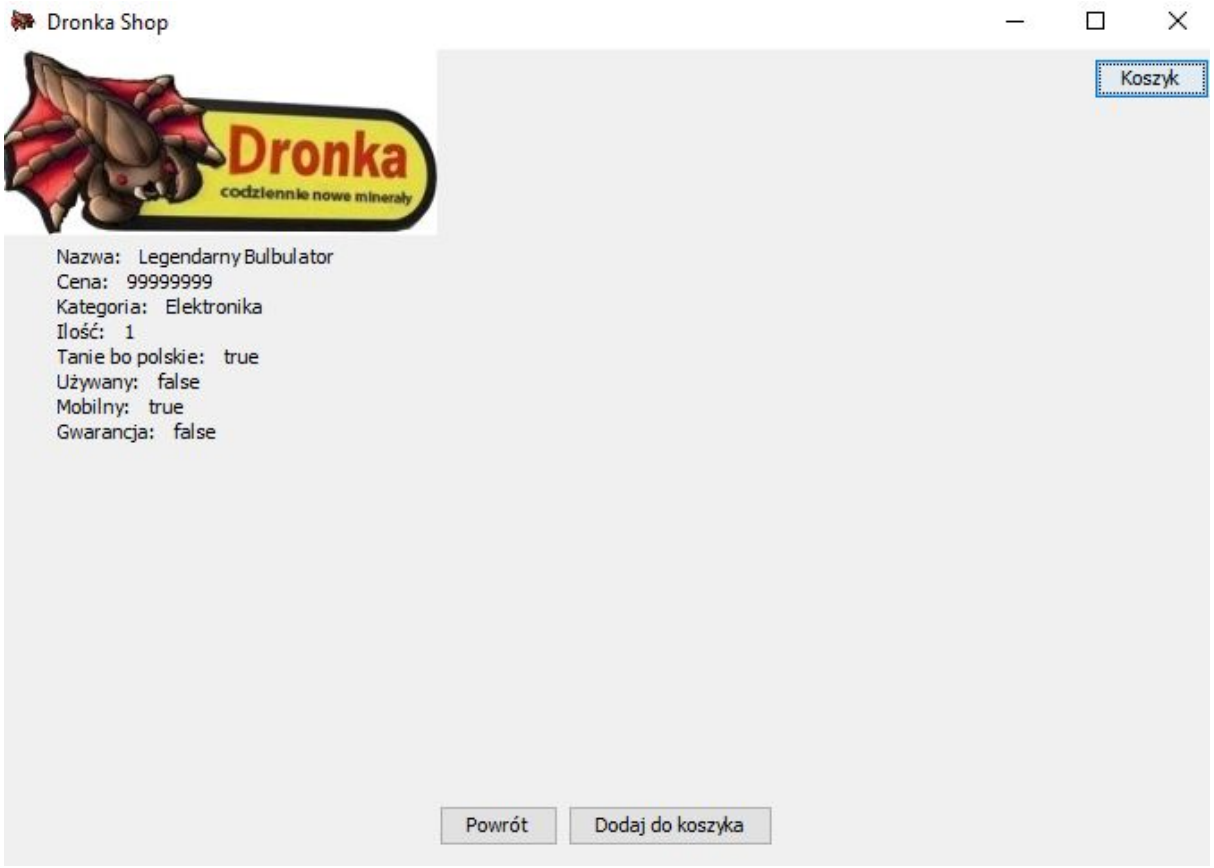


```
    private static Book readBookParams(String[] dataLine, CSVReader reader, String name, int price, int quantity, Category category){
        int numberOfPages = Integer.parseInt(reader.getValue(dataLine,  name: "Liczba stron"));
        boolean hardcover = Boolean.parseBoolean(reader.getValue(dataLine,  name: "Twarda oprawa"));
        Book book = new Book(name, category, price, quantity, numberOfPages, hardcover);
        return book;
    }

    private static Electronics readElectronicParams(String[] dataLine, CSVReader reader, String name, int price, int quantity, Category category){
        boolean mobile = Boolean.parseBoolean(reader.getValue(dataLine,  name: "Mobilny"));
        boolean guarantee = Boolean.parseBoolean(reader.getValue(dataLine,  name: "Gwarancja"));
        Electronics electronics = new Electronics(name, category, price, quantity, mobile, guarantee);
        return electronics;
    }

    private static Food readFoodParams(String[] dataLine, CSVReader reader, String name, int price, int quantity, Category category){
        String dateString = reader.getValue(dataLine,  name: "Data przydatności");
        DateFormat df = new SimpleDateFormat( pattern: "MM/dd/yyyy");
        Date date;
        try{
            date = df.parse(dateString);
        } catch(ParseException e){
            date = new Date( year: 2020,  month: 5,  date: 20);
        }
        Food food = new Food(name, category, price, quantity, date);
        return food;
    }

    private static Music readMusicParams(String[] dataLine, CSVReader reader, String name, int price, int quantity, Category category){
        String stringGenre = reader.getValue(dataLine,  name: "Gatunek muzyczny");
        MusicGenre genre;
        switch(stringGenre){
            case "POP":
                genre = MusicGenre.POP;
                break;
            case "Hip-Hop":
                genre = MusicGenre.HIP_HOP;
                break;
            case "Jazz":
                genre = MusicGenre.JAZZ;
                break;
            default:
                genre = MusicGenre.POP;
                break;
        }
        boolean video = Boolean.parseBoolean(reader.getValue(dataLine,  name: "Dołączone video"));
        Music music = new Music(name, category,price, quantity, genre, video);
        return music;

    }
```

Następnie, żeby dodatkowe właściwości wyświetlały się w panelu z właściwościami danego produktu, trzeba było je dodać do propertiesMap, mapy która zawierała wyświetlane właściwości. W tym celu zmodyfikowałem klasę PropertiesHelper, dodając metody odpowiedzialne za dodanie odpowiednich właściwości do mapy w zależności od kategorii produktu, a także dokonałem zmiany w klasie getPropertiesMap.

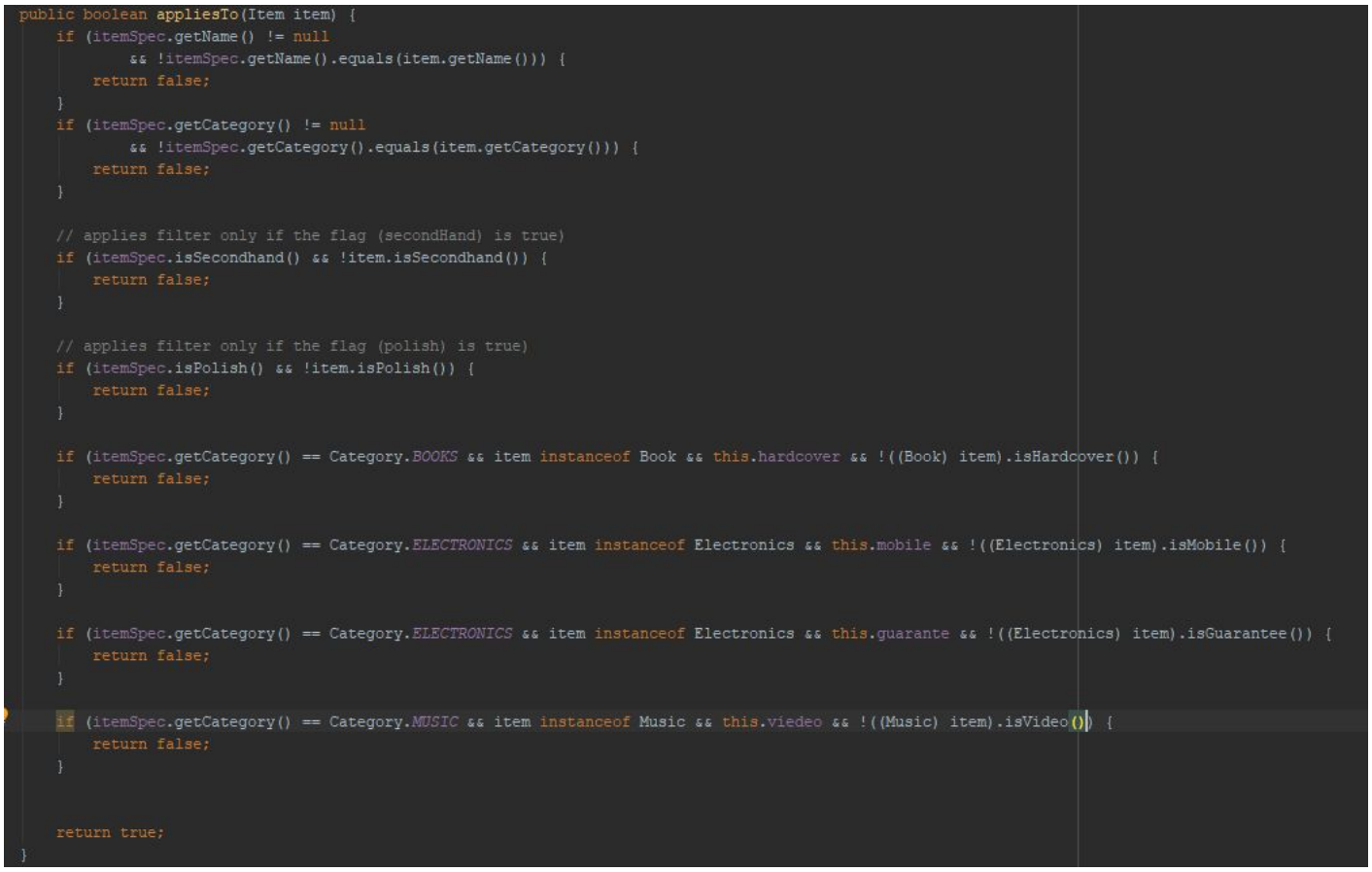
```java
public static Map<String, Object> getPropertiesMap(Item item) {
    Map<String, Object> propertiesMap = new LinkedHashMap<>();

    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());
    switch (item.getCategory()){
        case BOOKS:
            addBookProperties((Book) item, propertiesMap);
            break;
        case ELECTRONICS:
            addElectronicsProperties((Electronics) item, propertiesMap);
            break;
        case FOOD:
            addFoodProperties((Food) item, propertiesMap);
            break;
        case MUSIC:
            addMusicProperties((Music) item, propertiesMap);
            break;
        default:
            break;
    }

    return propertiesMap;
}

private static void addBookProperties(Book book, Map<String, Object> propertiesMap){
    propertiesMap.put("Liczba stron", book.getNumberOfPages());
    propertiesMap.put("Twarda oprawa", book.isHardcover());
}

private static void addElectronicsProperties(Electronics electronics, Map<String, Object> propertiesMap){
    propertiesMap.put("Mobilny", electronics.isMobile());
    propertiesMap.put("Gwarancja", electronics.isGuarantee());
}

private static void addFoodProperties(Food food, Map<String, Object> propertiesMap){
    propertiesMap.put("Data przydatności", food.getDateToEat());
}

private static void addMusicProperties(Music music, Map<String, Object> propertiesMap){
    propertiesMap.put("Gatunek muzyczny", music.getGenre().getDisplayName());
    propertiesMap.put("Dołączone video", music.isVideo());
}

}
```

W rezultacie wyświetlały się dodatkowe właściwości w panelu:



Zadanie 2.2

Żeby dodać filtry w zależności od kategorii, najpierw zmodyfikowałem klasę ItemFilter, by pozwalała dodawać filtry na nowe pola. Utworzyłem odpowiednie atrybuty w klasie ItemFilter wraz z odpowiednimi getterami i setterami, a także zmodyfikowałem metodę appliesTo.



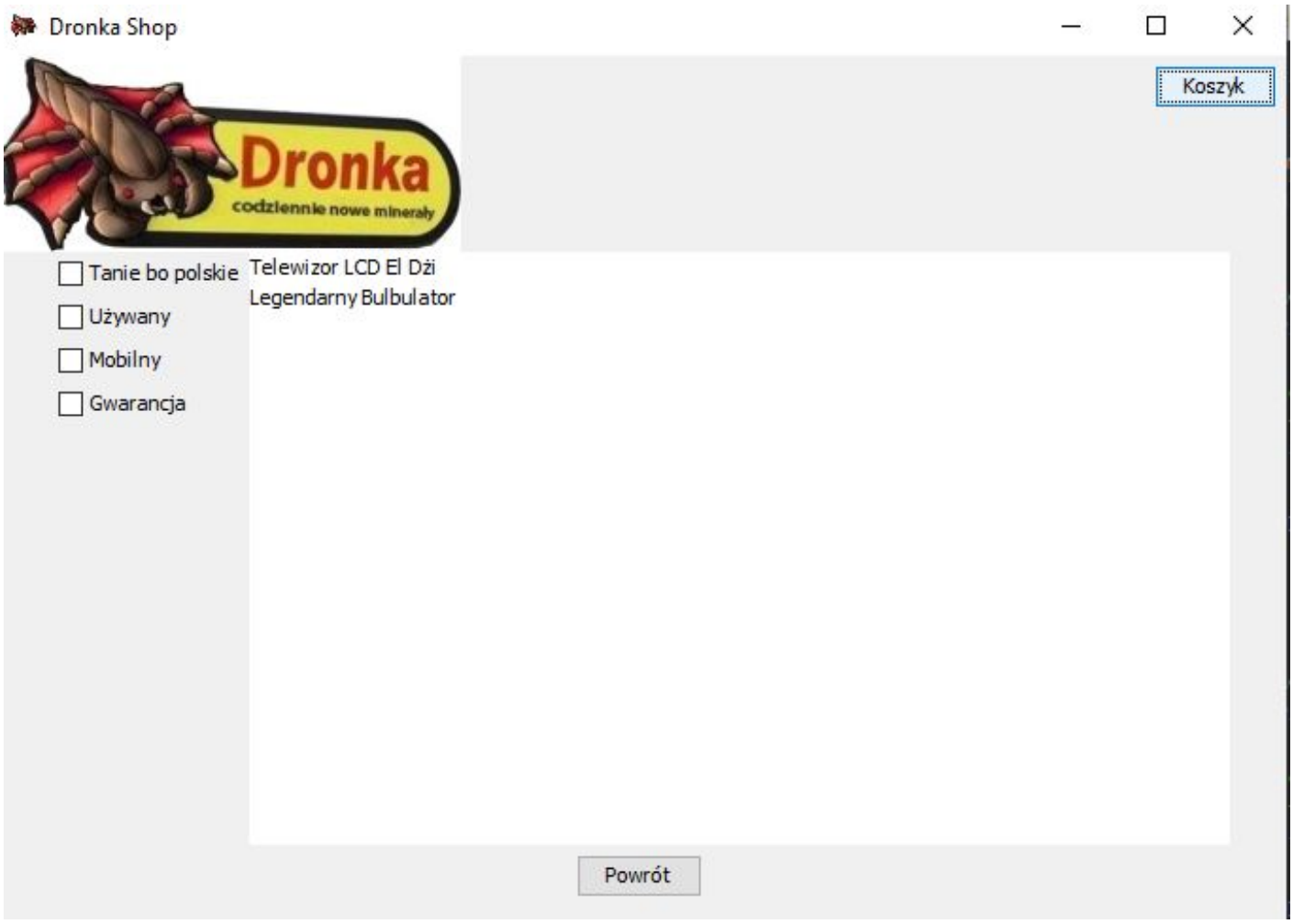
```
public boolean appliesTo(Item item) {
    if (itemSpec.getName() != null
            && !itemSpec.getName().equals(item.getName())) {
        return false;
    }
    if (itemSpec.getCategory() != null
            && !itemSpec.getCategory().equals(item.getCategory())) {
        return false;
    }

    // applies filter only if the flag (secondHand) is true)
    if (itemSpec.isSecondhand() && !item.isSecondhand()) {
        return false;
    }

    // applies filter only if the flag (polish) is true)
    if (itemSpec.isPolish() && !item.isPolish()) {
        return false;
    }

    if (itemSpec.getCategory() == Category.BOOKS && item instanceof Book && this.hardcover && !((Book) item).isHardcover()) {
        return false;
    }

    if (itemSpec.getCategory() == Category.ELECTRONICS && item instanceof Electronics && this.mobile && !((Electronics) item).isMobile()) {
        return false;
    }

    if (itemSpec.getCategory() == Category.ELECTRONICS && item instanceof Electronics && this.guarante && !((Electronics) item).isGuarantee()) {
        return false;
    }

    if (itemSpec.getCategory() == Category.MUSIC && item instanceof Music && this.viedeo && !((Music) item).isVideo()) {
        return false;
    }


    return true;
}
```

Żeby filtry wyświetlały się w panelu, musiałem go odpowiednio zmodyfikować dodajać odpowiednie checkboxy w zależności od kategorii. Zmodyfikowałem klasę PropertiesPanel, gdzie dodałem funkcje które tworzą odpowiednie checkboxy, a także zmodyfikowałem metodę fillProperties.

```java
public void fillProperties() {
    removeAll();

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

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

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

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

    switch(shopController.getCurrentCategory()){
        case BOOKS:
            addBookProperties();
            break;
        case ELECTRONICS:
            addElectronicsProperties();
            break;
        case MUSIC:
            addMusicProperties();
            break;
        default:
            break;
    }

}
```

```
private void addBookProperties(){
    add(createPropertyCheckbox( propertyName: "Twarda oprawa", new ActionListener() {

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

private void addElectronicsProperties(){
    add(createPropertyCheckbox( propertyName: "Mobilny", new ActionListener() {

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

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

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

private void addMusicProperties(){
    add(createPropertyCheckbox( propertyName: "Dołączone video", new ActionListener() {

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

Efekt końcowy był zgodny z wymaganiami i można było teraz filtrować po odpowiednich właściwościach.


Dronka Shop
Koszyk
Dronka
codziennie nowe minerały
Tanie bo polskie
Używany
Mobilny
Gwarancja
Telewizor LCD El Dżi
Legendarny Bulbulator
Powrót