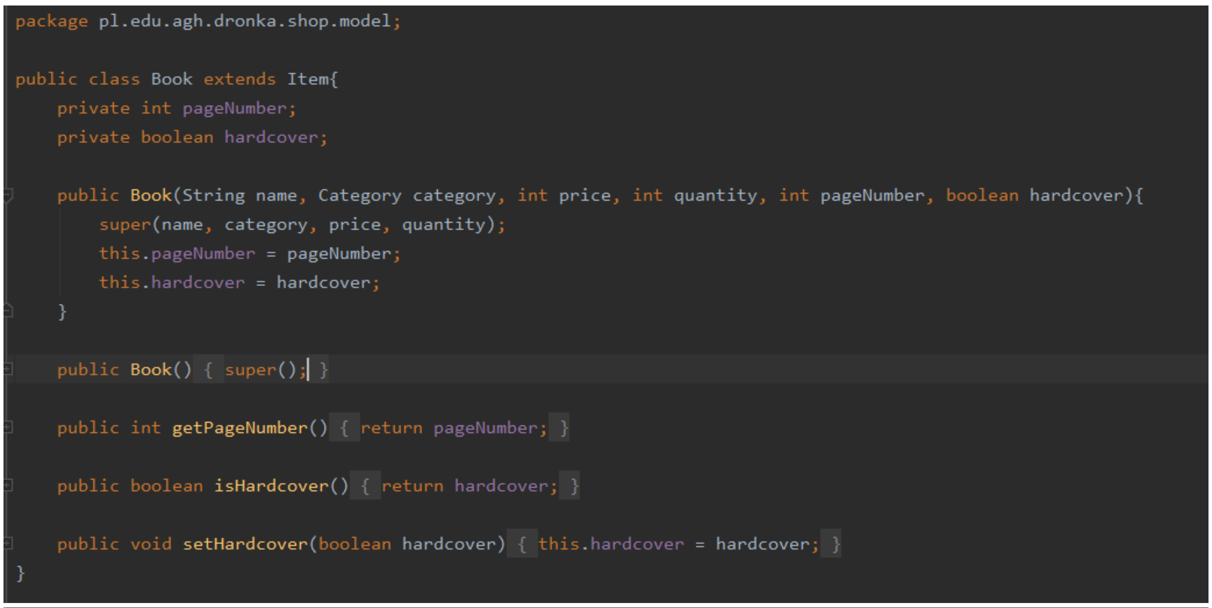
Sprawozdanie z ćwiczenia 2

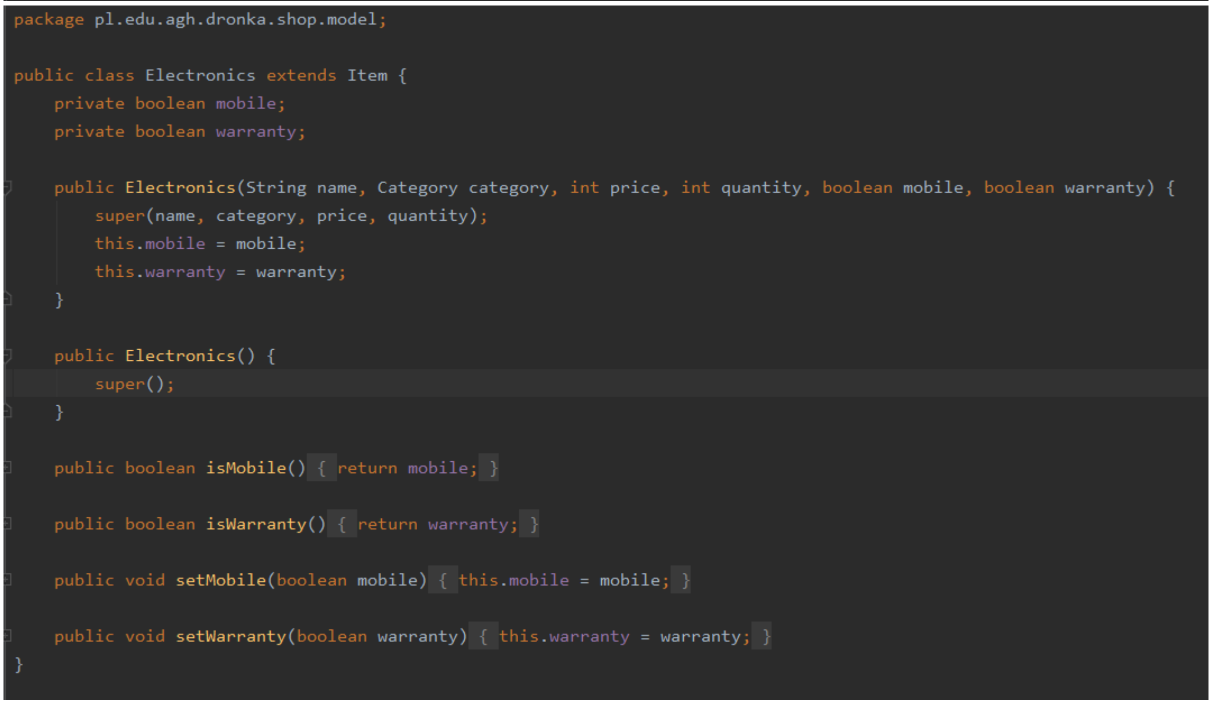
Krzysztof Nalepa i Jakub Solecki

Zadanie 1

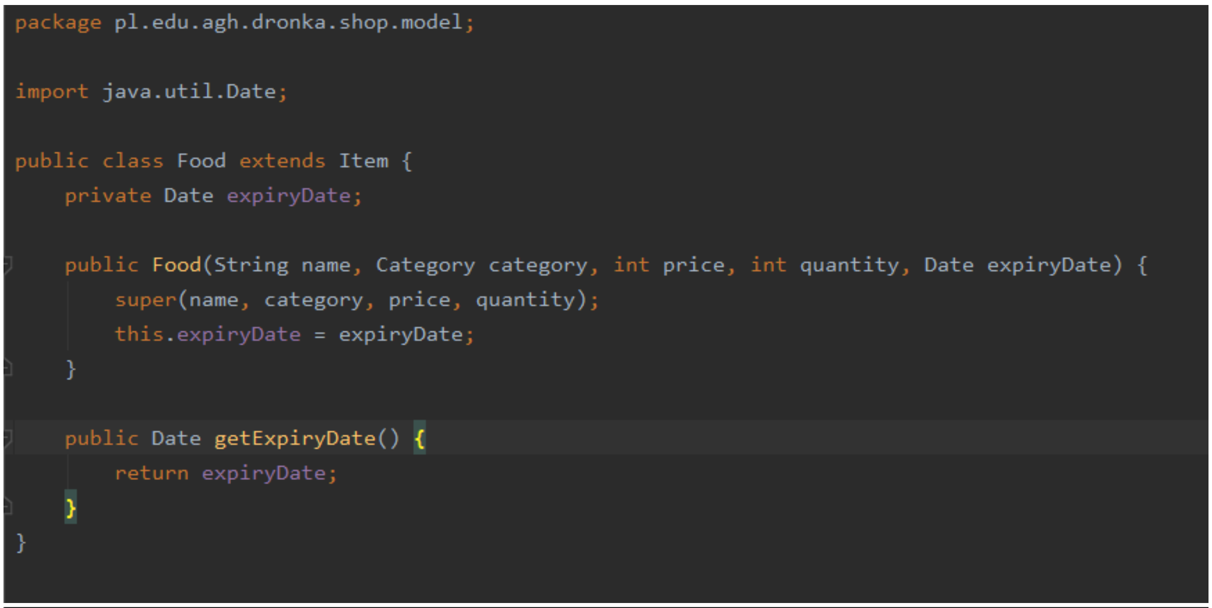
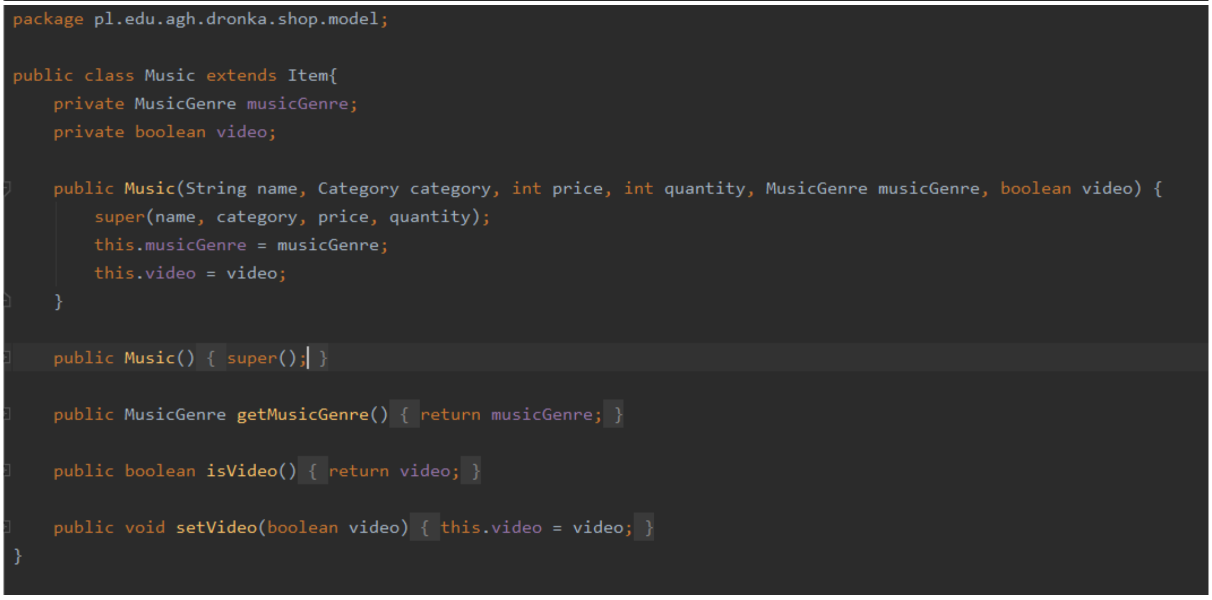
W celu dodania dodatkowych atrybutów dla każdej kategorii stworzyliśmy osobną klasę dla każdej z kategorii która dziedziczy z klasy Item. Dla każdej po za kategorią Sport która nie wnosi żadnych nowych atrybutów

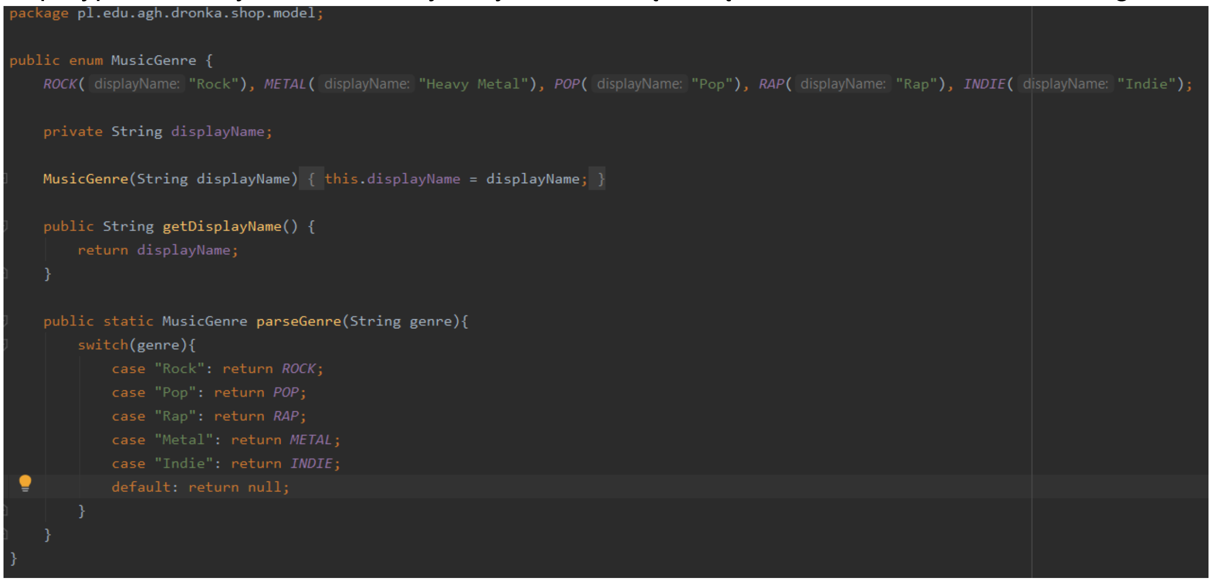


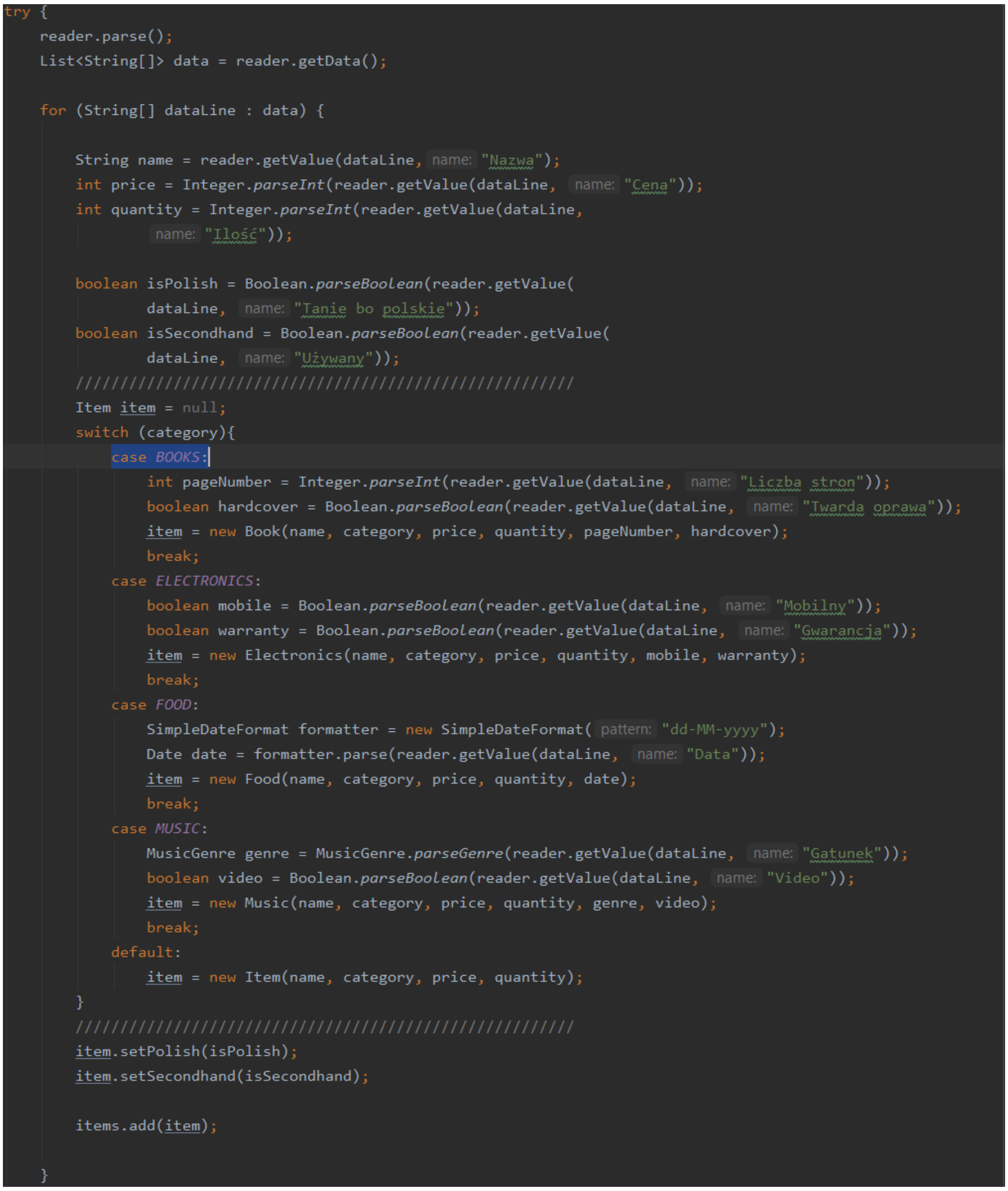
```java
package pl.edu.agh.dronka.shop.model;

public class Book extends Item{
    private int pageNumber;
    private boolean hardcover;

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

    public Book() { super(); }

    public int getPageNumber() { return pageNumber; }

    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 warranty;

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

    public Electronics() {
        super();
    }

    public boolean isMobile() { return mobile; }

    public boolean isWarranty() { return warranty; }

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

    public void setWarranty(boolean warranty) { this.warranty = warranty; }
}
```

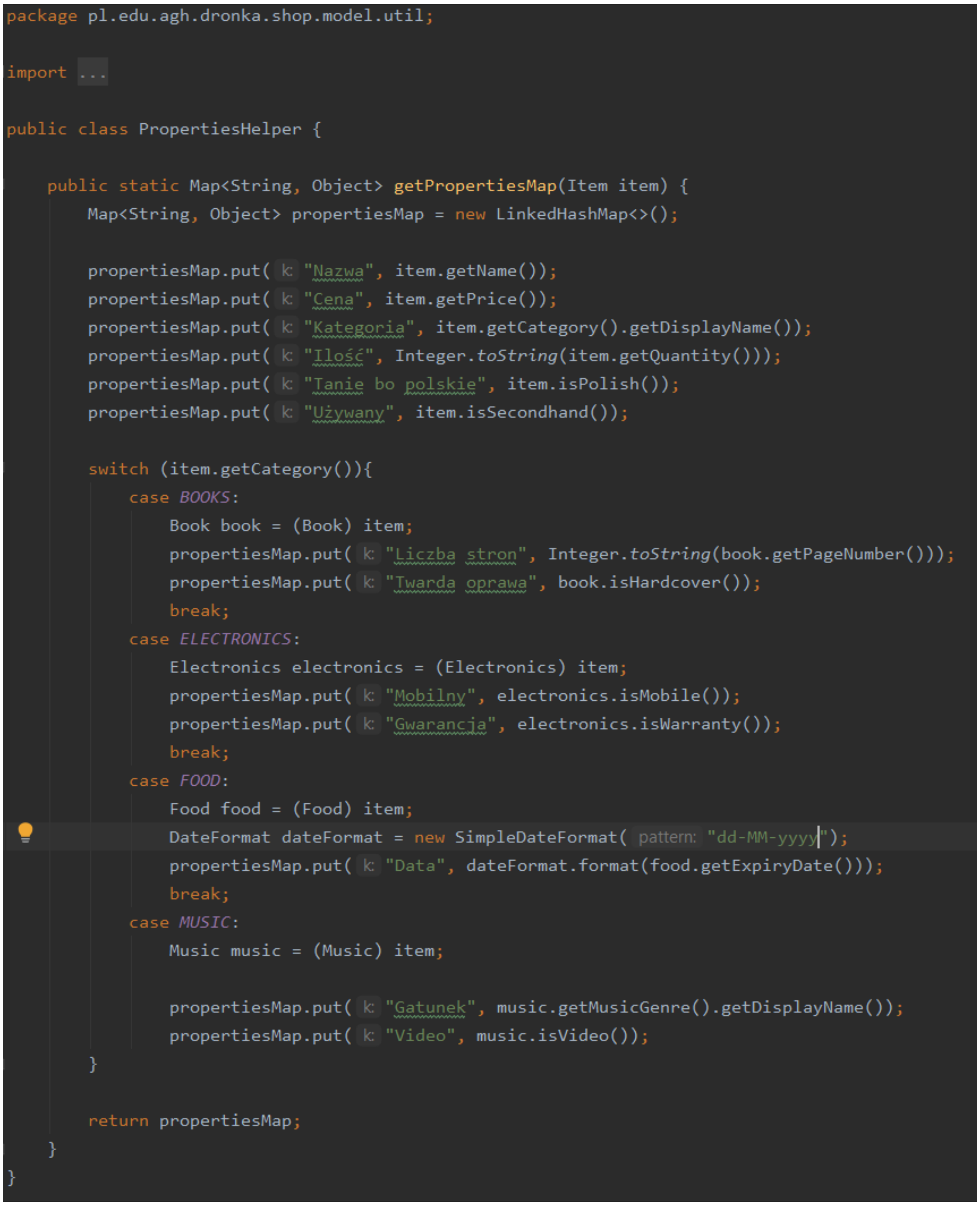

```java
package pl.edu.agh.dronka.shop.model;

import java.util.Date;

public class Food extends Item {
    private Date expiryDate;

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

    public Date getExpiryDate() {
        return expiryDate;
    }
}
```

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

public class Music extends Item{
    private MusicGenre musicGenre;
    private boolean video;

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

    public Music() { super(); }

    public MusicGenre getMusicGenre() { return musicGenre; }

    public boolean isVideo() { return video; }

    public void setVideo(boolean video) { this.video = video; }
}
```
W przypadku klasy Music stworzyliśmy dodatkową klasę enum w celu określenia kategorii


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

public enum MusicGenre {
    ROCK( displayName: "Rock"), METAL( displayName: "Heavy Metal"), POP( displayName: "Pop"), RAP( displayName: "Rap"), INDIE( displayName: "Indie");

    private String displayName;

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

    public String getDisplayName() {
        return displayName;
    }

    public static MusicGenre parseGenre(String genre){
        switch(genre){
            case "Rock": return ROCK;
            case "Pop": return POP;
            case "Rap": return RAP;
            case "Metal": return METAL;
            case "Indie": return INDIE;
            default: return null;
        }
    }
}
```
Następnie zmodyfikowaliśmy funkcję readItems z klasy ShopProvider



```
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 = null;
        switch (category){
            case BOOKS:
                int pageNumber = Integer.parseInt(reader.getValue(dataLine, name: "Liczba stron"));
                boolean hardcover = Boolean.parseBoolean(reader.getValue(dataLine, name: "Twarda oprawa"));
                item = new Book(name, category, price, quantity, pageNumber, hardcover);
                break;
            case ELECTRONICS:
                boolean mobile = Boolean.parseBoolean(reader.getValue(dataLine, name: "Mobilny"));
                boolean warranty = Boolean.parseBoolean(reader.getValue(dataLine, name: "Gwarancja"));
                item = new Electronics(name, category, price, quantity, mobile, warranty);
                break;
            case FOOD:
                SimpleDateFormat formatter = new SimpleDateFormat( pattern: "dd-MM-yyyy");
                Date date = formatter.parse(reader.getValue(dataLine, name: "Data"));
                item = new Food(name, category, price, quantity, date);
                break;
            case MUSIC:
                MusicGenre genre = MusicGenre.parseGenre(reader.getValue(dataLine, name: "Gatunek"));
                boolean video = Boolean.parseBoolean(reader.getValue(dataLine, name: "Video"));
                item = new Music(name, category, price, quantity, genre, video);
                break;
            default:
                item = new Item(name, category, price, quantity);
        }
        ////////////////////////////////////////////////////////
        item.setPolish(isPolish);
        item.setSecondhand(isSecondhand);

        items.add(item);

    }
```

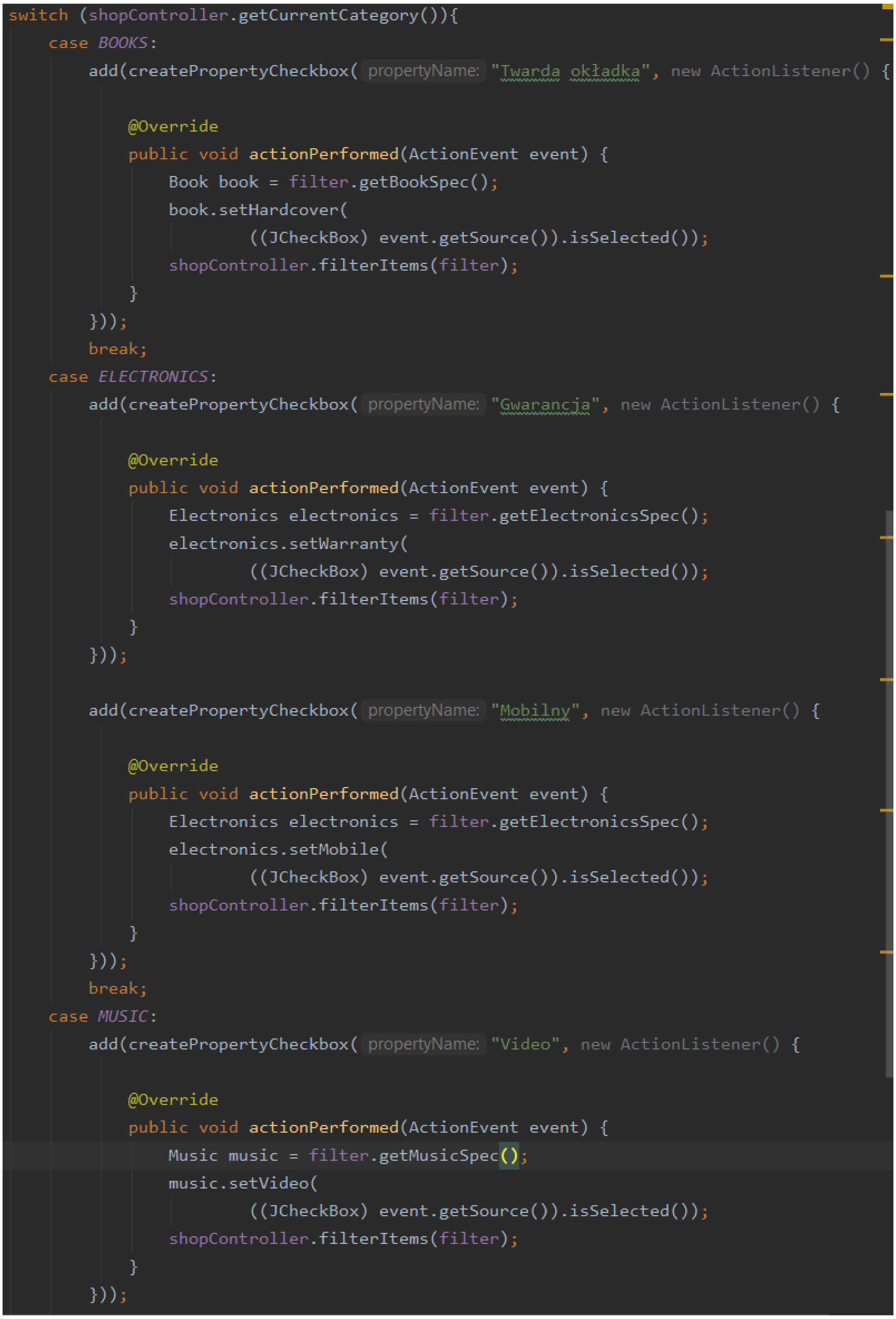
Następnie zmodyfikowaliśmy klasę ProportiesHelper



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

import ...

public class PropertiesHelper {

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

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

        switch (item.getCategory()){
            case BOOKS:
                Book book = (Book) item;
                propertiesMap.put( k: "Liczba stron", Integer.toString(book.getPageNumber()));
                propertiesMap.put( k: "Twarda oprawa", book.isHardcover());
                break;
            case ELECTRONICS:
                Electronics electronics = (Electronics) item;
                propertiesMap.put( k: "Mobilny", electronics.isMobile());
                propertiesMap.put( k: "Gwarancja", electronics.isWarranty());
                break;
            case FOOD:
                Food food = (Food) item;
                DateFormat dateFormat = new SimpleDateFormat( pattern: "dd-MM-yyyy");
                propertiesMap.put( k: "Data", dateFormat.format(food.getExpiryDate()));
                break;
            case MUSIC:
                Music music = (Music) item;

                propertiesMap.put( k: "Gatunek", music.getMusicGenre().getDisplayName());
                propertiesMap.put( k: "Video", music.isVideo());
        }

        return propertiesMap;
    }
}
```

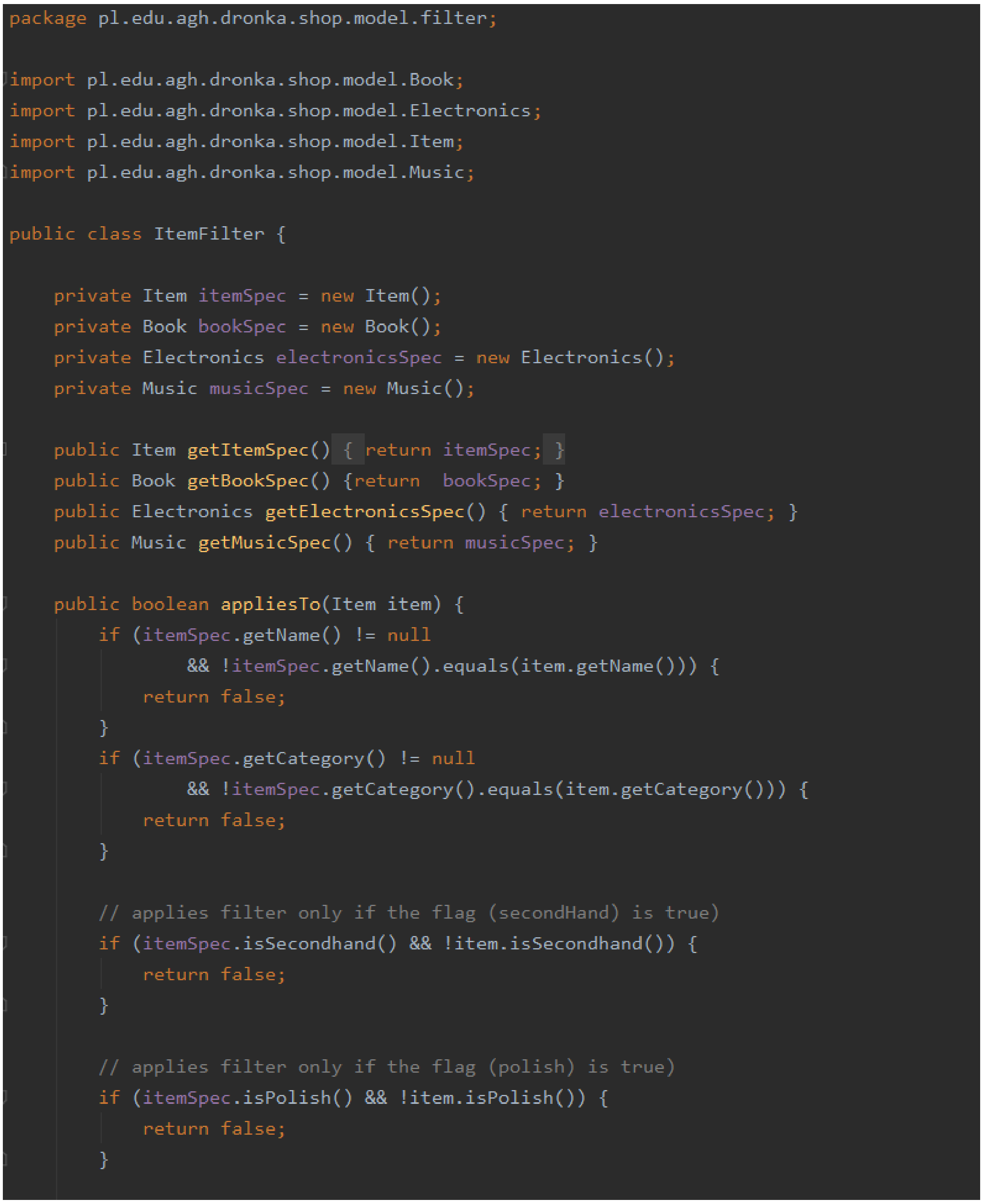
Zadanie 2

Dodaliśmy switcha w klasie PropertiesPanel

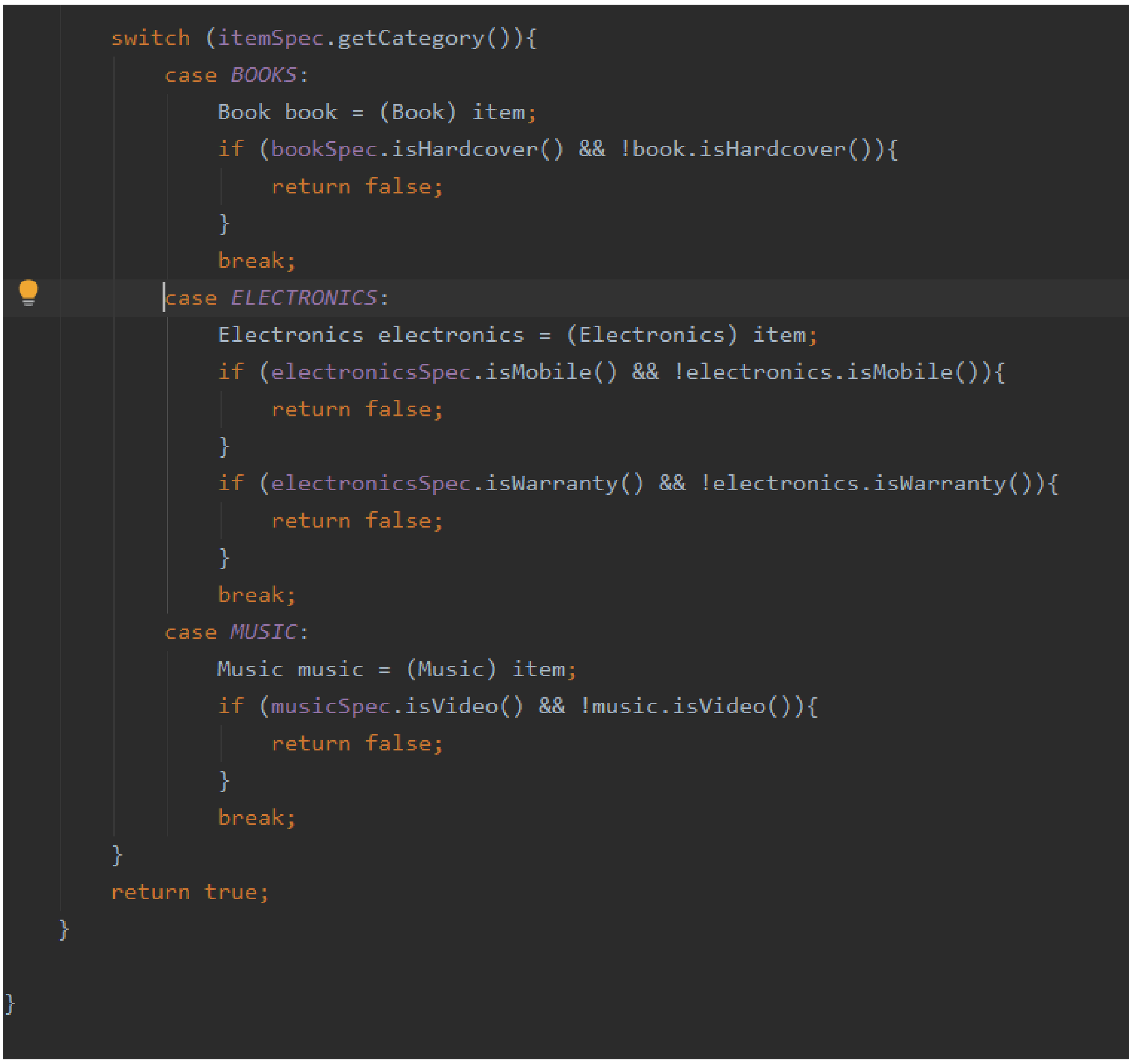


```
switch (shopController.getCurrentCategory()){
    case BOOKS:
        add(createPropertyCheckbox( propertyName: "Twarda okładka", new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent event) {
                Book book = filter.getBookSpec();
                book.setHardcover(
                        ((JCheckBox) event.getSource()).isSelected());
                shopController.filterItems(filter);
            }
        }));
        break;
    case ELECTRONICS:
        add(createPropertyCheckbox( propertyName: "Gwarancja", new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent event) {
                Electronics electronics = filter.getElectronicsSpec();
                electronics.setWarranty(
                        ((JCheckBox) event.getSource()).isSelected());
                shopController.filterItems(filter);
            }
        }));

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

            @Override
            public void actionPerformed(ActionEvent event) {
                Electronics electronics = filter.getElectronicsSpec();
                electronics.setMobile(
                        ((JCheckBox) event.getSource()).isSelected());
                shopController.filterItems(filter);
            }
        }));
        break;
    case MUSIC:
        add(createPropertyCheckbox( propertyName: "Video", new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent event) {
                Music music = filter.getMusicSpec();
                music.setVideo(
                        ((JCheckBox) event.getSource()).isSelected());
                shopController.filterItems(filter);
            }
        }));
```

Oraz zmodyfikowaliśmy klasę ItemFilter



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

import pl.edu.agh.dronka.shop.model.Book;
import pl.edu.agh.dronka.shop.model.Electronics;
import pl.edu.agh.dronka.shop.model.Item;
import pl.edu.agh.dronka.shop.model.Music;

public class ItemFilter {

    private Item itemSpec = new Item();
    private Book bookSpec = new Book();
    private Electronics electronicsSpec = new Electronics();
    private Music musicSpec = new Music();

    public Item getItemSpec() { return itemSpec; }
    public Book getBookSpec() {return  bookSpec; }
    public Electronics getElectronicsSpec() { return electronicsSpec; }
    public Music getMusicSpec() { return musicSpec; }

    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;
        }
```



```java
        switch (itemSpec.getCategory()){
            case BOOKS:
                Book book = (Book) item;
                if (bookSpec.isHardcover() && !book.isHardcover()){
                    return false;
                }
                break;
            case ELECTRONICS:
                Electronics electronics = (Electronics) item;
                if (electronicsSpec.isMobile() && !electronics.isMobile()){
                    return false;
                }
                if (electronicsSpec.isWarranty() && !electronics.isWarranty()){
                    return false;
                }
                break;
            case MUSIC:
                Music music = (Music) item;
                if (musicSpec.isVideo() && !music.isVideo()){
                    return false;
                }
                break;
        }
        return true;
    }

}
```