WPROWADZANIE ZMIAN W APLIKACJI

1. Dodatkowe właściwości dla produktów w poszczególnych kategoriach

Rozszerz aplikacje dronka-shop tak, aby na stronie “Szczegóły produktu” wyświetlała właściwości, specyficzne dla danej kategorii produktów. Zakłada się, że kategorie będą miały następujące (dodatkowe) właściwości:

1. Książki:
• liczba stron : int
• twarda oprawa : boolean

2. Elektronika:
• mobilny : boolean
• gwarancja : boolean

3. Żywność:
• data przydatności do spożycia : Date

4. Muzyka:
• gatunek muzyczny : enum
• dołączone video : boolean

5. Sport - brak dodatkowych właściwości

a. Stworzono kilka klas dziedziczących po klasie `Item`.

- klasa `Book`:

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

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

public class Book extends Item {

    private final int pageNumber;
    private final 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 int getPageNumber() {
        return pageNumber;
    }

    public boolean isHardCover() {
        return hardCover;
    }
}
```

- klasa `Device`:

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

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

public class Device extends Item {

    private final boolean mobile;
    private final boolean warranty;

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

    public boolean isMobile() {
        return mobile;
    }

    public boolean isWarranty() {
        return warranty;
    }
}
```

- klasa `Food`:

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

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

import java.util.Date;

public class Food extends Item {

    private final Date eatByDate;

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

    public Date getEatByDate() {
        return eatByDate;
    }
}
```

- typ wyliczeniowy `MusicGenre`:

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

public enum MusicGenre {
    ROCK, JAZZ, DUBSTEP, TECHNO, BLUES, COUNTRY, POP
}
```

- klasa `Music`:

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

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

public class Music extends Item {

    private MusicGenre musicGenre;
```

```
    private boolean hasVideo;

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

    public MusicGenre getGenre() {
        return musicGenre;
    }

    public boolean isHasVideo() {
        return hasVideo;
    }
}
```

- klasa `Sport`:

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

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);
    }
}
```

b. Zmodyfikowano kod metody `readItems` w klasie `ShopProvider` tak, aby w zależności od przekazanej kategorii metoda ta tworzyła odpowiedni obiekt:

```
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 Book(name, category, price, quantity,
Integer.parseInt(reader.getValue(dataLine, "Liczba stron")),
                        Boolean.parseBoolean(reader.getValue(dataLine, "Twarda oprawa")));
                case ELECTRONICS -> item = new Device(name, category, price, quantity,
                        Boolean.parseBoolean(reader.getValue(dataLine, "Mobilny")),
                        Boolean.parseBoolean(reader.getValue(dataLine, "Gwarancja")));
                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);
                }
                case MUSIC -> item = new Music(name, category, price, quantity,
                        MusicGenre.valueOf(reader.getValue(dataLine, "Gatunek muzyczny")),
                        Boolean.parseBoolean(reader.getValue(dataLine, "Czy ma wideo")));
                case SPORT -> item = new Sport(name, category, price, quantity);
                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;
}
```

c. w klasie PropertiesHelper zmodyfikowano metodę `getPropertiesMap`:

```
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 Book) {
        propertiesMap.put("Liczba stron", ((Book)
item).getPageNumber());
        propertiesMap.put("Twarda oprawa", ((Book)
item).isHardCover());
    } else if (item instanceof Device) {
        propertiesMap.put("Mobilny", ((Device) item).isMobile());
        propertiesMap.put("Gwarancja", ((Device)
item).isWarranty());
    } else if (item instanceof Food) {
        propertiesMap.put("Data przydatności", ((Food)
item).getEatByDate());
    } else if (item instanceof Music) {
        propertiesMap.put("Gatunek muzyczny", ((Music)
item).getGenre());
        propertiesMap.put("Czy ma wideo", ((Music)
item).isHasVideo());
    }

    return propertiesMap;
}
```

W zależności od typu obiektu, z którym przyjdzie nam pracować, musimy dodać do słownika odpowiednie atrybuty.

d. dostosowano także pliki csv pod nową implementację:

- books.csv:

```
Nazwa,Cena,Ilość,Tanie bo polskie,Używany,Liczba
stron,Twarda oprawa
Nowe Przygody Gangu Czworga,50,2,TRUE,TRUE,500,TRUE
Zamodeluj swoje życie. Technologie obiektowe for
dummies,120,15,FALSE,FALSE,200,FALSE
When The Smoke is Going Down : Wprowadzenie do testów
wydajnościowych,90,3,TRUE,FALSE,2000,TRUE
```

- electronics.csv:

```
Nazwa,Cena,Ilość,Tanie bo
polskie,Używany,Mobilny,Gwarancja
Telewizor LCD El Dżi,2000,10,FALSE,TRUE,FALSE,TRUE
Legendarny Bulbulator,99999999,1,TRUE,FALSE,TRUE,FALSE
```

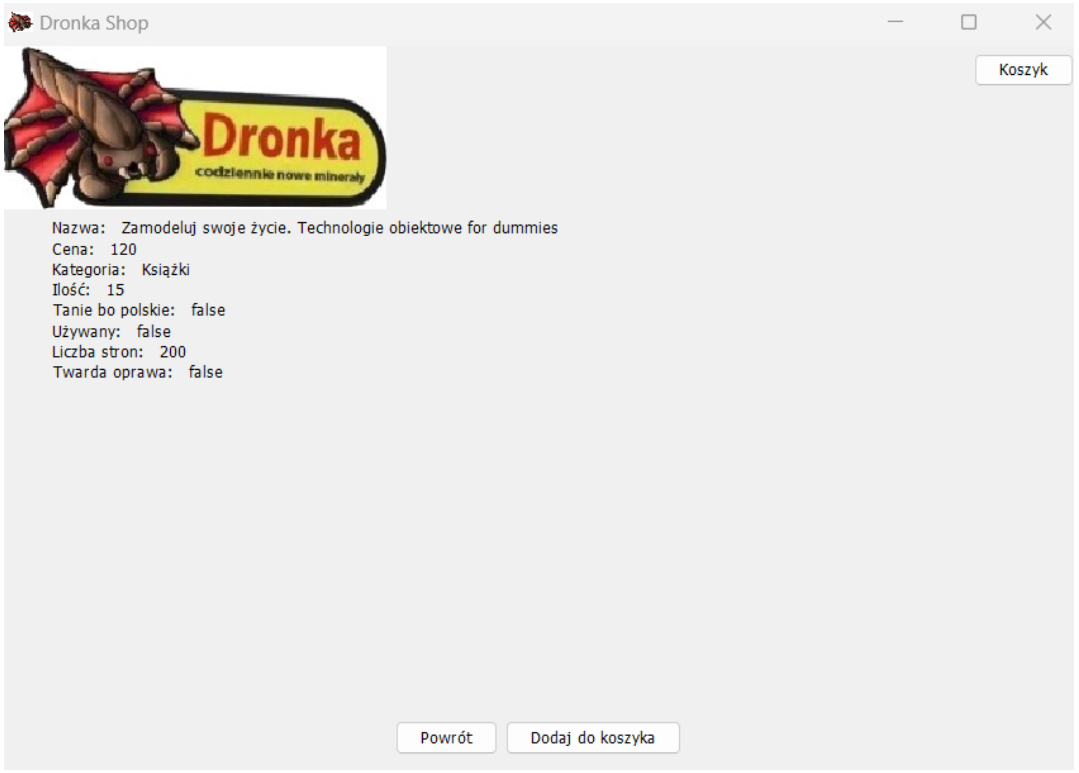
- food.csv:

```
Nazwa,Cena,Ilość,Tanie bo polskie,Używany,Data
przydatności
Zupa Studencka Instant,2,100,TRUE,FALSE,2022-10-10
Sznycel mrożony,6,30,TRUE,FALSE,2022-11-11
```

- music.csv:

```
Nazwa,Cena,Ilość,Tanie bo polskie,Używany,Gatunek
muzyczny,Czy ma wideo
Ciley Myrus - Big Ball of Mud,60,20,TRUE,FALSE,ROCK,FALSE
```

e. widok zmodyfikowanej aplikacji:



2. Filtrowanie produktów po właściwościach specyficznych dla danej kategorii.

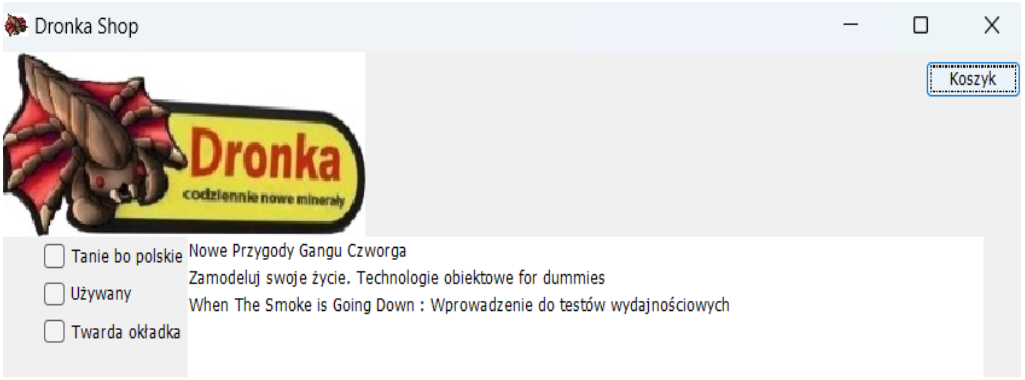
a. Dodano settery do klas dziedziczących po klasie `Item` (`Book`, `Device`, `Food`, `Music`) i przy okazji usunięto słowo final z atrybutów klas, których settery dotyczą.

b. W klasie `PropertiesPanel` dodano switch case'a tak, aby w zależności od typu obiektu, z którym pracujemy, stworzyć odpowiedniego checkboxa.

```
switch (shopController.getCurrentCategory()) {
    case BOOKS -> add(createPropertyCheckbox("Twarda okładka", event -> {
        if (!(item instanceof Book)) return;
        ((Book) item).setHardCover(((JCheckBox) event.getSource()).isSelected());
        shopController.filterItems(filter);
    }));
    case ELECTRONICS -> {
        add(createPropertyCheckbox("Mobilny", event -> {
            if (!(item instanceof Device)) return;
            ((Device) item).setMobile(((JCheckBox) event.getSource()).isSelected());
            shopController.filterItems(filter);
        }));
        add(createPropertyCheckbox("Gwarancja", event -> {
            ((Device) item).setWarranty(((JCheckBox)
event.getSource()).isSelected());
```

```
            shopController.filterItems(filter);
    }));
    }
    case MUSIC -> add(createPropertyCheckbox("Wideo", event -> {
        if (!(item instanceof Music)) return;
        ((Music) item).setVideo(((JCheckBox) event.getSource()).isSelected());
        shopController.filterItems(filter);
    }));
}
```

c. Po uruchomieniu aplikacji możemy zobaczyć nowe checkboxy:



d. Dodano do każdej klasy dziedziczącej po klasie `Item` konstruktor bezargumentowy.

e. Zmodyfikowano klasę `ItemFilter` tak, aby implementowała funkcjonalność filtrowania także po cechach szczególnych dla konkretnego obiektu:

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

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

public class ItemFilter {

    private final Item itemSpec;

    public ItemFilter(Category category) {
        switch (category) {
            case BOOKS -> itemSpec = new Book();
            case ELECTRONICS -> itemSpec = new Device();
            case FOOD -> itemSpec = new Food();
            case MUSIC -> itemSpec = new Music();
            case SPORT -> itemSpec = new Sport();
            default -> itemSpec = new Item();
```

```
        }
    }

    public Item getItemSpec() {
        return itemSpec;
    }

    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;
        }

        switch (item.getCategory()) {
            case BOOKS -> {
                if (((item instanceof Book) && (itemSpec instanceof
Book)) && (((Book) itemSpec).isHardCover() && !((Book)
item).isHardCover())) {
                    return false;
                }
            }
            case ELECTRONICS -> {
                if (((item instanceof Device) && (itemSpec instanceof
Device)) && (((Device) itemSpec).isMobile() && !((Device)
item).isMobile())) {
                    return false;
                }
                if (((item instanceof Device) && (itemSpec instanceof
Device)) && (((Device) itemSpec).isWarranty() && !((Device)
item).isWarranty())) {
                    return false;
                }
```

```
            }
            case MUSIC -> {
                if (((item instanceof Music) && (itemSpec instanceof
Music)) && (((Music) itemSpec).isHasVideo() && !((Music)
item).isHasVideo())) {
                    return false;
                }
            }
        }
        return true;
    }
}
```

f. Widok aplikacji wyświetlającej tylko polskie używane książki: