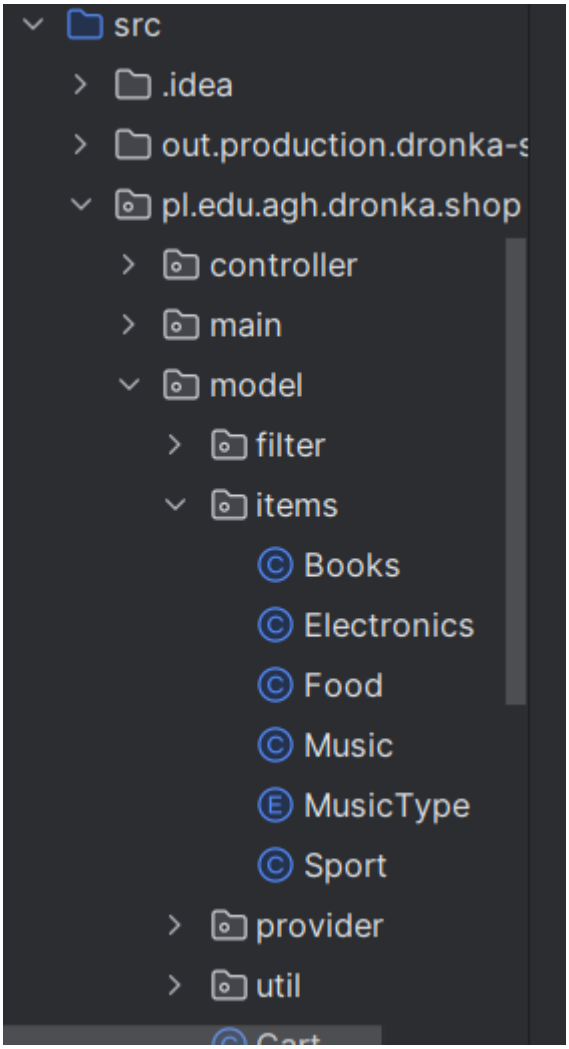
Mateusz Bobula

Zadanie 1

Dodanie klas dziedziczących z item oraz enuma dla klasy Music



Books

```java
public class Books extends Item {
    3 usages
    private int pageAmount;
    3 usages
    private boolean isHardCovered;
    1 usage
    public Books(String name, Category category, int price, int quantity, int pageAmount, boolean isHardCovered) {
        super(name, category, price, quantity);
        this.pageAmount = pageAmount;
        this.isHardCovered = isHardCovered;
    }
    1 usage
    public Books(){category = Category.BOOKS;}

    no usages
    public boolean isHardCovered() { return isHardCovered; }

    1 usage
    public int getPageAmount() { return pageAmount; }

    no usages
    public void setPageAmount(int pageAmount) { this.pageAmount = pageAmount; }
    1 usage
    public void setHardCovered(boolean hardCovered) { isHardCovered = hardCovered; }
}
```

Electronics

```java
public class Food extends Item {
    3 usages
    private LocalDate dateOfConsumption;

    1 usage
    public Food(String name, Category category, int price, int quantity, LocalDate dateOfConsumption) {
        super(name, category, price, quantity);
        this.dateOfConsumption = dateOfConsumption;
    }
    1 usage
    public Food(){category = Category.FOOD;}
    1 usage
    public LocalDate getDateOfConsumption() { return dateOfConsumption; }

    no usages
    public void setDateOfConsumption(LocalDate dateOfConsumption) { this.dateOfConsumption = dateOfConsumption; }
}



    no usages
    public boolean hasWarranty() { return hasWarranty; }

    1 usage
    public void setMobile(boolean isMobile) { this.isMobile = isMobile; }

    1 usage
    public void setHasWarranty(boolean hasWarranty) { this.hasWarranty = hasWarranty; }
}
```

Food

```java
public class Food extends Item {
    3 usages
    private LocalDate dateOfConsumption;

    1 usage
    public Food(String name, Category category, int price, int quantity, LocalDate dateOfConsumption) {
        super(name, category, price, quantity);
        this.dateOfConsumption = dateOfConsumption;
    }
    1 usage
    public Food(){category = Category.FOOD;}
    1 usage
    public LocalDate getDateOfConsumption() { return dateOfConsumption; }

    no usages
    public void setDateOfConsumption(LocalDate dateOfConsumption) { this.dateOfConsumption = dateOfConsumption; }
}
```

Music

```java
public class Music extends Item {
    3 usages
    private MusicType musicType;
    3 usages
    private boolean hasVideo;

    1 usage
    public Music(String name, Category category, int price, int quantity, MusicType musicType, boolean hasVideo) {
        super(name, category, price, quantity);
        this.musicType = musicType;
        this.hasVideo = hasVideo;
    }
    1 usage
    public Music(){category = Category.MUSIC;}
    1 usage
    public MusicType getMusicType() { return musicType; }

    no usages
    public boolean hasVideo() { return hasVideo; }

    no usages
    public void setMusicType(MusicType musicType) { this.musicType = musicType; }

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

Sport

```java
public class Sport extends Item {
    1 usage
    public Sport(String name, Category category, int price, int quantity) {
        super(name, category, price, quantity);
    }
    1 usage
    public Sport(){category = Category.SPORT;}
}
```

MusicType

```java
public enum MusicType {
    1 usage
    RAP, POP, DISCOPOLO, HIPHOP, OTHER;

    1 usage
    public static MusicType parse(String str){
        return switch (str) {
            case "RAP" -> RAP;
            case "POP" -> POP;
            case "DISCOPOLO" -> DISCOPOLO;
            case "HIPHOP" -> HIPHOP;
            default -> OTHER;
        };
    }
}
```

Zmiany w istniejących klasach

ShopProvider.readItems()

Dodanie switch'a względem typów dziedziczących z Item

```java
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 -> {
                    int pageAmount = Integer.parseInt(reader.getValue(dataLine, name: "Liczba stron"));
                    boolean isHardCovered = Boolean.parseBoolean(reader.getValue(dataLine, name: "Twarda oprawa"));
                    item = new Books(name, category, price, quantity, pageAmount, isHardCovered);
                }

                case ELECTRONICS -> {
                    boolean isMobile = Boolean.parseBoolean(reader.getValue(dataLine, name: "Mobilny"));
                    boolean hasWarranty = Boolean.parseBoolean(reader.getValue(dataLine, name: "Gwarancja"));
                    item = new Electronics(name, category, price, quantity, isMobile, hasWarranty);
                }
                case FOOD -> {
                    String dateString = reader.getValue(dataLine, name: "Data spożycia");
                    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");
                    LocalDate date = LocalDate.parse(dateString, formatter);
                    item = new Food(name, category, price, quantity, date);
                }
                case MUSIC -> {
                    MusicType musicType = MusicType.parse(reader.getValue(dataLine, name: "Typ"));
                    boolean hasVideo = Boolean.parseBoolean(reader.getValue(dataLine, name: "Wideo"));
                    item = new Music(name, category, price, quantity, musicType, hasVideo);
                }
                case SPORT -> item = new Sport(name, category, price, quantity);
                default -> {continue;}
            }

            item.setPolish(isPolish);
            item.setSecondhand(isSecondhand);
            items.add(item);
        }

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

    return items;
}
```

PropertiesHelper.getPropertiesMap()

Tu podobnie

```java
public class PropertiesHelper {

    1 usage
    public static Map<String, Object> getPropertiesMap(Item item) {
        Map<String, Object> propertiesMap = new LinkedHashMap<>();
        Category category = item.getCategory();

        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(category){
            case BOOKS -> {
                propertiesMap.put("Strony", ((Books) item).getPageAmount());
                propertiesMap.put("Twarda okładka", ((Books) item).isHardCovered());
            }
            case ELECTRONICS -> {
                propertiesMap.put("Mobilny", ((Electronics) item).isMobile());
                propertiesMap.put("Gwarancja", ((Electronics) item).hasWarranty());
            }
            case FOOD -> propertiesMap.put("Data spożycia", ((Food) item).getDateOfConsumption());

            case MUSIC -> {
                propertiesMap.put("Typ", ((Music) item).getMusicType());
                propertiesMap.put("Wideo", ((Music) item).hasVideo());
            }
        }

        return propertiesMap;
```

Zadanie 2

ItemFilter

Konstruktor + dodanie dodatkowych opcji filtrowania względem kategorii w appliesTo()

Switch względem kategorii, i poprzez rzutowanie wywoływanie odpowiednich metod

```java
public class ItemFilter {
    18 usages
    private Item itemSpec;
    1 usage
    public ItemFilter(Category category){
        switch(category){
            case BOOKS -> itemSpec = new Books();
            case ELECTRONICS -> itemSpec = new Electronics();
            case FOOD -> itemSpec = new Food();
            case MUSIC -> itemSpec = new Music();
```

```
            case SPORT -> itemSpec = new Sport();
            default -> itemSpec = new Item();
        }
    }

    6 usages
    public Item getItemSpec() { return itemSpec; }
    1 usage
    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 (itemSpec.getCategory()){
            case BOOKS -> {
                if (((Books) itemSpec).isHardCovered() && !((Books) item).isHardCovered()) {
                    return false;
                }
            }
            case ELECTRONICS -> {
                if (((Electronics) itemSpec).hasWarranty() && !((Electronics) item).hasWarranty()) {
                    return false;
                }
                if(((Electronics) itemSpec).isMobile() && !((Electronics) item).isMobile()){
                    return false;
                }
            }
            case MUSIC -> {
                if (((Music) itemSpec).hasVideo() && !((Music) item).hasVideo()) {
                    return false;
                }
            }
        }

        return true;
    }
}
```

PropertiesPanel.fillProperties()

Switch względem kategorii, i poprzez rzutowanie wywoływanie odpowiednich metod

```
public class PropertiesPanel extends JPanel {

    no usages
    private static final long serialVersionUID = -2804446079853846996L;
    9 usages
    private ShopController shopController;

    13 usages
    private ItemFilter filter;

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

    1 usage
    public void fillProperties() {
        removeAll();
        filter = new ItemFilter(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 -> {
                add(createPropertyCheckbox( propertyName: "Twarda oprawa", new ActionListener() {

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

```
                shopController.filterItems(filter);
            }
        }));
    }
    case ELECTRONICS -> {
        add(createPropertyCheckbox( propertyName: "Mobilny", new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent event) {
                ((Electronics)filter.getItemSpec()).setMobile(
                        ((JCheckBox) event.getSource()).isSelected());
                shopController.filterItems(filter);
            }
        }));
        add(createPropertyCheckbox( propertyName: "Gwarancja", new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent event) {
                ((Electronics)filter.getItemSpec()).setHasWarranty(
                        ((JCheckBox) event.getSource()).isSelected());
                shopController.filterItems(filter);
            }
        }));
    }
    case MUSIC -> {
        add(createPropertyCheckbox( propertyName: "Wideo", new ActionListener() {

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

Oraz jeszcze drobne zmiany w tych plikach .csv z resources, bo brakowało np. dla muzyki typu