Dokumentacja Programu Zarządzającego Sprzętem Sportowym

Wstęp

Celem niniejszej dokumentacji jest przedstawienie funkcji oraz działania programu do zarządzania sprzętem sportowym. Program został stworzony w celu ułatwienia monitorowania, dodawania, aktualizacji i usuwania danych dotyczących sprzętu sportowego.

Cel Projektu

Celem projektu jest dostarczenie prostego i intuicyjnego narzędzia do efektywnego zarządzania danymi dotyczącymi sprzętu sportowego. Program umożliwia użytkownikowi dodawanie nowych pozycji, aktualizowanie istniejących oraz usuwanie sprzętu, a także przechowywanie tych informacji w pliku, aby były dostępne nawet po ponownym uruchomieniu programu.

Opis Problemu

W dzisiejszych czasach, gdzie zarządzanie danymi staje się coraz bardziej istotne, organizacje, sklepy sportowe czy pasjonaci fitness potrzebują efektywnego sposobu na śledzenie dostępnego sprzętu. Ręczne utrzymywanie listy lub arkusza kalkulacyjnego może być czasochłonne i podatne na błędy. Nasz program oferuje rozwiązanie tego problemu, umożliwiając skuteczne zarządzanie informacjami o sprzęcie sportowym.

Instrukcja Instalacji

Aby zainstalować aplikację, należy postępować zgodnie z poniższymi krokami:

<u>1. Pobierz Kod Źródłowy:</u>

- Sklonuj repozytorium z kodem źródłowym z https://github.com/QueCatcher/project-java.git lub pobierz go jako plik ZIP i wypakuj.

2. Otwórz Projekt w Środowisku Programistycznym:

- Zaimportuj projekt do preferowanego środowiska programistycznego, np. IntelliJ IDEA lub Eclipse.

3. Uruchom Projekt:

- Uruchom główną klasę `Main.java`, aby rozpocząć działanie aplikacji.

Instrukcja Uruchamiania

1. Rozpocznij Aplikację:

- Po uruchomieniu projektu, wybierz opcję uruchomienia aplikacji z głównej klasy `Main.java`.

2. Menu Główne:

- Po uruchomieniu, zostaniesz powitany w menu głównym, gdzie możesz wybrać różne opcje.

3. Opcje Menu:

- Wybierz odpowiednią opcję z menu, wpisując numer odpowiadający Twojej akcji.

4. Dodawanie Sprzętu:

- Aby dodać nowy sprzęt, wybierz opcję 1, a następnie postępuj zgodnie z instrukcjami wprowadzania danych.

5. Wyświetlanie Dostępnego Sprzętu:

- Opcja 2 pozwala na wyświetlenie dostępnego sprzętu.

6. Aktualizacja Sprzętu:

- Opcja 3 umożliwia aktualizację informacji o istniejącym sprzęcie.

7. Usuwanie Sprzętu:

- Opcja 4 pozwala na usunięcie sprzętu z listy.

8. Zakończenie Aplikacji:

- Ostatnia opcja (numer 5) umożliwia zakończenie działania programu.

Instrukcja Obsługi Aplikacji

- Dodawanie Nowego Sprzętu:

- Wybierz opcję 1, a następnie podaj wymagane informacje, takie jak nazwa, opis, cena i dostępność.

- Wyświetlanie Dostępnego Sprzętu:

- Opcja 2 pozwala na wyświetlenie informacji o dostępnym sprzęcie.

- Aktualizacja Sprzętu:

- Wybierz opcję 3, a następnie podaj identyfikator sprzętu do zaktualizowania oraz nowe informacje.

- Usuwanie Sprzętu:

- Opcja 4 pozwala na usunięcie sprzętu z listy. Podaj identyfikator sprzętu do usunięcia.

- Zakończenie Aplikacji:

- Wybierz opcję 5, aby zakończyć działanie aplikacji.

Fragmenty Kodu:

1. Fragment Kodu z Klasy `SportEquipment.java`:

```
public class SportEquipment {
    private static int lastId = 0;
    // ...
    public SportEquipment(String name, String description, double price, boolean availability) {
        this.id = lastId;
        // ...
        lastId = lastId + 1;
    }
}
```

Opis:

Ten fragment kodu pochodzi z klasy `SportEquipment` i odpowiada za inicjalizację obiektu sprzętu sportowego. Statyczna zmienna `lastId` jest używana do nadawania unikalnych identyfikatorów dla każdego nowego sprzętu.

2. Fragment Kodu z Klasy `DataHandler.java`:

```
public class DataHandler {
    // ...
    public static List<SportEquipment> createEmptyList() {
        return new ArrayList<>();
    }
}
```

Opis:

W tym fragmencie kodu z klasy `DataHandler` znajduje się metoda `createEmptyList()`, która tworzy nową pustą listę obiektów `SportEquipment`. Ta lista jest używana, gdy dane nie mogą być załadowane z

pliku, a program musi utworzyć nową listę.

3. Fragment Kodu z Klasy `EquipmentManager.java`:

```
public class EquipmentManager {

    // ...


    public static void addNewEquipment(Scanner scanner, List<SportEquipment> equipmentList) {

        // ...

        SportEquipment newEquipment = new SportEquipment(name, description, price,
availability);

        equipmentList.add(newEquipment);

        // ...

    }

}
```

Opis:

Ten fragment kodu pochodzi z klasy `EquipmentManager` i przedstawia metodę `addNewEquipment`, która pozwala na dodanie nowego sprzętu do listy `equipmentList`. Użytkownik jest pytany o dane takie jak nazwa, opis, cena i dostępność, a następnie tworzony jest nowy obiekt `SportEquipment` i dodawany do listy.

4. Fragment Kodu z Klasy `Messages.java`:

```
public class Messages {

    // ...

    public static void displayInvalidChoiceMessage() {

        System.out.println(Colors.RED + "Invalid choice. Please try again." + Colors.RESET);
```

```
        }
    }
```

Opis:

Ten fragment kodu z klasy `Messages` odpowiada za wyświetlanie komunikatu o błędnej opcji wyboru. Komunikat ten jest wyświetlany, gdy użytkownik poda nieprawidłowy numer opcji w menu głównym. Komunikat ten jest sformatowany, aby przyciągnąć uwagę użytkownika, używając kolorów.

5. Fragment Kodu - Sprawdzenie Unikalności ID:

```
public class DataHandler {
    // ...
    // Nowa metoda do sprawdzania unikalności ID
    public static boolean isIdUnique(List<SportEquipment> equipmentList, int id) {
        for (SportEquipment equipment : equipmentList) {
            if (equipment.getId() == id) {
                return false; // ID nie jest unikalne
            }
        }
        return true; // ID jest unikalne
    }
}
```

Opis:

Ten fragment kodu przedstawia nową metodę w klasie `DataHandler` - `isIdUnique`, która sprawdza, czy podane ID jest unikalne w kontekście listy sprzętu. Jest to używane, aby zapobiec nadawaniu dwóm sprzętom tego samego ID.

6. Fragment Kodu - Zapis Danych do Pliku:

```
public class DataHandler {
```

```
    // ...
    public static void saveData(List<SportEquipment> equipmentList) {
        try (Writer writer = new FileWriter(FILE_PATH)) {
            gson.toJson(equipmentList, writer);
            System.out.println("Data saved successfully.");
        } catch (IOException e) {
            e.printStackTrace();
            System.out.println("Error saving data.");
        }
    }
}
```

Opis:

Ten fragment kodu przedstawia metodę `saveData` w klasie `DataHandler`, która zapisuje listę sprzętu do pliku JSON. Używa ona obiektu `gson` do przekształcenia listy obiektów `SportEquipment` na format JSON i zapisuje go do pliku o nazwie zdefiniowanej przez `FILE_PATH`.

7. Fragment Kodu - Aktualizacja Sprzętu w Menu Głównym:

```
public class Main {
    // ...
    case 3:
        EquipmentManager.updateEquipment(scanner, equipmentList);
        // Po aktualizacji sprzętu, zapisz listę do pliku
        DataHandler.saveData(equipmentList);
        break;
}
```

Opis:

Ten fragment kodu pochodzi z `Main.java` i pokazuje, jak w menu głównym programu wywołać metodę `updateEquipment` z klasy `EquipmentManager` do aktualizacji istniejącego sprzętu. Po dokonaniu aktualizacji, lista jest zapisywana ponownie do pliku.

8. Fragment Kodu - Usuwanie Sprzętu w Menu Głównym:

```
public class Main {
    // ...
    case 4:
        EquipmentManager.removeEquipment(scanner, equipmentList);
        // Po usunięciu sprzętu, zapisz listę do pliku
        DataHandler.saveData(equipmentList);
        break;
}
```

Opis:

Ten fragment kodu pochodzi również z `Main.java` i ilustruje, jak w menu głównym programu używać metody `removeEquipment` z klasy `EquipmentManager` do usunięcia istniejącego sprzętu. Po usunięciu sprzętu, lista jest zapisywana ponownie do pliku.

Podsumowanie:

Projekt "Sports Equipment Management Program" jest prostym systemem do zarządzania informacjami o sprzęcie sportowym. Program umożliwia dodawanie, aktualizację, usuwanie oraz wyświetlanie dostępnego sprzętu. Głównym celem projektu było stworzenie intuicyjnego narzędzia, które pozwala na skuteczne zarządzanie bazą danych sprzętu sportowego.

Wnioski Końcowe:

1. Skalowalność i Rozszerzalność:

- Projekt został zbudowany z myślą o prostocie i łatwej rozbudowie. Dodanie nowych funkcji czy modyfikacja istniejących nie powinno sprawić problemów, dzięki modularnej strukturze kodu.

2. Bezpieczeństwo Danych:

- Wprowadzono sprawdzenie unikalności ID, aby zapobiec nadawaniu dwóm sprzętom tego samego identyfikatora. Dodatkowo, dane są przechowywane w formacie JSON, co ułatwia zabezpieczenie i przenoszenie informacji.

3. Intuicyjny Interfejs:

- Program oferuje prosty interfejs w konsoli, co sprawia, że jest łatwy w obsłudze nawet dla osób nieobeznanych z technicznymi aspektami programowania.

4. Zarządzanie Pamięcią:

- Wprowadzono mechanizm zapisywania danych do pliku po każdej operacji dodawania, aktualizacji lub usuwania sprzętu. Zapewnia to trwałość danych nawet po zakończeniu działania programu.

5. Możliwość Rozwoju:

- Program może być rozwijany w przyszłości, na przykład poprzez dodanie nowych funkcji, takich jak filtrowanie sprzętu, statystyki, czy obsługa różnych typów sprzętu.

Rekomendacje:

Projekt stanowi solidną podstawę do dalszego rozwoju, zwłaszcza jeśli celem jest stworzenie bardziej zaawansowanego systemu zarządzania sprzętem sportowym. Dalsze prace nad programem mogą obejmować implementację interfejsu graficznego, dodanie funkcji wyszukiwania, czy integrację z bazą

danych. Warto również kontynuować dbałość o bezpieczeństwo danych i zoptymalizować kod pod kątem wydajności.