Dokumentacja Projektu - Spis Klas

`Worker`

Klasa `Worker` reprezentuje pracownika firmy. Przechowuje podstawowe informacje takie jak imię, nazwisko, adres e-mail oraz wynagrodzenie.

Konstruktor:

```
public Worker(String firstName, String lastName, String email)
```

Atrybuty:

- `firstName`: `String` - Imię pracownika.
- `lastName`: `String` - Nazwisko pracownika.
- `email`: `String` - Adres e-mail pracownika.
- `salary`: `double` - Wynagrodzenie pracownika.

Metody:

- `displayInfo()`: `void` - Wyświetla informacje o pracowniku.
- `changeInfo(String newFirstName, String newLastName, String newEmail)`: `void` - Zmienia informacje o pracowniku.
- `setSalary(double newSalary)`: `void` - Ustanawia wynagrodzenie pracownika.
- `increaseSalary(double amount)`: `void` - Zwiększa wynagrodzenie pracownika.

`ProjectManager`

Klasa `ProjectManager` reprezentuje kierownika projektu. Klasa dziedziczy po klasie `Worker`, jednak przechowuje dodatkowo informacje o projektach, w które zaangażowany jest kierownik projektu.

Konstruktor:

```
public ProjectManager(String firstName, String lastName, String email)
```

Atrybuty:

- *Odziedziczone po klasie* `Worker`
- `projects`: `ArrayList<Project>` - Lista projektów, którymi aktualnie zarządza.

Metody:

- *Odziedziczone po klasie* `Worker`
- `showProjects()`: `void` - Wyświetla listę projektów, którymi aktualnie zarządza kierownik.

- `addProject(Project project)`: `boolean` - Dodaje nowy projekt do listy projektów kierownika, pod warunkiem że nie zarządza 3 projektami jednocześnie.
- `removeProject(Project project)`: `void` - Usuwa projekt z listy projektów kierownika.
- `showReport(TeamMember teamMember)`: `void` - Wyświetla raport złożony przez członka zespołu pracującego nad projektem.

TeamMember

Klasa `TeamMember` reprezentuje członka zespołu pracującego nad projektem. Klasa dziedziczy po klasie `Worker`, jednak przechowuje dodatkowe informacje takie jak unikalny indeks członka zespołu, stopień uprawnień, zespół, do którego należy, sporządzony raport, lista przydzielonych zadań oraz data dołączenia do zespołu.

Obiekty tej klasy powstają automatycznie podczas wywołania metody `addMember(Worker newMember, int permissionStatus)` na obiekcie klasy `Team`.

Konstruktor:

```
public TeamMember(String firstName, String lastName, String email,
                  int permissionStatus)
```

Atrybuty:

- *Odziedziczone po klasie `Worker`*
- `index`: `int` - Unikalny indeks, przydzielany podczas dodawania pracownika do zespołu za pomocą metody `addMember(Worker newMember, int permissionStatus)` wywołanej na obiekcie klasy `Team`.
- `permissionStatus`: `int` - Stopień uprawnień, jaki posiada dany członek zespołu. Określa on, które zadania mogą zostać mu przydzielone.
- `memberOf`: `Team` - Informacja, do którego zespołu należy pracownik. Jest automatycznie aktualizowana podczas dodawania pracownika do zespołu.
- `report`: `String` - Raport sporządzany przez członka zespołu.
- `memberTasks`: `ArrayList<Task>` - Lista zadań przydzielonych członkowi zespołu.
- `memberSince`: `LocalDate` - Data dołączenia pracownika do zespołu.

Metody:

- *Odziedziczone po klasie `Worker`*
- `submitReport(String weeklyReport)`: `void` - Umożliwia pracownikowi spisanie swojej dotychczasowej pracy w dniu określonym przez harmonogram. Sporządzony raport jest dostępny do wglądu kierownikowi projektu.
- `showTasks()`: `void` - Wyświetla listę zadań przydzielonych pracownikowi.
- `isAllowed(int requiredPermissionStatus)`: `boolean` - Informuje, czy dany członek zespołu ma odpowiednie uprawnienia do wykonania danego zadania.
- `addToCommonFile(String idea)`: `void` - Umożliwia pracownikowi dodanie idei, którą chciałby się podzielić z zespołem, do wspólnego pliku dzielonego przez zespół.

Team

Klasa `Team` reprezentuje zespół, który pracuje nad danym projektem. Klasa przechowuje informacje takie jak *ksywka* zespołu, lista pracowników, którzy do niego należą, harmonogram zespołu, plik zespołu, do którego dostęp mają wszyscy członkowie oraz licznik członków zespołu, który umożliwia nadawanie im unikalnych indeksów.

Obiekt tej klasy powstaje automatycznie podczas inicjalizacji obiektu klasy `Project`.

Atrybuty:

- `nickname`: `String` - *Ksywka* nadana zespołowi.
- `members`: `ArrayList<TeamMember>` - Lista członków zespołu.
- `schedule`: `Schedule` - Harmonogram zespołu.
- `commonFile`: `File` - Plik umożliwiający członkom zespołu dzielenie się ideami.
- `memberIndex`: `int` - Licznik klasy umożliwiający nadawanie unikalnych indeksów.

Metody:

- `displayInfo()`: `void` - Wyświetla informacje o zespole.
- `setNickname(String nickname)`: `void` - Ustanawia nazwę zespołu.
- `setCommonFile(File commonFile)`: `void` - Przypisuje zespołowi jego wspólny plik.
- `addMember(Worker newMember, int permissionStatus)`: `void` - Dodaje nowego pracownika do zespołu wraz z jego uprawnieniami.
- `removeMember(TeamMember member)`: `void` - Usuwa pracownika z zespołu.
- `getMemberByIndex(int index)`: `TeamMember` - Zwraca członka zespołu na podstawie przekazanego indeksu.

Meeting

Klasa `Meeting` reprezentuje pojedyncze spotkanie. Przechowuje informacje takie jak nazwa, miejsce, czas oraz przewidywana długość trwania spotkania.

Konstruktor:

```
public Meeting(String title, String place, LocalDateTime time, double duration)
```

Atrybuty:

- `title`: `String` - Nazwa spotkania.
- `place`: `String` - Miejsce spotkania.
- `time`: `LocalDateTime` - Czas (data i godzina) spotkania.
- `duration`: `double` - Przewidywany czas spotkania podawany w godzinach.

Metody:

- `displayInfo()`: `void` - Wyświetla informacje o spotkaniu.

Schedule

Klasa `Schedule` reprezentuje harmonogram przypisany zespołowi. Przechowuje informacje takie jak dzień składania raportów przez pracowników oraz lista nadchodzących spotkań.

Obiekt tej klasy powstaje automatycznie podczas inicjalizacji obiektu klasy `Team`.

Atrybuty:

- `reportDay`: `DayOfWeek` - Dzień tygodnia przeznaczony na składanie raportów.
- `meetings`: `ArrayList<Meeting>` - Lista nadchodzących spotkań.

Metody:

- `setReportDay(DayOfWeek day)`: `void` - Ustanawia dzień tygodnia, który będzie przeznaczony na pisanie raportów.
- `showMeetings()`: `void` - Wyświetla spotkania zapisane w harmonogramie.
- `addMeeting(Meeting meeting)`: `void` - Dodaje spotkanie do harmonogramu.
- `removeMeeting(Meeting meeting)`: `void` - Usuwa spotkanie z harmonogramu.

Task

Klasa `Task` reprezentuje pojedyncze zadanie do wykonania w ramach projektu. Przechowuje informacje takie jak nazwa oraz opis zadania, wymagany poziom uprawnień, termin wykonania, szacowany koszt oraz status zadania.

Obiekty tej klasy są przypisywane obiektom klasy `TeamMember` jako zadania do wykonania. Obiektami tej klasy zarządza obiekt klasy `TaskManager`.

Konstruktor:

```
public Task(String name, int requiredPermissionStatus, LocalDate deadline,
                double estimatedCost)
```

Atrybuty:

- `name`: `String` - Nazwa zadania.
- `description`: `String` - Opis zadania.
- `requiredPermissionStatus`: `int` - Wymagany poziom uprawnień, który musi mieć członek zespołu, aby mógł podjąć się danego zadania.
- `deadline`: `LocalDate` - Termin wykonania zadania.
- `estimatedCost`: `double` - Szacowany koszt wykonania zadania.
- `isCompleted`: `boolean` - Informacja o tym, czy zadanie zostało wykonane.

Metody:

- `setDescription(String description)`: `void` - Ustanawia opis zadania.
- `showDescription()`: `void` - Wyświetla opis zadania.

- `updateEstimatedCost(double newEstimatedCost)`: `void` - Aktualizuje szacowane koszty wykonania zadania.

TaskManager

Klasa `TaskManager` reprezentuje menedżera do spraw zadań. Przechowuje informacje takie jak projekt, do którego jest przypisany, lista zadań pozostałych do wykonania oraz lista zadań już wykonanych.

Obiekt tej klasy powstaje automatycznie podczas inicjalizacji obiektu klasy `Project`.

Atrybuty:

- `project`: `Project` - Projekt, w ramach którego postał obiekt `TaskManager`.
- `tasks`: `ArrayList<Task>` - Lista zadań pozostałych do wykonania.
- `completedTasks`: `ArrayList<Task>` - Lista wykonanych zadań.

Metody:

- `assignTask(Task task, TeamMember member)`: `void` - Przypisuje dane zadanie konkretnemu członkowi zespołu.
- `setAsCompleted(Task task, TeamMember member)`: `void` - Zmienia status zadania na *wykonane*. Usuwa je z listy zadań do wykonania pracownika oraz przenosi je do listy wykonanych zadań. Aktualizuje budżet projektu, zmniejszając go o koszt wykonania zadania.
- `moveTask(Task task, TeamMember removeFrom, TeamMember moveTo)`: `void` - Przepisuje zadanie do wykonania z jednego członka zespołu na drugiego członka zespołu.
- `changeTaskDeadline(Task task, LocalDate newDeadline)`: `void` - Zmienia termin wykonania zadania.
- `addTask(Task task)`: `void` - Dodaje zadanie do listy zadań do wykonania.
- `removeTask(Task task)`: `void` - Usuwa zadanie z listy zadań do wykonania.
- `generateReport()`: `void` - Generuje raport na podstawie wykonanych oraz pozostałych zadań. Wyświetla, jaka część projektu (procentowo) została zrealizowana.

Project

Klasa `Project` reprezentuje projekt, który ma zostać wykonany. Przechowuje informacje takie jak nazwa oraz opis projektu, termin wykonania, kierownik projektu, zespół pracujący nad projektem, menedżer do spraw zadań, sponsor oraz przeznaczony budżet.

Podczas inicjalizacji obiektu tej klasy, zostaje mu przypisany obiekt klasy `Team` oraz obiekt klasy `TaskManager`.

Konstruktor:

```
public Project(String name, String description, LocalDate deadline,
               String sponsor, double budget)
```

Atrybuty:

- `name`: `String` - Nazwa projektu.
- `description`: `String` - Szczegółowy opis projektu.
- `deadline`: `LocalDate` - Termin wykonania projektu.
- `projectManager`: `ProjectManager` - Przydzielony kierownik projektu.
- `team`: `Team` - Zespół przydzielony do prac nad projektem.
- `taskManager`: `TaskManager` - Menedżer zadań przydzielony do zarządzania zadaniami.
- `sponsor`: `String` - Sponsor projektu.
- `budget`: `double` - Budżet przeznaczony na projekt.

Metody:

- `setProjectManager(ProjectManager manager)`: `void` - Przydziela projektowi własnego kierownika projektu.
- `displayInfo()`: `void` - Wyświetla podstawowe informacje o projekcie.
- `updateBudget(double newBudget)`: `void` - Aktualizuje budżet przeznaczony na projekt.
- `checkBudget()`: `void` - Monitoruje budżet. Wyświetla wykorzystaną kwotę, kwotę potrzebną do zrealizowania jeszcze niewykonanych zadań oraz pozostały budżet lub informację o przekroczeniu budżetu.
- `timeLeft()`: `void` - Wyświetla, ile czasu (lat, miesięcy i dni) pozostało do zrealizowania projektu.