**課題管理プログラム**

チーム名:フル単

35714012　井田　礼慈

35714026　大橋　蒼一朗

35714128　松岡　遼

1. 自由課題の概要

本課題では、チームで協力して「課題管理プログラム」を開発しました。このプログ

ラムは、以下の5項目のデータを管理します。

* 課題 ID
* 課題名
* 出題した授業名
* 締め切り
* 課題が完了したかどうか

1. 各ファイルの概要

|  |  |
| --- | --- |
| ファイル名 | 概要 |
| Task.java | 課題データを表すデータモデル |
| DBUtil.java | データベース接続用のユーティリティクラス |
| TaskDAO.java | タスクに関するデータ操作の契約を定義 |
| TaskDAOImpl.java | “TaskDAO.java”インターフェースの実装クラス．JDBCを用いてSQLiteデータベースとやり取りをする． |
| Main.java | アプリケーションのエントリーポイント．主な動作内容は項目参照． |
| Command.java | 以下”AddTask.java”～”Exit.java”のコマンドパターンのみを実装したインターフェース |
| AddTask.java | Commandインターフェースを実装する具体的なコマンドクラス．「タスクの追加」を表現． |
| DeleteTask.java | Commandインターフェースを実装する具体的なコマンドクラス．「タスクの削除」を表現． |
| CompleteTask.java | Commandインターフェースを実装する具体的なコマンドクラス．「タスク完了」を表現． |
| ShowAll.java | Commandインターフェースを実装する具体的なコマンドクラス．「全タスクの表示」を表現． |
| SortTasks.java | Commandインターフェースを実装する具体的なコマンドクラス．「タスクのソート」を表現． |
| Exit.java | Commandインターフェースを実装する具体的なコマンドクラス．「終了」を表現． |
| SortStrategy.java | 以下”TitleSortStrategy.java”～”TaskSorter.java”ソート戦略のインターフェース．List<Task>を受け取り、ソートされたリストを返す． |
| TitleSortStrategy.java | 課題名(タイトル)でソート．ひらがな，カタカナ→漢字の順にソート． |
| DeadlineSortStrategy.java | 締切日でソート |
| CompletedSortStrategy.java | 完了状態でソート |
| SubjectSortStrategy.java | 授業名(科目)でソート．ひらがな，カタカナ→漢字の順にソート． |
| TaskSorter.java | ソート戦略を動的に切り替えてソートを実行．Strategyパターンの「コンテキスト」に相当． |

1. 各ファイルの詳細説明

* Task.java

構成内容

* フィールド
* id:タスクの識別子
* title:タスクのタイトル
* subject:タスクに関連する授業名(科目名)
* deadline:タスクの締切日
* completed:タスクが完了しているかどうか(boolean)
* その他各種ゲッタ・セッタ
* DBUtil.java

構成内容

* メソッド
* setUrl: 接続先URLを指定するためのメソッド. デフォルトは"jdbc:sqlite:tasks.db"で、テスト時に切り替える.
* getConnection:SQLiteデータベースへの接続を取得するための静的メソッド．SQLExceptionをスローする．JDBCのDriverManagerを使用．
* TaskDAO.java

構成内容

* メソッド
* findAll:全タスクを取得するメソッド．戻り値はTaskのリスト．
* findAt:引数で指定したIDのタスクを取得するメソッド. 戻り値はTask.
* addTask:新たなタスクを追加するメソッド．引数にTaskオブジェクトを取る．
* markTaskCompleted:指定されたIDのタスク(completed)を完了状態に更新するメソッド
* deleteTask:指定されたIDのタスクを削除するメソッド．
* close: 接続をクローズするメソッド.
* TaskDAOImpl.java

構成内容

* コンストラクタ:データベースに接続し，Taskテーブルが存在しない場合は新たに作成する．
* findAll:tasksテーブルからすべてのレコードを取得し，Taskオブジェクトのリストとして返す
* findAt: 指定したIDのタスクを取得する. 内部でexecuteQueryを呼び出し、findAllとクエリ実行と結果取得のロジックを共通化
* addTask:新たなタスクをtasksテーブルに挿入．
* markTaskCompleted:指定されたIDのタスク(completed)をtrueに更新
* deleteTask:指定されたIDのタスクをtasksテーブルから削除
* close: フィールドとして持っているConnectionを閉じる.
* Main.java

処理内容

* “TaskDAO.java”の実装クラスを用いてタスクを追加
* “findAll”メソッドで全タスクを取得し、各タスクの情報を標準出力に表示
* “markTaskCompleted”メソッドで指定されたIDのタスクを完了状態に更新
* “deleteTask”で指定されたIDのタスクを削除

1. コンパイル・実行方法

・コンパイル: javac -cp .:sqlite-jdbc-3.30.1.jar:junit-4.13.2.jar:hamcrest-core-1.3.jar [filename]

・テスト実行: java -cp .:sqlite-jdbc-3.30.1.jar:junit-4.13.2.jar:hamcrest-core-1.3.jar org.junit.runner.JUnitCore [testname]

・Main実行: java -cp .:sqlite-jdbc-3.30.1.jar:junit-4.13.2.jar:hamcrest-core-1.3.jar Main

1. 制限

* データベース

SQLiteを用いて”tasks.db”に、課題データを永続的に保存・管理しています。DBUtil.javaでデータベース接続を行い、TaskDAOImpl.javaでCRUD操作を実装しました。

* Junitテストコード
  1. TaskTest.java

Task.javaのコンストラクタ、アクセッサの動作のテストを行う。

* 1. TaskDAOImplTest.java

SQLiteのインメモリDB を使用し、TaskDAOImpl.javaの各メソッドの単体テストを行う。

* 1. TaskSorterTest.java  
     各種ソートの切り替えのテストを行う。
  2. TitleSortStrategyTest.java  
     SubjectSortStrategyTest.java

DeadlineSortStrategyTest.java

CompletedSortStrategyTest.java  
各種ソートの動作のテストを行う。

* GoFデザインパターン

使用パターン:Strategyパターン

課題のソート方法を柔軟に変更できるように、Strategyパターンを採用しました。

・SortStrategyインターフェースを定義

* 1. 以下の具体的な戦略クラスを実装
  + TitleSortStrategy
  + DeadlineSortStrategy
  + CompletedSortStrategy
  + SubjectSortStrategy

TaskSorterクラスで戦略を切り替え可能にし、ユーザーの選択に応じてソートを実行

* Java17

Sealed Classes(密封クラス)

* Commandインターフェースをsealedとして定義し、許可されたクラスのみが実装できるように制限。
* これにより、コマンドの種類を明確に制御し、安全性と可読性を向上。

Pattern Matching for switch(switch文のパターンマッチ）

* TaskDAOImpl.javaにて、Commandの種類に応じた処理をswitch文で分岐。
* Java17のパターンマッチング構文を活用し、より簡潔で安全なコードを実現。