小テスト 00

1. MiniTest\_00/Façade/Façade.slnを起動する。このプロジェクトにはSchoolClassクラスとTestResultクラスがある。Program.csのMain関数ではクラスの情報とテストの結果をロードしている。そしてテストの結果とクラスの情報を結合して、新しいcsvファイルを作成している。これらの処理をFacadeパターンを使用したTestIRというクラスを作成してCombineAndSaveという関数を呼ぶだけで完了するようにリファクタリングを行いなさい。また、CombineAndSave関数は下記の仕様で実装しなさい。

//

//テストの結果とクラス情報を結合したcsvファイルを出力する。

//classNameはクラスを表す文字。例えばAクラスであれば’A’という文字が渡される。

void CombineAndSave(char className)

Main関数はTestIRのインスタンスのCombine関数を使用するように実装を変更しなさい。

1. Adapter/Adapter.slnを起動する。このプロジェクトはマルチプラットフォームの3Dモデルの読み込みや表示を行うための疑似コードが記述されている。

このプロジェクトではモデルのインターフェースのIModelを継承したDirectX用のModelDXクラスとOpenGL用のModelGLクラスがある。

ある日、プロジェクト進行中にPlayStation4でもこのゲームを発売することが決定した。PlayStation4はソニーからモデルの読み込みや表示処理の機能を提供するsnModelPS4というクラスが提供されている。snModelPS4はソニーが提供しているモデル表示クラスとなっており、あなたは変更することはできないものとする。Adapterパターンを活用して、進行中のプロジェクトのモデル関連の処理と、PlayStation4のモデル関連の処理とうまく結合できるように設計と実装を行いなさい。

1. Strategy/Strategy.slnを起動する。このプロジェクトはマルチプラットフォームの3Dモデルの読み込みや表示を行うための疑似コードが記述されている。

プラットフォームごとにモデルの処理の分岐はif文やswitch文が多用される実装となっている。このモデル関連の処理をStrategyパターンを活用してリファクタリングを行いなさい。

1. State/State.slnを起動する。このプロジェクトはゲームのモンスターのAIに関する疑似コードが記述されている。モンスターのAIにはStateパターンを活用したFSMが採用されている。このプロジェクトを改造して、モンスターに攻撃状態、走り状態を追加して、一定の確率で状態が切り替わるようにしなさい。