プログラミング応用レポート

35714123: 藤 本 皇 汰 ,35719001: 秋 田 悠 希 ,35719003: 天 野 歩 夢

2025 年 5 月 26 日

1　はじめに

本レポートは、2025 年度名古屋工業大学プログラミング応用の課題レポートである。本レポートでは、我々のチーム「ムガルパレス名工大店」が開発した「WizardSpellNameQuiz」の仕様の解説を行う。本レポートは、以下の構成で進める。

- 1 章: はじめに
- 2 章: プログラムの概要
- 3 章: 解説
- 4 章: おわりに

本プロダクトは、ハリーポッター api[api の github のリンク] を使用しており、ハリーポッターに登場する魔法の名前を当てるクイズゲームである。

2　プログラムの概要

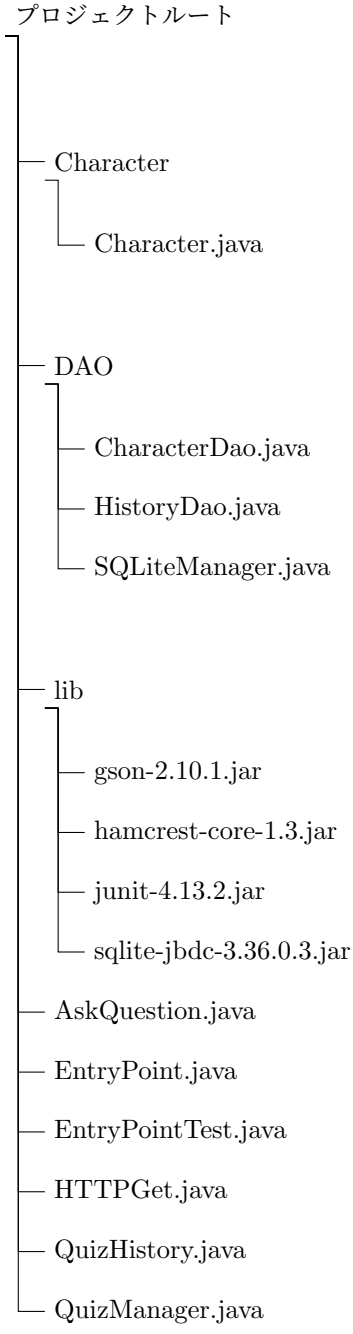
今回のプログラムは、課題の指示に従い作成されている。課題の指示は以下の通りである。

- データベースを使うこと
- Junit を用いたテストコードを作成すること
- GoF のデザインパターンを 1 つ以上使うこと
- Java 17 で新たに採用された言語仕様を調べ、それを用いたプログラムとすること

これらの指示に従い、我々は以下のようなプログラムを作成した。

- データベースを使うこと
  - SQLite を使用して、ハリーポッター api から取得したデータの id、魔法の名前及びその説明をデータベースに保存した。
  - データベースは、SQLite の JDBC ドライバを使用して接続した。
- Junit を用いたテストコードを作成すること
  - Junit を使用して、プログラムの各機能のテストコードを作成した。
  - テストコードは、プログラムの各機能が正しく動作することを確認するために使用した。
- GoF のデザインパターンを 1 つ以上使うこと
  - GoF のデザインパターンとして、Strategy パターンを使用した。
  - Strategy パターンを用いて出題形式の変換を行う機能を実装した。
- Java 17 で新たに採用された言語仕様を調べ、それを用いたプログラムとすること
  - Java 17 で新たに採用された言語仕様として、Pattern Matching for switch を使用した。
  - Pattern Matching for switch を使用して、ユーザーのモード選択の例外処理を行った。

今回のプロダクトのアーキテクチャは、以下のようになっている。



また、今後の解説に使用するためプログラムの実行結果を以下に示す。

```
こんにちは、ムガルパレス名工大店へようこそ！
これはハリーポッターのクイズゲームです。

オプションを選択してください:
[1] クイズに挑戦する
[3] 履歴を表示する
[4] 履歴をクリアする
[5] 終了
番号を入力してください: 1
説明: Bogey Hex - Turns the target's boogers into bats
1: Bat
2: Levicorpus
3: Ferula
4: Incendio
正しい名前の番号を選んでください（1-4）: 2
不正解。正解は 1: Bat です。

オプションを選択してください:
[1] クイズに挑戦する
[2] モードを変更する
[3] 履歴を表示する
[4] 履歴をクリアする
[5] 終了
番号を入力してください: 2

--- クイズモードの変更 ---
現在のモード: 4択モード
新しいモードを選択してください:
[0] 4択問題モード
[1] 2択問題モード
番号を入力してください: 1
モードを 2択問題モード に変更しました。

オプションを選択してください:
[1] クイズに挑戦する
[2] モードを変更する
[3] 履歴を表示する
[4] 履歴をクリアする
[5] 終了
番号を入力してください: 1
説明: One of three Unforgivable Curses, it causes unbearable pain in the target
名前: Crucio
この説明と名前の組み合わせは正しいですか？（○: y / ×: n）: y
正解！

オプションを選択してください:
```

```
[1] クイズに挑戦する
[2] モードを変更する
[3] 履歴を表示する
[4] 履歴をクリアする
[5] 終了
番号を入力してください: 3
履歴を表示します。

--- クイズの履歴 ---
[2025-05-26 05:53:37] 問題: One of three Unforgivable Curses, it causes unbearable pain in
     the target | 表示名: Crucio | 正解: Crucio | 回答: y | 正誤: ○ | モード: ×
    ○
[2025-05-26 05:53:22] 問題: Bogey Hex - Turns the target's boogers into bats | 表示名:
    Levicorpus | 正解: Bat | 回答: 2 | 正誤: ×
     | モード: 4択

オプションを選択してください:
[1] クイズに挑戦する
[2] モードを変更する
[3] 履歴を表示する
[4] 履歴をクリアする
[5] 終了
番号を入力してください: 4
履歴をクリアしました。

オプションを選択してください:
[1] クイズに挑戦する
[2] モードを変更する
[3] 履歴を表示する
[4] 履歴をクリアする
[5] 終了
番号を入力してください: 3
履歴を表示します。

--- クイズの履歴 ---
履歴はまだありません。

オプションを選択してください:
[1] クイズに挑戦する
[2] モードを変更する
[3] 履歴を表示する
[4] 履歴をクリアする
[5] 終了
番号を入力してください: 5
ゲームを終了します。ご利用ありがとうございました！
```

3 解説

ここからは、実行結果に基づいて解説を行う。
まず、以下の部分の解説を行う。

```
こんにちは、ムガルパレス名工大店へようこそ！
これはハリーポッターのクイズゲームです。
オプションを選択してください:
[1] クイズに挑戦する
[2] モードを変更する
[3] 履歴を表示する
[4] 履歴をクリアする
[5] 終了
番号を入力してください: 1
説明: Bogey Hex - Turns the target's boogers into bats
1: Bat
2: Levicorpus
3: Ferula
4: Incendio
正しい名前の番号を選んでください（1-4）: 2
不正解。正解は 1: Bat です。
```

この部分は、クイズの出題と回答の部分である。ユーザは、1 を入力することでクイズを開始することができる。クイズでは、ハリーポッターに登場する魔法の説明が表示され、その説明に対応する魔法の名前を選択肢から選ぶ形式である。選択肢は、1 から 4 までの番号で表示され、ユーザはその中から正しい番号を選ぶ必要がある。不正解の場合は、正解の番号とその魔法の名前が表示される。

実際にこの部分のロジックを解説を行う。まず、HTTP.Java にある「fetchAndSaveSpells」という関数を用いてハリーポッター api から情報を取得して、事前に構築した Character.java のクラスに保存する。その後、保存した情報を jbdc を用いて datas テーブルに保存する。ここまでの処理は、実際にプロダクトを動かす前に行う必要がある。最初にプロダクトを起動する際には EntryPoint.java の「main」関数を実行する。その後、QuizManager.java の「startQuiz」関数を実行することでクイズが開始される。そして、関数の実行により AskQuestion.java の「start」関数が呼び出される。それにより、クイズの出題が行われる。

実際にクイズの出題のロジックは、2 択の場合はデータベースからランダムにデータを取得して、その説明と名前のペアを保存しておく、その後、ユーザに説明と名前を表示し、それらが一致するかどうかを出題する。これらの情報は、事前に作成した QuizHistory.java の History クラスに保存後、History テーブルに保存される。 4 択の場合は、データベースからランダムに 4 つのデータを取得して、そのうち 1 つを正解として、残りの 3 つを不正解として選択肢を作成する。

次に、以下の部分の解説を行う。

```
オプションを選択してください:
[1] クイズに挑戦する
```

```
[2] モードを変更する
[3] 履歴を表示する
[4] 履歴をクリアする
[5] 終了
番号を入力してください: 2
--- クイズモードの変更 ---
現在のモード: 4択モード
新しいモードを選択してください:
[0] 4択問題モード
[1] 2択問題モード
番号を入力してください: 1
モードを 2択問題モード に変更しました。
```

この部分は、クイズのモードを変更する部分である。この部分では課題の要件である GoF のデザインパターンを使用している。実際に使用しているデザインパターンは、4 択の出題形式と 2 択の出題形式という複数のアルゴリズムを切り替える、Strategy パターンを使用した。

具体的には、EntryPoint.java のにより QuizManager.java の「changeMode」関数が呼び出される。その後、QuizManager.java の「changeMode」関数により AskQuestion.java に使用される変数を変更することで出題形式を変更することができる。

次に、以下の部分の解説を行う。

```
  オプションを選択してください:
[1] クイズに挑戦する
[2] モードを変更する
[3] 履歴を表示する
[4] 履歴をクリアする
[5] 終了
番号を入力してください: 1
説明: One of three Unforgivable Curses, it causes unbearable pain in the target
名前: Crucio
この説明と名前の組み合わせは正しいですか？（○: y / ×: n）: y
正解！
```

この部分は、2 択の出題形式でのクイズの出題と回答の部分である。先ほどの解説で説明したため、ここでは詳細な解説は省略する。

次に、以下の部分の解説を行う。

```
オプションを選択してください:
[1] クイズに挑戦する
[2] モードを変更する
[3] 履歴を表示する
[4] 履歴をクリアする
[5] 終了
```

```
番号を入力してください: 3
履歴を表示します。
--- クイズの履歴 ---
[2025-05-26 05:53:37] 問題: One of three Unforgivable Curses, it causes unbearable pain in
     the target | 表示名: Crucio | 正解: Crucio | 回答: y | 正誤: ○ | モード: ×
    ○
[2025-05-26 05:53:22] 問題: Bogey Hex - Turns the target's boogers into bats | 表示名:
    Levicorpus | 正解: Bat | 回答: 2 | 正誤: ×
     | モード: 4択
```

この部分は、クイズの履歴を表示する部分である。

QuizManager.java の「showHistory」関数が呼び出されることで、QuizHistory.java の「getHistory」関数が呼び出され、出題の際に History クラスに保存されていた、出題の説明、表示名、正解、回答、正誤、モードを取得する。その後、取得した履歴を表示する。

次に、以下の部分の解説を行う。

```
オプションを選択してください:
[1] クイズに挑戦する
[2] モードを変更する
[3] 履歴を表示する
[4] 履歴をクリアする
[5] 終了
番号を入力してください: 4
履歴をクリアしました。
```

この部分は、クイズの履歴をクリアする部分である。

QuizManager.java の「clearHistory」関数が呼び出されることで、QuizHistory.java の「clearHistory」関数が呼び出され、History テーブルのデータを全て削除する。

最後に、以下の部分の解説を行う。

```
オプションを選択してください:
[1] クイズに挑戦する
[2] モードを変更する
[3] 履歴を表示する
[4] 履歴をクリアする
[5] 終了
番号を入力してください: 5
ゲームを終了します。ご利用ありがとうございました！
```

この部分は、プログラムを終了する部分である。

EntryPoint.java の Switch 文により、ユーザが 5 を入力した場合はプログラムを終了する。

以上が、実行結果に基づく解説である。また、実行結果では示していないが、EntryPoint.java の起動後に

4 を入力すると、History テーブルのデータを全て削除することができる。

今回のプロダクトは、ハリーポッター api を使用しており、ハリーポッターに登場する魔法の名前を当てるクイズゲームである。

プログラムの実行結果を見てわかるように、クイズの出題形式を 4 択と 2 択で切り替えることができる。

また、クイズの履歴を表示することができる。

今回のプロダクトは、Java 17 で新たに採用された Pattern Matching for switch を使用しており、それを使用している部分のコードを以下に示す。

```
String text = scanner.nextLine();

      Object obj = null;

      if (isInteger(text)) {
        obj = Integer.parseInt(text);
      } else if (isDouble(text)) {
          obj = Double.parseDouble(text);
      } else if (isBoolean(text)) {
          obj = Boolean.parseBoolean(text);
      } else {
          obj = text; // 文字列として扱う
      }
      switch (obj) {
        case Integer i:
        switch (i) {
          case 1:
            QuizManager.startQuiz(scanner);
            break;
          case 2:
            QuizManager.changeMode(scanner);
            break;
          case 3:
            System.out.println("履歴を表示します。");
            QuizHistory.showHistory();
            break;
          case 4:
            QuizHistory.clearHistory();
            break;// または break + while 条件に false
          case 5:
            System.out.println("ゲームを終了します。ご利用ありがとうございました！");
            scanner.close();
            return; // または break + while 条件に false
          default:
            System.out.println("無効な入力です。1から5を選んでください。");
        }
          break;

        case String s:
```

```
40            System.out.println("無効な入力です。数字を入力してください。");
41            break;
42          default:
43            break;
44        }
```

このコードでは、Pattern Matching for switch を使用して、Object 型の変数 obj を Integer 型にキャストしている。これにより、switch 文の case で Integer 型の変数 i を使用することができる。

また、JUnit テストコードの作成も行っており、EntryPointTest.java にテストコードを記述している。

```
1  import static org.junit.Assert.*;
2  import org.junit.Test;
3  import java.io.*;
4
5  public class EntryPointTest {
6      @Test
7      public void testMainPrintsWelcomeMessage() throws Exception {
8          InputStream originalIn = System.in;
9          PrintStream originalOut = System.out;
10         ByteArrayInputStream in = new ByteArrayInputStream("5\n".getBytes()); // 5で即終了
11         ByteArrayOutputStream out = new ByteArrayOutputStream();
12         System.setIn(in);
13         System.setOut(new PrintStream(out));
14
15         EntryPoint.main(new String[]{});
16
17         String output = out.toString();
18         assertTrue(output.contains("こんにちは、ムガルパレス名工大店へようこそ！"));
19         assertTrue(output.contains("これはハリーポッターのクイズゲームです。"));
20         assertTrue(output.contains("ゲームを終了します。ご利用ありがとうございました！"));
21
22         System.setIn(originalIn);
23         System.setOut(originalOut);
24     }
25
26     @Test
27     public void testInvalidInputShowsErrorMessage() throws Exception {
28         InputStream originalIn = System.in;
29         PrintStream originalOut = System.out;
30         ByteArrayInputStream in = new ByteArrayInputStream("a\n5\n".getBytes());
31         ByteArrayOutputStream out = new ByteArrayOutputStream();
32         System.setIn(in);
33         System.setOut(new PrintStream(out));
34
35         EntryPoint.main(new String[]{});
36         String output = out.toString();
```

```
37          assertTrue(output.contains("数字を入力してください") || output.contains("無効な入
                力です"));
38          assertTrue(output.contains("ゲームを終了します。ご利用ありがとうございました！"));
39
40          System.setIn(originalIn);
41          System.setOut(originalOut);
42      }
43
44      @Test
45      public void testClearHistoryOption() throws Exception {
46          InputStream originalIn = System.in;
47          PrintStream originalOut = System.out;
48          ByteArrayInputStream in = new ByteArrayInputStream("4\n3\n5\n".getBytes());
49          ByteArrayOutputStream out = new ByteArrayOutputStream();
50          System.setIn(in);
51          System.setOut(new PrintStream(out));
52
53          EntryPoint.main(new String[]{});
54          String output = out.toString();
55          assertTrue(output.contains("履歴をクリアしました。"));
56          assertTrue(output.contains("履歴はまだありません。"));
57          assertTrue(output.contains("ゲームを終了します。ご利用ありがとうございました！"));
58
59          System.setIn(originalIn);
60          System.setOut(originalOut);
61      }
62
63      @Test
64      public void testShowHistoryAndAnswer() throws Exception {
65          InputStream originalIn = System.in;
66          PrintStream originalOut = System.out;
67          ByteArrayInputStream in = new ByteArrayInputStream("2\n0\n1\n1\n2\n1\n1\ny\n3\n5\n
                ".getBytes());
68          ByteArrayOutputStream out = new ByteArrayOutputStream();
69          System.setIn(in);
70          System.setOut(new PrintStream(out));
71
72          EntryPoint.main(new String[]{});
73          String output = out.toString();
74          assertTrue(output.contains("正しい名前の番号を選んでください（1-4）: "));
75          assertTrue(output.contains("正解！") || output.contains("不正解。正解は"));
76          assertTrue(output.contains("この説明と名前の組み合わせは正しいですか？（○: y / ×
                : n）: "));
77          assertTrue(output.contains("正解！") || output.contains("不正解。正解は"));
78          assertTrue(output.contains("履歴を表示します。"));
79          assertTrue(output.contains("--- クイズの履歴 ---"));
80          assertTrue(output.contains("ゲームを終了します。ご利用ありがとうございました！"));
81
82          System.setIn(originalIn);
```

```
83          System.setOut(originalOut);
84      }
85
86      @Test
87      public void testChangeModeOption() throws Exception {
88          InputStream originalIn = System.in;
89          PrintStream originalOut = System.out;
90          ByteArrayInputStream in = new ByteArrayInputStream("2\n0\n2\n1\n5\n".getBytes());
91          ByteArrayOutputStream out = new ByteArrayOutputStream();
92          System.setIn(in);
93          System.setOut(new PrintStream(out));
94
95          EntryPoint.main(new String[]{});
96          String output = out.toString();
97          assertTrue(output.contains("クイズモードの変更"));
98          assertTrue(output.contains("モードを 4択問題モード に変更しました。") || output.
                contains("4択問題モード"));
99          assertTrue(output.contains("モードを 2択問題モード に変更しました。") || output.
                contains("2択問題モード"));
100         assertTrue(output.contains("ゲームを終了します。ご利用ありがとうございました！"));
101
102         System.setIn(originalIn);
103         System.setOut(originalOut);
104     }
105
106
107 }
```

このテストコードでは、EntryPoint.java の「main」関数をテストしており、ユーザが入力した値に応じて正しい処理が行われることを確認している。テストは 5 つのケースに分かれており、各ケースで異なる入力値を与えて、期待される出力が得られるかを確認している。

具体的には、以下のようなテストを行っている。

- きちんとコードが実行され、Welcome メッセージが表示されることを確認する。そして 5 を入力した場合は、プログラムが終了することを確認する。
- ユーザが数字以外の無効な入力をした場合、適切なエラーメッセージが表示されることを確認する。
- ユーザが 4 を入力した場合、履歴がクリアされることを確認する。
- 1 を押すと問題が出され、解答を行い、正誤判定が行われ、履歴が保存され、ユーザが 3 を入力した場合、履歴が表示されることを確認する。
- ユーザが 2 を入力した場合、モードが変更されることを確認する。

4　おわりに

今回のプロダクトは、事前にメンバー内で仕事を細分化し、仕事担当領域を割り当てて開発を行った為、スムーズに開発を進めることができた。また、GitHub を使用してコードの管理を行い、メンバー間でのコミュニケーションを円滑にすることができた。きちんと準備をしたチーム開発は、ストレスが少なく制作が進んでいくので、今後もこのような開発スタイルを続けていきたい。なお、今回のプロダクトで各自が担当した領域を示す。

- 天野歩夢

  : API から情報を取得し、データベースに保存する部分の実装. 及び本レポートの作成

- 秋田悠希

  : データベースから得た情報を用いてクイズの出題と回答の部分の実装.JUnit テストコードの作成

- 藤本皇汰

  : エントリーポイントの作成、プロダクトの設計、開発方針の決定

追記

提出した圧縮フォルダには, 容量オーバーのため lib フォルダが含まれていません。lib フォルダには、以下のライブラリが含まれています。

- gson-2.10.1.jar
- hamcrest-core-1.3.jar
- junit-4.13.2.jar
- sqlite-jdbc-3.36.0.3.jar

これらのライブラリは、プロジェクトのビルドパスに追加する必要があります。実行する際には、これらのライブラリをプロジェクトの lib フォルダに配置し、ビルドパスに追加してください。なお、gson-2.10.1.jar 以外は、講義資料内で触れているものと同じバージョンのものを使用しています。gson-2.10.1.jar は、圧縮フォルダと一緒に提出しています。