情報学群実験第1　最終課題「マインスイーパー」
最終レポート

1240381 渡部奨矢
2021年8月3日

1.概要

このレポートでは情報学群実験第1の授業での最終課題である「マインスイーパー」の実装方法を示している。「マインスイーパー」の基本機能を私がどのように実装したのか、また追加機能で私がどんなものを追加しようと考え、それをどのように実装していったのかということを示している。レポート中には雛形にあったフィールド名、メソッド名の英単語や自作したメソッド名の英単語などが登場する。

2.実装

2−1.基本機能の実装

2−1−1.仕様1：開始時、盤面にランダムに地雷を設置

setMine() , initTable() に主に実装、showTable()を追加。

setMine() では乱数により地雷をどこに置くのか決定し、出てきた番地に-90という値をセットすることで地雷の設置を実現した。（地雷の値を-90としたのはデバックをしやすくするためである。）

initTable() ではまず setMine() を呼び出し、地雷をセット。その後地雷に隣接するマスにそれぞれ1を加算することによって、マインスイーパーの機能を実現した。

またこれは追加機能と言うまででもないのかもしれないが、showTable メソッドを追加した。これは神になれるメソッド（本当はただのデバック用メソッド）で、盤面がターミナル上に出力され、どこに地雷があるのかがかる。

2−1−2.仕様2：パネルを左クリック、クリックしたパネルを開く

openTile() に主に実装。

openTile() では table の値を文字列に変換し、それをクリックしたタイルにセットするという処理をインターフェースにある setTextToTile() などを使いながら実装した。またフラグが立っているところには実行できないように if 文を使い処理させた。

2−1−3.仕様3：開いたパネルに地雷があれば、すべてのパネルを開く

openTile() , openAllTiles()に主に実装。

openTile内にif文を作り、クリックされたポイントの table の値が負の場合、そこは地雷があるため、openAllTiles() を実行するように呼び出した。

openAllTiles() では for 文と if 文を使い、すべてのタイルに文字を代入することを実装した。負の値であれば地雷を意味する”Boo”の文字列を入れ、そうでなければ table の数値を文字列に変換し、それをタイルに入れるということをした。

2−1−4.仕様4：パネルを右クリック、フラグを立てる

setFlag() に主に実装した。

if 文を使い、すでにフラグをセット済なのか、そうではないのかによって処理を変えて実装した。

2−1−5.仕様**5**：ゲームクリア、ゲームオーバ時のダイアログ

openTile() 内で主に実装、winjudege() メソッドを追加。
　openTile()内で、勝敗判定を行ったあと、gui.lose(), gui.win()を呼び出すことによって実装。
　勝敗判定については次のように実装した。負け判定は先述したメソッド内ですべてのタイルを開くのと同じタイミングで実行し、勝利判定は winJudege()というメソッドを新たに定義し、オープンタイルをするたびに呼び出す。winJudege()ではすべてのタイルを見て、開かれていないマスをカウントし、カウントの値がちょうど地雷数となったら同じになったらプレイヤーの勝利というロジックを書くことで勝敗判定を行った。

2-2.追加機能の実装

2−1−1.追加機能**1**：ゼロチェック

zeroCheck()に主に実装。（追加メソッド）
　この機能はクリックしたタイルが 0 だったとき、隣接するタイルを自動でオープンするという機能である。実装のきっかけは、プレイ中にゼロの周りを全部開けるのがめんどくさかったため、追加機能として実装した。
　zeroCheck()では if 文を使いこの処理を実装した。

2−1−2.追加機能**2**：最初のクリックで地雷を踏まないための機能

openTile(), resetTable()に主に実装。
　この機能は最初に地雷を踏まないために、最初にクリックした場所がゼロでない間、resetTable()を繰り返し、盤面を作り直し続けるという処理をする機能です。
　この機能によって、ユーザーは最初は必ずゼロから始められるため、初手でゲームオーバーになることを防げるほか、ゼロをクリックすることにより隣接マスも開くという気持ち良いスタートを切ることができるようになっている。

2−1−3.追加機能**3.**：色の設定

setFlagColor() , setDefaultColor()などに実装。（追加メソッド）
　この機能は、見えにくさの改善やゲームの楽しさの向上のために追加された。まずはフラグが見にくかったので文字色を変えることで視認性を上げた。そしてゲームのクリア時、オーバー時に盤面全体の色を変化させることで楽しさの向上を図った。
　setFlagColor(), setLoseColor(), setDefaultColor() などの今回追加したメソッドは Main で実装し、MineSweeper のインターフェースにも記載した。それぞれ、Buttonクラスの背景色や文字色の設定をするメソッドによって実装した。
（参考文献1, 2, 3：ボタンクラスでの色の変更方法、Javaの色情報）

2−1−4.追加機能**4**：ゲットタイルラベル

getTileLabel()に主に実装。
　この機能は開発者側の改善なので、機能とは言えないかもしれないが工夫した記述でコードがスッキリしたので一応記載する。タイルの文字列を取得するコードをできるだけ無駄なく書くために、定義した。setLabelがあることからgetLabelも使えるのではないかと、リファレンスで検索し、使い方を考えて実装した。
（参考文献3：Javaリファレンス、Buttonクラス）

3.まとめ

今回の課題で私はマインスイーパーを作成したが、コードを書く、コードを動かすということを体に染み込ませる良い機会となったと思っている。仕様があるなかでの基本機能の実装、仕様のない追加機能の実装ともに、とても楽しかった。なぜなら、これまで授業で習ってきたJavaを、ゲームを作るという面白い題材でアウトプットする機会が与えられたからである。まずはこの課題にとても感謝している。

実装の中での苦労も多少はあった。特に実装していく上で大体は動いているが、細かいところで微妙に自分の意図したとおりに動いてくれいということが頻発した。そして、実装が終わり、このレポートを書いている今この段階でも、もしかしたらプログラムコードには微妙に意図していない動作をするものがあるかもしれないという心配があることを自白しておく。

ただ、全体としてはその苦労よりも楽しさであったり、コードを書くということが染み付く感覚のほうが強かった。正直、これまでの授業の中では、このように（ゲームという）自分たちとの距離が近い課題は少なく、また毎回の課題は授業を受けた直後での実装ということもあり、本当にコードを書くことが自分自身に身についているか少し不安に感じることもあった。しかし今回の課題を1人で実装しているときにしっかりと自分の意図で実装できている感覚があったため、それはとても嬉しかったし、授業でやっていたことも自分なりにしっかり身につけられていたんだと感じた。

最後に、この授業とこの課題に感謝しています。ありがとうございました。

4.参考文献

参考文献 1：TATSUO IKURA,「Let's プログラミング」
https://www.javadrive.jp/tutorial/jbutton/index2.html
閲覧日2021年7月28日

参考文献 2：stealthjong,「stack overflow」
https://stackoverflow.com/questions/1358398/how-to-get-jbutton-default-background-color
閲覧日2021年7月28日

参考文献 3：Oracle and/or its affiliates. All rights reserved. ,「APIリファレンス」
https://docs.oracle.com/javase/jp/7/api/java/awt/Color.html
閲覧日2021年7月28日

参考文献 4：Oracle and/or its affiliates. All rights reserved. ,「APIリファレンス」
https://docs.oracle.com/javase/jp/8/docs/api/java/awt/Button.html
閲覧日2021年7月28日