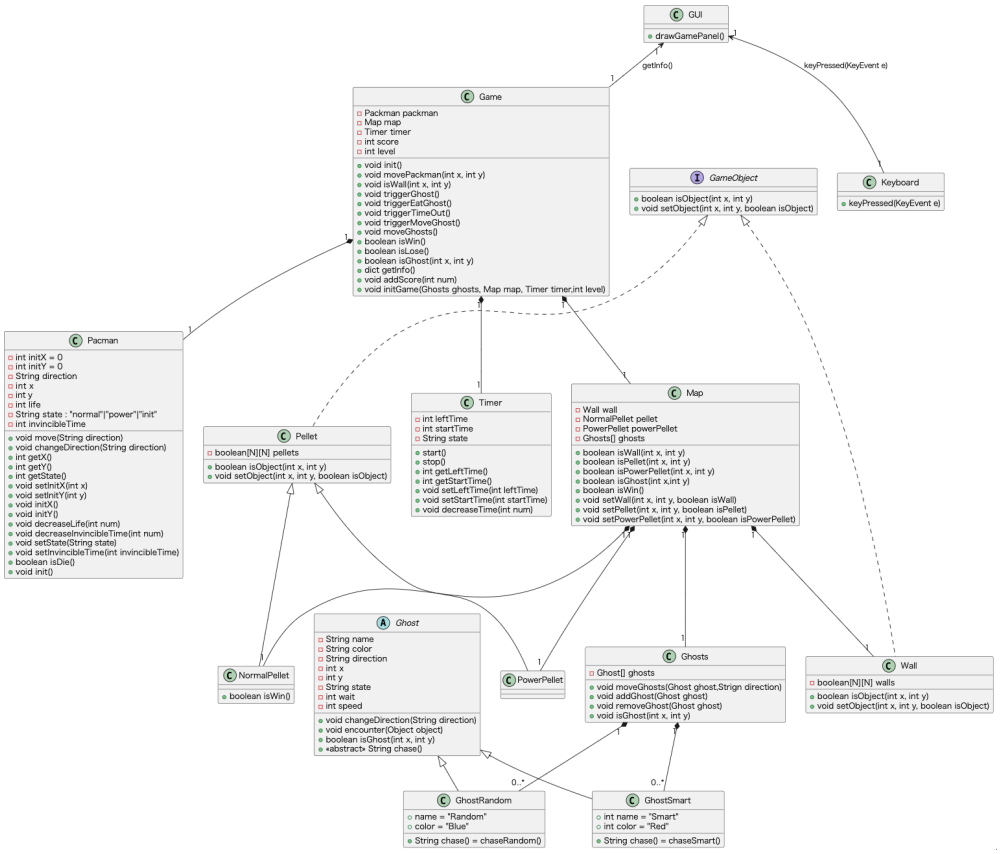
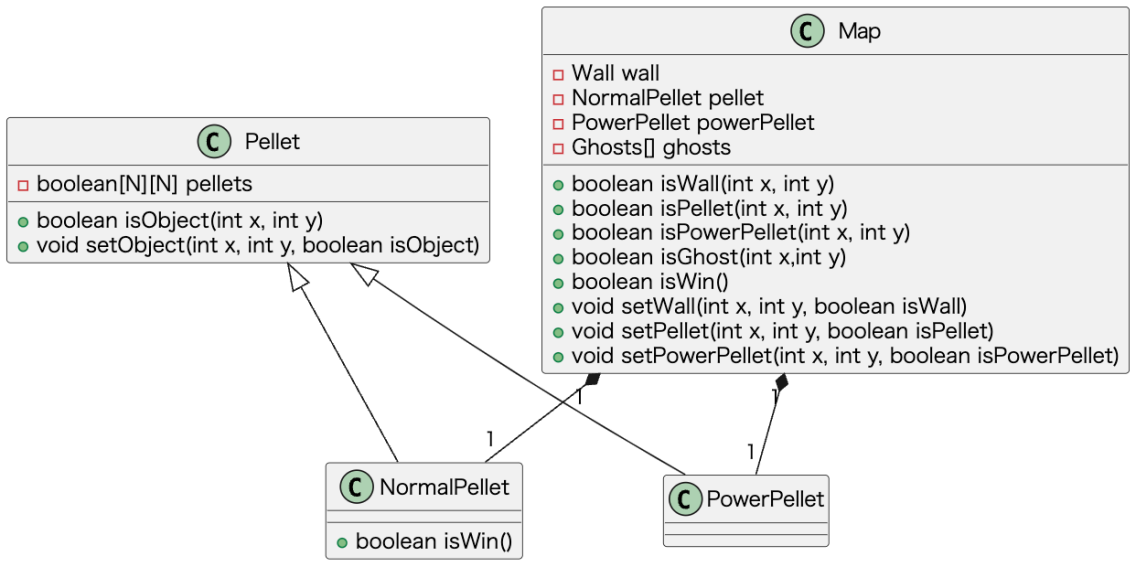
中間レポート　22B30098　一瀬達矢

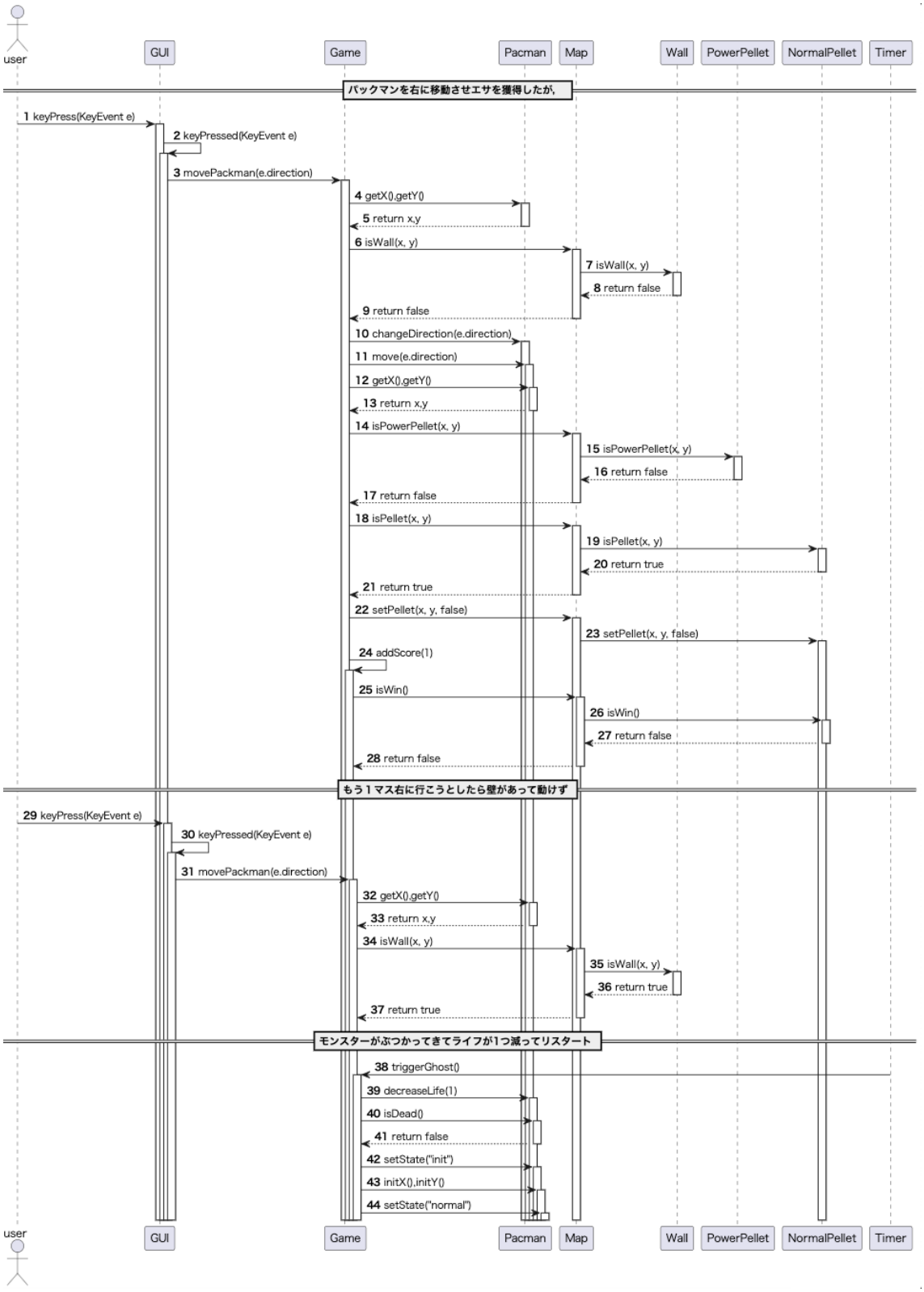
1. モデリングしたクラス図



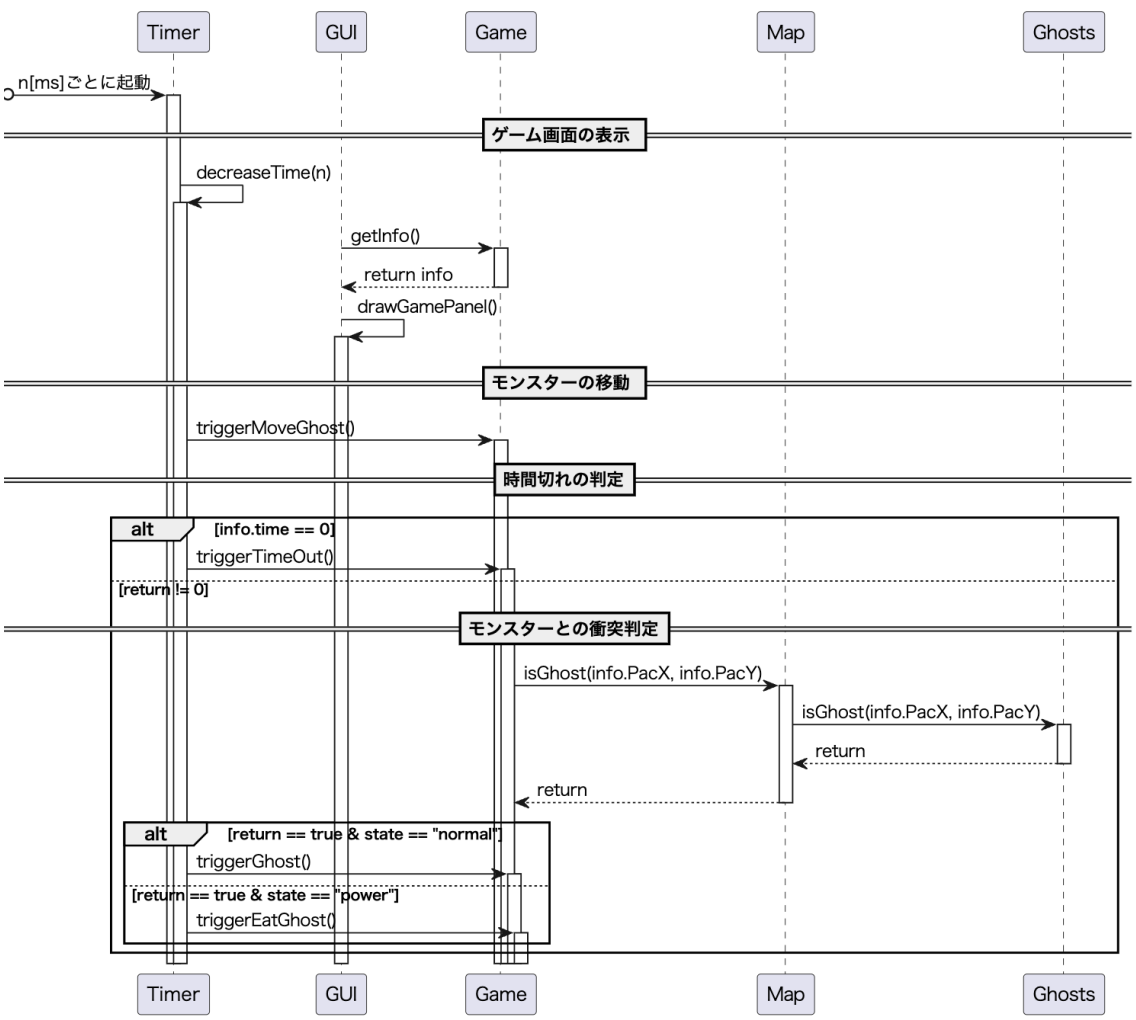
クラス図の中で Map,Pallet,PowerPellet,NormalPellet の関係についての表示が見にくくなっている箇所がある。詳細は以下のようになっている。



2. 設計したクラス群が以下のシナリオを実行するシーケンス図 • 「パックマンを右に移動させエサを獲得したが，もう 1 マス右に行こうとしたら壁があって動けず モンスターがぶつかってきてライフが 1 つ減ってリスタート」


user
GUI
Game
Pacman
Map
Wall
PowerPellet
NormalPellet
Timer
バックマンを右に移動させエサを獲得したが,
1 keyPress(KeyEvent e)
2 keyPressed(KeyEvent e)
3 movePackman(e.direction)
4 getX().getY()
5 return x,y
6 isWall(x, y)
7 isWall(x, y)
8 return false
9 return false
10 changeDirection(e.direction)
11 move(e.direction)
12 getX().getY()
13 return x,y
14 isPowerPellet(x, y)
15 isPowerPellet(x, y)
16 return false
17 return false
18 isPellet(x, y)
19 isPellet(x, y)
20 return true
21 return true
22 setPellet(x, y, false)
23 setPellet(x, y, false)
24 addScore(1)
25 isWin()
26 isWin()
27 return false
28 return false
もう１マス右に行こうとしたら壁があって動けず
29 keyPress(KeyEvent e)
30 keyPressed(KeyEvent e)
31 movePackman(e.direction)
32 getX().getY()
33 return x,y
34 isWall(x, y)
35 isWall(x, y)
36 return true
37 return true
モンスターがぶつかってきてライフが1つ減ってリスタート
38 triggerGhost()
39 decreaseLife(1)
40 isDead()
41 return false
42 setState("init")
43 initX(),initY()
44 setState("normal")

定期的な処理



3. 独自に想定した仮定（なければ「なし」とだけ記述）
   なし
4. 主要なクラスの簡単な説明．設計理由が含まれると好ましい
   全体的に構造に関しては Composite パターン、振る舞いに関しては Observer パターンを採用した。
   以下の構成は Composite パターンを設計理由としている。
   Game クラスは以下の主要な 4 つのフィールドを持つ。
   Pacman: パックマンの動作に関するメソッドを持つ
   Map : ゲーム内のオブジェクト全般を管理する
   Timer: ゲームの状態を管理する
   さらに、Map,Timer を具体的に述べると、

   Map はエサや壁、ゴーストをフィールドに持ち、それぞれに関するメソッドを持

つ。エサと壁については GameObject というインターフェイスを実現している。
ゴーストについてはゴーストごとに戦略が変わるので、抽象クラスで定義しゴーストの特色ごとに chase()というメソッドを変えている。

Timer は Observer パターンを参考にし、通常のタイマー機能に加えて、定期的に「ゴーストと衝突したかどうか」「ゲームオーバーかどうか」などを判定し、これらに該当する場合は notify に相当する trigger{トリガー名}という Game クラスのメソッドを呼ぶことで、ゲームの状態も管理している。
(trigger{トリガー名}は例えば、triggerTimeOut()など)

動作責務は主に Game が持ち、
判定責務は「ゴーストと接触したかどうか」「タイムアウトかどうか」を Timer が、「エサを全て食べたか」「パワーエサを食べたか」「パックマンが壁と衝突したか」などの Pacman を動かさない限り発生しないような状況の判定は Game が行っている。

工夫点
主要なクラスは Game クラスが全てを管理している、GUI は Game クラスにアクセスするだけで良く、その下でどのクラスが実際に実行しているかは気にしなくて良い
また、クラス図を見たときにゲームの構成が分かりやすいように、木構造を形成している。
(Composite パターン)
加えて、Timer というクラスにより定期的に「ゴーストと接触したかどうか」「タイムアウトかどうか」などを一括で管理することで可読性が向上している。

5. 感想
パックマンのゲームほどシンプルなものでも様々なモデリングが考えられることを実感した。実際のソフトウェア開発ではさらに複雑になると思うので、開発前にモデリングを議論することは極めて有意義なことだと思った。

今回のソースコードについては以下の Github 上に公開しています。
https://github.com/Tatsuya736482/reports/tree/main/CSCT361/midreport