知能プログラミング演習II 課題3

グループ8
29114003 青山周平

2019年11月05日

提出物　rep3

グループ　グループ8

メンバー

学生番号	氏名	貢献度比率
29114003	青山周平	null
29114060	後藤拓也	null
29114116	増田大輝	null
29114142	湯浅範子	null
29119016	小中祐希	null

1　課題の説明

必須課題 3-1 セマンティックネットのプログラムを参考に，グループメンバー全員（およびその周辺人物）についてのセマンティックネットを構築せよ．個人レポートには自分のみ（とその周辺）に関するセマンティックネットを示し，グループレポートには全員（とその周辺）に関するセマンティックネットを示せ．

必須課題 3-2 フレームのプログラムを参考に，自分達の興味分野に関する知識をフレームで表現せよ．その分野の知識を表す上で必須となるスロットが何かを考え，クラスフレームを設計すること．個人レポートには自分が作ったインスタンスフレームのみ（クラスフレームの設計担当者はクラスフレームも）を示し，グループレポートにはクラスフレームおよび全員分のインスタンスフレームを示せ．

必須課題 3-3 課題 3-1 または 3-2 で作った知識表現を用いた質問応答システムを作成せよ．なお，ユーザの質問は英語や日本語のような自然言語が望ましいが，難しければ課題 2 で扱ったような変数を含むパターン (クエリー) でも構わない．

発展課題 3-4 課題 3-1 または 3-2 で作った知識表現を図として示すためのユーザインターフェース (GUI) を設計し実装せよ．

発展課題 3-5 上記 3-3 で作成した質問応答システムを，DBpedia あるいは Wikidata 中の知識を使って質問に答えられるよう，拡張せよ．

2　必須課題 3-1

セマンティックネットのプログラムを参考に，グループメンバー全員（およびその周辺人物）についてのセマンティックネットを構築せよ．個人レポートには自分のみ（とその周辺）に関するセマンティックネットを示し，グループレポートには全員（とその周辺）に関するセマンティックネットを示せ．

私（とその周辺）に関するセマンティックネットの構造は，下図の通りである．

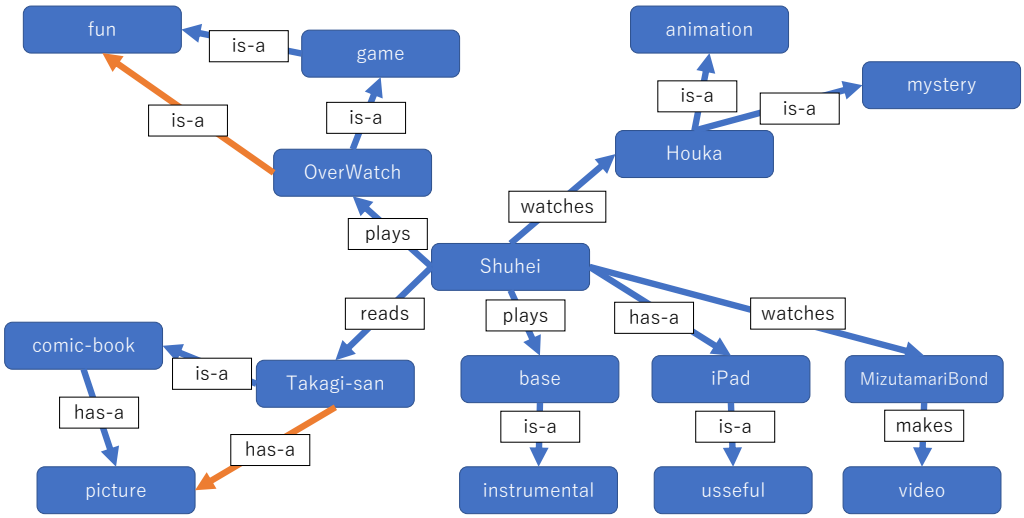

図 1: セマンティックネット

3　必須課題 3-2

フレームのプログラムを参考に，自分達の興味分野に関する知識をフレームで表現せよ．その分野の知識を表す上で必須となるスロットが何かを考え，クラスフレームを設計すること．
個人レポートには自分が作ったインスタンスフレームのみ（クラスフレームの設計担当者はクラスフレームも）を示し，グループレポートにはクラスフレームおよび全員分のインスタンスフレームを示せ．

私が作ったインスタンスフレームは下図のような設計となっている．

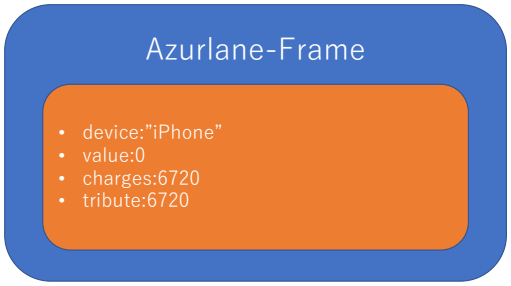

図 2: インスタンスフレーム

4　発展課題 3-4

課題 3-1 または 3-2 で作った知識表現を図として示すためのユーザインターフェース (GUI) を設計し実装せよ．

私の担当箇所は，発展課題 3-4 における GUI の Swing を用いた実装である．

4.1　手法

セマンティックネットのための GUI を実装するにあたり，以下のような方針を立てた．

1. ノードやリンクのデータを受け取る．
2. ノードを表示するためのパネルを作り，リンクを表示するための描写を設定する．また，パネルをマウスで移動できるようにする．
3. 検索・追加・削除の機能を追加する．

1．に関して，他のプログラムとのデータのやりとりの仲介プログラムを他の班員に作ってもらうことで，作業の分担・効率化を図った．

２．に関して，セマンティックネットのような複雑な図をユーザが自身で最も見やすい表示にするために，マウスによるドラッグでパネルを移動できるような仕様とした．

３．に関して，これらの機能をセマンティックネットとは別のパネルに分けたことで，より構造化された，拡張性・保守性の高い管理しやすいプログラムとなるようにした．

また，セマンティックネットの GUI を元にして，フレームの GUI も作った．

4.2 実装

セマンティックネットの GUI に関するプログラム SemNetGUI.java には以下のクラスが含まれる．

- SemNetGUI: メソッド main, クラス MenuPanel を実装した，フレームとメニューバーを実装するためのクラス．
- RelationMap: セマンティックネット全体を管理するパネルを実装するためのクラス．
- NodePanel: 1 つのノードに関するパネルを実装するためのクラス．
- LinkPanel: 1 つのリンクに関するパネルを実装するためのクラス．

フレームの GUI に関するプログラム FrameGUI.java には以下のクラスが含まれる．

- FrameGUI: メソッド main を実装したフレームを実装するためのクラス．
- FrameMap: 全フレームを管理するパネルを実装するためのクラス．
- FramePanel: 1 つのフレームに関するパネルを実装するための抽象クラス．
- ClassFramePanel: クラスフレームのときに用いる FramePanel クラスを継承したクラス．
- InstanceFramePanel: インスタンスフレームのときに用いる FramePanel クラスを継承したクラス．
- LinkPanel: 2 つのフレーム間の 1 つの継承関係に関するパネルを実装するためのクラス．

4.2.1 ノードやリンクのデータを受け取る

ノードやリンクの作成や取得には SemanticNet.java を呼び出す必要がある．そこで，その仲介のためのプログラム AccessData.java を他の班員

に作ってもらった．私の担当箇所である SemNetGUI.java からは AccessData.java を介することで SemanticNet.java で必要な処理をより簡単に行えるようにした．

RelationMap クラスのコンストラクタにおける AccessData を用いた処理をソースコード 1 に示す．

ソースコード 1: RelationMap クラスのコンストラクタ

```
1   RelationMap(String filename) {
2       sn = new SemanticNet();
3       ad = new AccessData(sn);
4       ad.start(filename); // セマンティックネットの構築
5       ...
6       ArrayList<Node> nodeList = ad.getNodes(); // セマ
            ンティックネットからノードの取得
7       ...
8       ArrayList<Link> linkList = ad.getLinks(); // セマ
            ンティックネットからリンクの取得
9       ...
10      }
11  }
```

4.2.2 ノードを表示するためのパネルを作り，リンクを表示するための描写を設定する．また，パネルをマウスで移動できるようにする

ノードのパネルの作成には JPanel クラスを継承した NodePanel クラスを実装し，利用した．このパネルはマウスでドラッグできるようにする必要があるため，それを実装するために MouseAdapter クラスの mousePressed メソッドと MouseMotionAdapter クラスの mouseDragged メソッドをオーバライドすることで実装した．

また，これらのイベントモデルは全て匿名クラスで実装した．これにより，ソースコードがスッキリし，挙動も分かりやすくすることができた．

NodePanel クラスのコンストラクタにおけるドラック処理をソースコード 2 に示す．

ソースコード 2: NodePanel クラスのコンストラクタ

```
1   NodePanel(Node node) {
2       ...
3       addMouseListener(new MouseAdapter() {
```

```
4               @Override
5               public void mousePressed(MouseEvent e) {
6                   draggedAtX = e.getX();
7                   draggedAtY = e.getY();
8               }
9           });
10
11          addMouseMotionListener(new MouseMotionAdapter() {
12              @Override
13              public void mouseDragged(MouseEvent e) {
14                  setLocation(e.getX() - draggedAtX +
                        getLocation().x, e.getY() - draggedAtY
                         + getLocation().y);
15
16                  for (LinkPanel lp : departFromMeLinks) {
17                      lp.update();
18                      lp.repaint();
19                  }
20                  for (LinkPanel lp : arriveAtMeLinks) {
21                      lp.update();
22                      lp.repaint();
23                  }
24              }
25          });
26          ...
27      }
```

まず，mousePressed メソッドでは e.getX(), e.getY() メソッドによって，マウスがクリックされた時点での座標が取得される．これを NodePanel クラスのフィールドである int 型変数 draggedAtX, draggedAtY に代入し，保持している．

次に mouseDragged メソッドでは，マウスがクリックされたままドラッグされる度に setLocation メソッドによって自身のインスタンスの座標を更新している．ここで setLocation にて行われている計算式について，x 軸について特筆して述べると，getLocation().x は NodePanel が配置されるコンポーネントにおける座標であるのに対して，e.getX() と draggedAtX は NodePanel 内における座標である．すなわち setLocation メソッドが実行される度に e.getX() の座標は draggedAtX に一致するため，この計算式で正しく動作することが証明できる．

また，mouseDragged メソッドでは，update メソッドと repaint メソッ

ドの呼び出しが行われている．これにより，ノードに対応するリンクについてもパネルが更新されている．

リンクのパネルについて，ノードのパネルと違い，パネルの移動だけでなく，パネルの拡大縮小・矢印の再描写が求められる．そこで，パネルの座標と大きさの設定を行える setSize メソッド，setSize メソッドを用いて更にラベルの位置も更新する update メソッドを実装した．LinkPanel クラス内のこれらのメソッドをソースコード 3 に示す．

ソースコード 3: setSize メソッド・update メソッド

```
1     void setSize() {
2         int lpX = getLeft();
3         int lpY = getTop();
4         int lpWidth = getRight() - lpX;
5         int lpHeight = getBtm() - lpY;
6         setBounds(lpX, lpY, lpWidth, lpHeight);
7     }
8
9     void update() {
10        Rectangle source = tail.getBounds();
11        Rectangle distance = head.getBounds();
12        setShortestDistance(source, distance);
13        setSize();
14
15        int fitX = -(mainPanel.getWidth() / 2);
16        int fitY = -(mainPanel.getHeight() / 2);
17        mainPanel.setLocation((getRight() - getLeft()) /
              2 + fitX, (getBtm() - getTop()) / 2 + fitY);
18    }
```

getLeft, getTop, getRight, getBtm メソッドはパネルの座標を取得するために実装されている．

4.2.3 検索・追加・削除の機能を追加する

検索・追加・削除の機能を一括して行うためのパネル MenuPanel は，構造的に SemNetGUI クラスの一部と考えられたため，SemNetGUI クラスの内部クラスとして実装した．これにより構造の意図を明確化できた．

検索・追加・削除のボタンが押されたときの処理は RelationMap クラス内の searchNode, addLink, removeLink メソッドを介して AccessData ク

ラス内の検索・追加・削除機能が呼び出されるような構造となっている．これにおいても，メソッドの位置や機能・関係を工夫したことで，クラスの構造化を考慮できた実装とすることができた．RelationMap クラスの addLink メソッドをソースコード 4 に示す．

ソースコード 4: addLink メソッド

```
1     boolean addLink(String text) {
2         Link link = ad.addData(text);
3         if(link == null) {
4             return false;
5         }
6         Node tail = link.getTail();
7         NodePanel tailPanel = nodes.get(tail);
8         if(!nodes.containsKey(tail)) {
9             tailPanel = new NodePanel(tail);
10            tailPanel.setBounds(20, 20, 180, 30);
11            add(tailPanel);
12            nodes.put(tail, tailPanel);
13        }
```

AccessData クラスのインスタンス ad を用いて，追加したリンク link を受け取っている．link の中身が null だったら失敗として false を返している．

新しく追加された LinkPanel は add メソッドによって自身の RelationMap インスタンス自身に追加することで，表示面での更新が行われている．

4.2.4 セマンティックネットの GUI を元にして，フレームの GUI を作る

FrameGUI.java は，基本的に SemNetGUI.java を元にして作られている．しかし Frame には Node と異なり，クラスフレームやインスタンスフレームといった概念が存在する．これを解決するために，FramePanel クラスを抽象クラスとして，クラスフレームのためのパネルには ClassFramePanel クラス，インスタンスフレームのためのパネルには InstanceFramePanel クラスを実装して利用することで，それぞれの挙動に合わせたパネルを作り分けることができた．

ClassFramePanel クラス，InstanceFramePanel クラスをソースコード 5 に示す．

ソースコード 5: ClassFramePanel クラス・InstanceFramePanel クラス

```
1  class ClassFramePanel extends FramePanel {
2      private ArrayList<InstanceFramePanel> list;
3
4      ClassFramePanel(String name, List<String> slotLabels,
           List<String> values) {
5          super(name, slotLabels, values);
6          list = new ArrayList<>();
7      }
8
9      void addInstance(InstanceFramePanel ifp) {
10         list.add(ifp);
11     }
12 }
13
14 class InstanceFramePanel extends FramePanel {
15     private ClassFramePanel par;
16
17     InstanceFramePanel(String name, ClassFramePanel cfp,
           List<String> values) {
18         super(name, new ArrayList(cfp.getSlots().keySet
               ()), values);
19         par = cfp;
20         cfp.addInstance(this);
21     }
22 }
```

4.3　実行例

実行時の第一引数を”aoyama”として，SemNetGUIを実行したところ，下図のような画面が得られる．

図 3: 初期状態

マウスのドラッグ操作により，ノードの配置を整理したところ，下図のような画面が得られる．

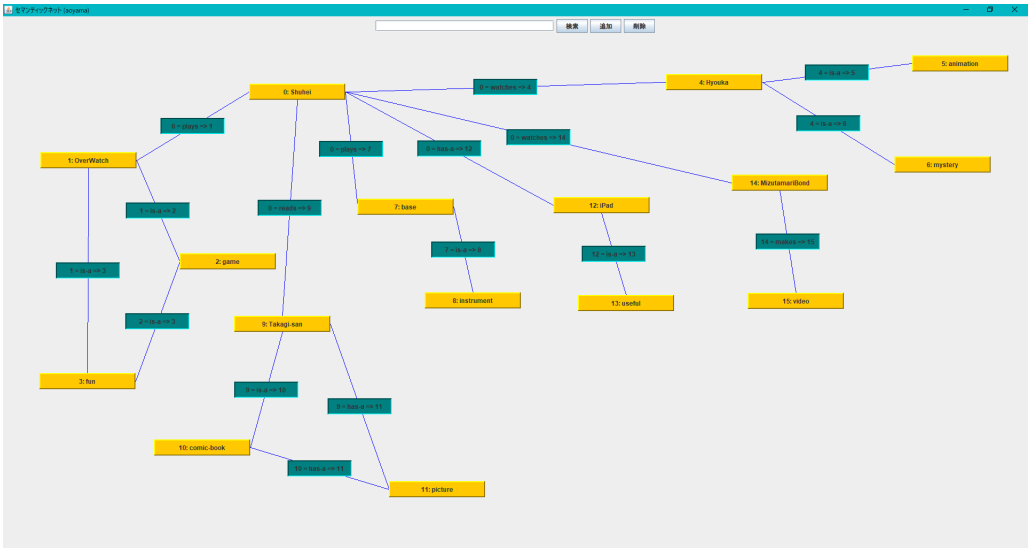

図 4: ドラッグ操作

”What does Shuhei plays ?”を入力し，検索ボタンをクリックしたところ，下図のような画面が得られる．画面最下部にその結果が表示されている．

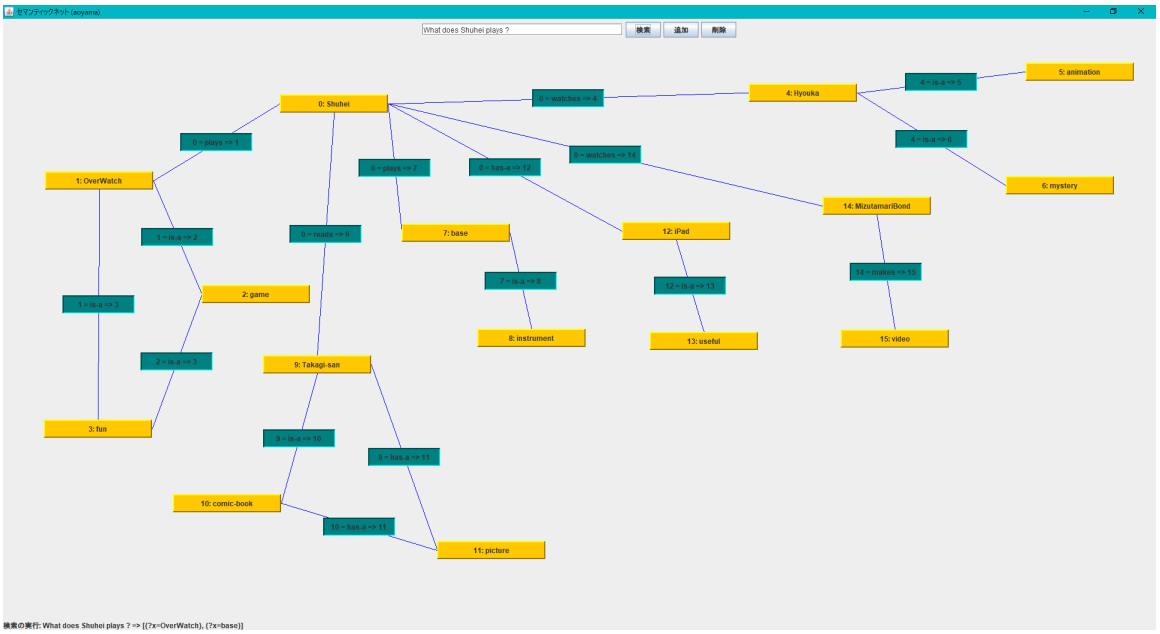

図 5: 検索操作

”Shuhei = is-a =¿ student”を入力し，追加ボタンをクリックしたところ，下図のような画面が得られる．

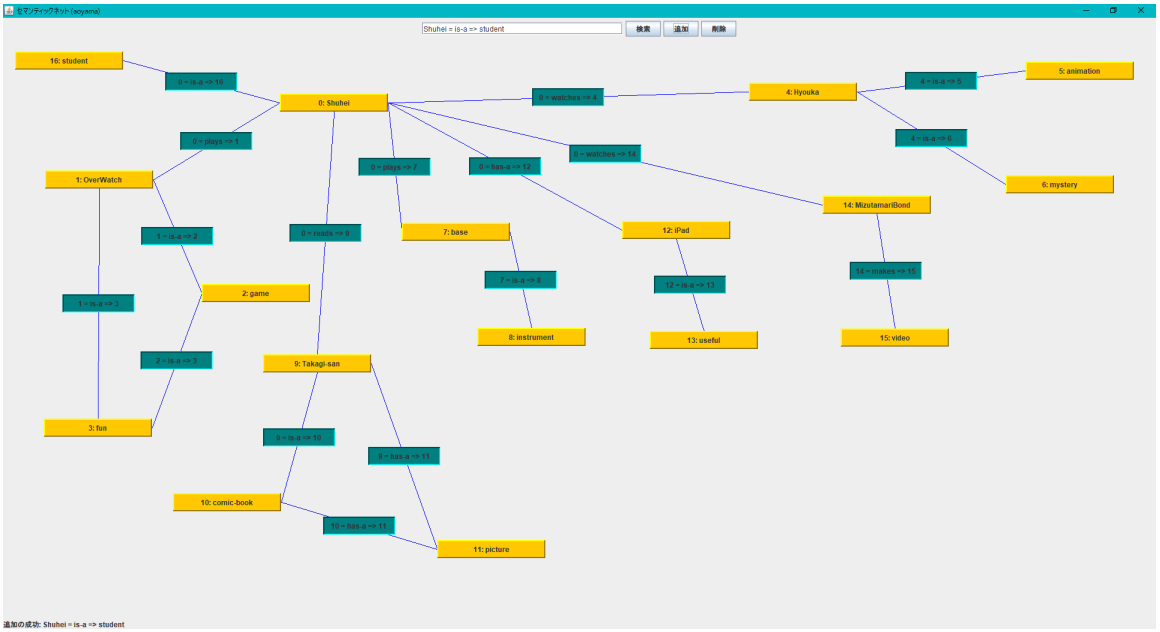

図 6: 追加操作

”Shuhei = has-a =¿ iPad”を入力し，削除ボタンをクリックしたところ，下図のような画面が得られる．

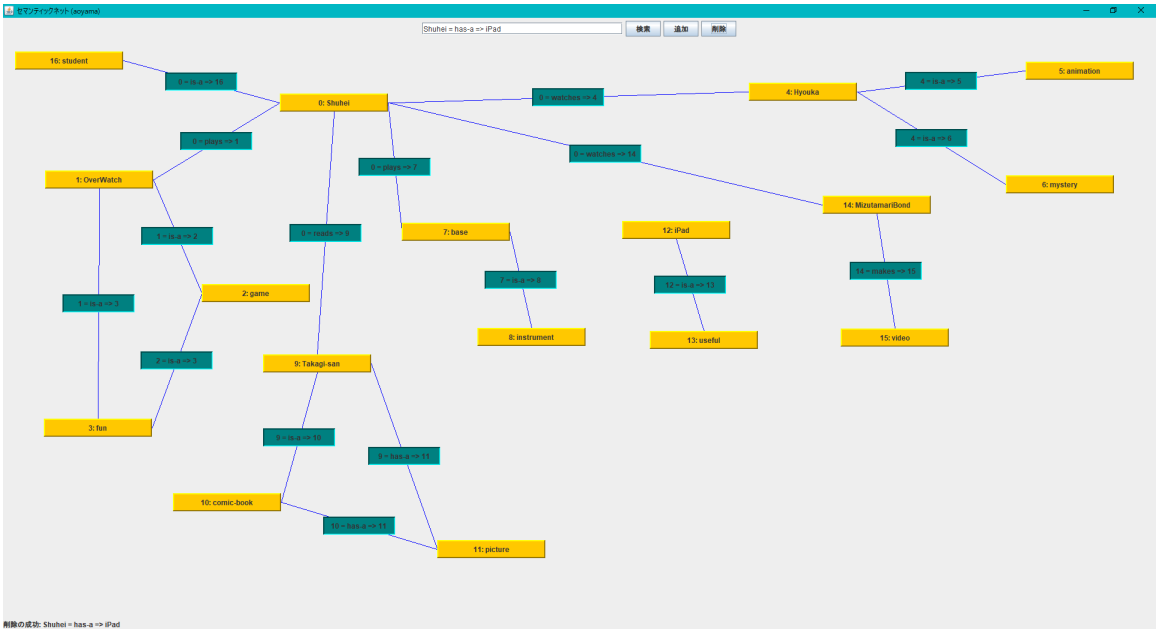

図 7: 削除操作

FrameGUI を実行したところ，下図のような画面が得られる．

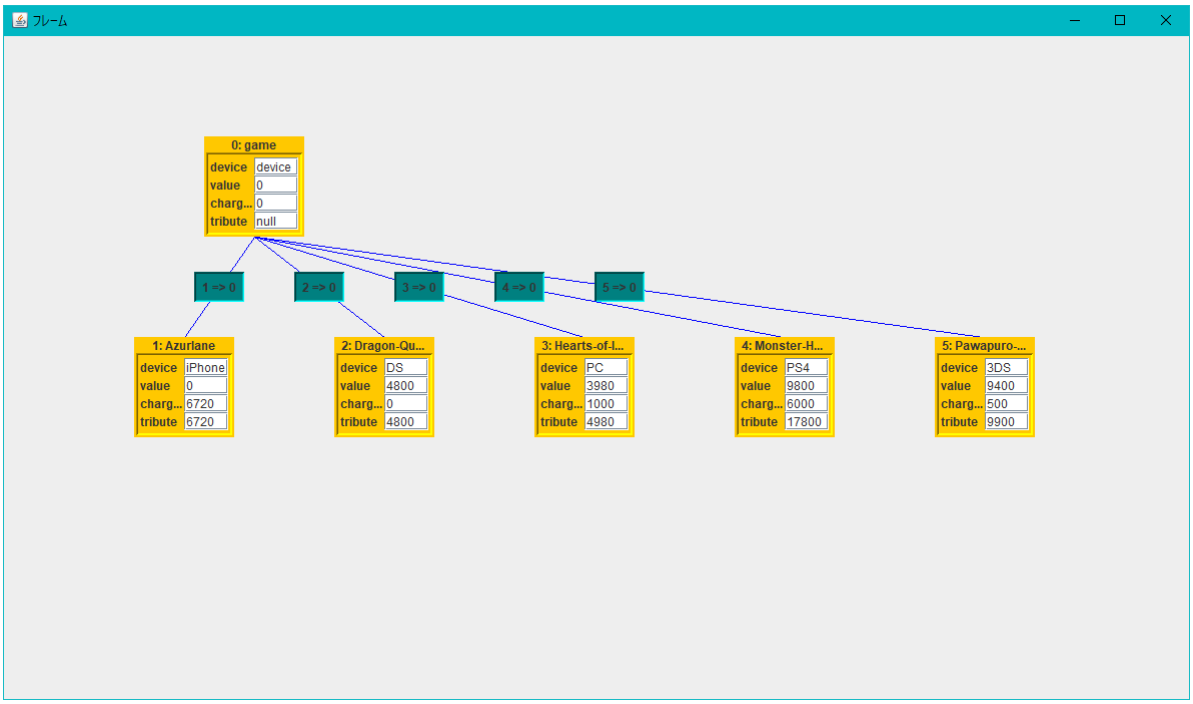

図 8: FrameGUI の実行

4.4 考察

これまでの 3 回に渡った Swing での GUI 実装をしてきたことによる，さまざまなコンポーネントやレイアウト等を利用した経験から考えると，図の表示には該当パネル内のレイアウトを無効にするのが最善策であると考えられる．第 1 回課題の状態空間の表示では SpringLayout を利用したが，そもそも 1 つ 1 つのパネルの関係性が独立した図においては，相対的な座標をレイアウトとして保持する必要がなく，今回のようにレイアウト無しで座標指定してパネルを配置したことは，より内部的にも真っ当で冗長さを削減したプログラムにすることができたと考えられる．

一方でレイアウトを無しにしたことで，コンポーネントの位置やサイズが自動で調整されるといったことも無くなるため，コンポーネントが表示されないと勘違いして躓いた節もあった．そのため，利用するクラス等を変える際は，変更前のクラスと比べてどのような特徴があるかをあらかじめ確認しておき，それを利用してプログラムを作るときも，常に今自分がどの機能を使っており，プログラムはどんな状態にあるかをイメージし続けることが重要だと考えられる．

Swing をより堅固に使いたいと思い，今回の実装ではクラス分けによる階層化をより一層意識して行った．結果として，パネルの表示が上手くいかなかったときの修正や，あるパネルの挙動を確かめたいとき，パネル内のコンポーネントを一括して管理したいときなどにおいて，RelationMap クラスや NodePanel クラスを作成し，階層化された構造にしておいたおかげで各インスタンスの挙動が把握しやすくなり，これらの実現が用意にできた．一方で，その場限りのクラスを作りたいときには，匿名クラスを用いて実装したり，内部クラスとして実装したりすることで，クラスが増えすぎて逆に複雑になってしまうことを防ぐことができた．

今回の実装で出来たような，クラスを中心とする階層化を意識して，オブジェクト指向の特徴を捉えた Java プログラミングを今後も心がけてゆきたい．

5 感想

GUI 上で図を表示することについては第 1 回課題で実装した基盤があるため，その分楽にできたが，更に高みを目指すべく，パネルの更新やマ

ウスによるドラッグなど，拡張した機能の実現に積極的に取り組み，実装までできてよかった．

しかし，第 1 回のときと比べて，実際にメインの処理を行う方のプログラム（今回で言う SemanticNet.java や AIFrameSystem.java といった GUI とは別の部分）が圧倒的に複雑であり，それを読み解いて GUI に合わせた処理を考える，ということが非常に大変だった．その点，班員に協力を仰いで AccessData.java のようなデータの仲介のためのプログラムを作ってもらえたのは非常に助かり，また，上手に連携できたため，私の中でグループワークをするスキルが上がったことを感じ，嬉しく思う．

今後も，個人では限界があるものでも，グループメンバーと協力・分担することで，より高みを目指せるようなプログラム作りを意識して取り組んでゆきたい．

参考文献

TATSUO IKURA : 『Swing を使ってみよう - Java GUI プログラミング』 https://www.javadrive.jp/tutorial/ (2019/11/05 アクセス)