知能プログラミング演習 II 課題 3

グループ 8
29114116 増田大輝

2019 年 10 月 25 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	NoData
29114060	後藤拓也	NoData
29114116	増田大輝	NoData
29114142	湯浅範子	NoData
29119016	小中祐希	NoData

1　課題の説明

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.

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.

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.

2　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.

2.1　手法

教科書 3.1 節のセマンティックネットのプログラムを参考にしてテキストファイルからデータを読み込み, セマンティックネットを構築する. semantic_net/下に以下のプログラムを配置した.

Node.java　セマンティックネットを構成するノード. 主に名詞に当たる.
Link.java　ノード間を結ぶリンク. 主に動詞に当たる.
SemanticNet.java　ノードとリンクにより構成されるセマンティックネット.
SemanticNetTest.java　テキストファイルのデータからセマンティックネットを構成する.

特に,SemanticNetTest.java では, 指定したテキストファイルから各学生のデータを読むことにより, 名詞をノード, 動詞をリンクとしてセマンティックネットを構築する. 以下に, 今回の課題で作成した個人のセマンティックネットを示す.

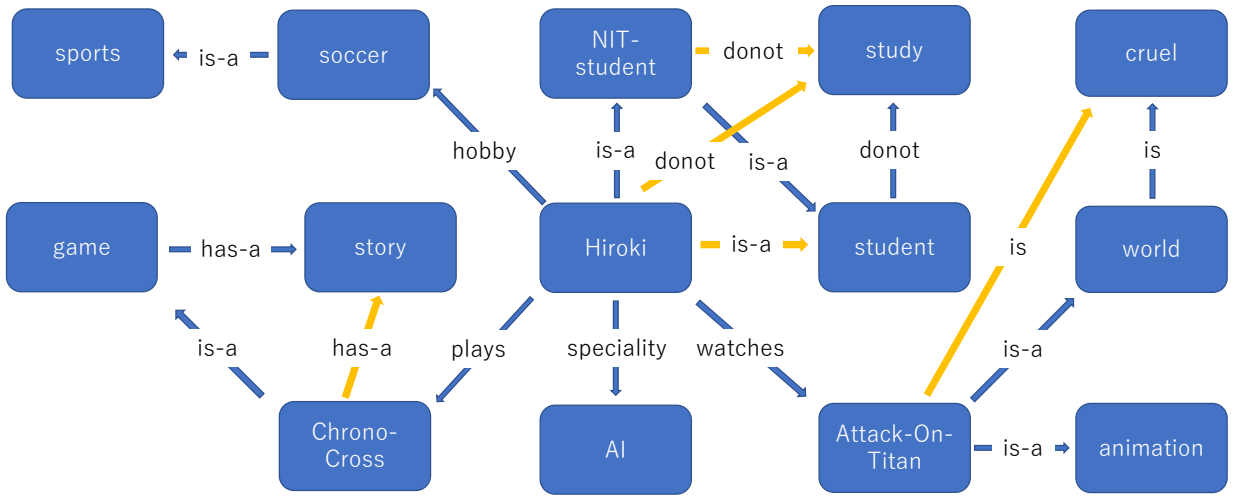

図 1　増田セマンティックネットの概念図

2.2　実装

この課題では, 教科書のセマンティックネットのプログラムを利用する SemanticNetTest.java を作成した. このクラスでは, 実行時に第一引数として渡された文字列に該当する members/下と queries/下のテキストファイルからデータを読み取って実行する. 以下で, 作成したメソッドとその役割について簡単に述べる.

readTextFile　テキストファイルファイルからデータを読み込み,List¡String¿として返却する.

splitStatement　単文を空白で分割する.

addLink　分割された単文からリンクを作成し, セマンティックネットに追加する.

addQuery　分割された単文からクエリを作成し, セマンティックネットに追加する.

以上の関数を組み合わせ, テキストファイルから読み込まれたデータからセマンティックネットを構築・検索し, 出力を行うように main() 文に実装した. main 文内における具体的な実装手順を以下のソースコードに示す.

ソースコード 1　SemanticNetTest.java の main() 内部

```
1     SemanticNet sn = new SemanticNet();
2     if(args.length == 1) {
3         // 個人のデータをmembers/下のテキストファイルから読み込む
4         List<String> statementList = readTextFile("members/"+args
              [0]+".txt");
5         for(String statement: statementList) {
6             // 単文を主格･動詞･目的格に 3分割する
7             List<String> splitList = splitStatement(statement);
8             // セマンティックネットにリンクを追加する
9             addLink(sn, splitList);
10        }
11        sn.printLinks();
12        sn.printNodes();
13
14        // 検索クエリをqueries/下のテキストファイルから読み込む
15        List<String> queryList = readTextFile("queries/"+args[0]+".txt
              ");
16        ArrayList<Link> query = new ArrayList<Link>();
17        for(String queryStatement: queryList) {
18            // 単文を主格･動詞･目的格に 3分割する
19            List<String> splitList = splitStatement(queryStatement);
20            // 新たなクエリを追加する
21            addQuery(query, splitList);
22        }
23        sn.query(query);
24    }
```

このソースコードからも読み取れるように, セマンティックネットを構築する際のプログラム (3〜10 行目) と, クエリを追加する際のプログラム (14〜22) では, 構造が非常に似ている.

これは,readTextFile メソッドと splitStatement メソッドによって両者で共通する処理が行われるためである.

反対に, 両者の差をそれぞれ addLink メソッドと addQuery メソッドによって吸収している.

共通する処理をメソッドに切り出すことで main() 文内の複雑化を防ぐと共に, 各種メ

ソッドを目的が明確かつ汎用性があるように設計した．このため,main() 文では目的に応じて使用するメソッドを切り替えているに止まり, 差分の吸収機能を果たすのみであることから構造が単純なものとなっている．

2.3 実行例

セマンティックネットを構築するためのデータとして以下のようなデータを semantic_net/members/下に用意した．

ソースコード 2 semantic_net/members/masuda.txt

```
1  soccer is-a sports
2  Hiroki is-a NIT-student
3  Hiroki speciality AI
4  Chrono-Cross is-a game
5  game has-a story
6  Hiroki hobby soccer
7  Hiroki plays Chrono-Cross
8  NIT-student is-a student
9  student donot study
10 Attack-On-Titan is-a animation
11 Hiroki watches Attack-On-Titan
12 Attack-On-Titan is-a world
13 world is cruel
```

また, クエリによる検索を行うための以下のデータを semantic_net/queries/下に用意した．

ソースコード 3 semantic_net/queries/masuda.txt

```
1  ?y plays Chrono-Cross
2  ?y is-a student
3  ?y hobby soccer
4  ?y watches Attack-On-Titan
```

semantic_net/下においてコンパイル後に以下のコマンドを実行することで対象のテキストファイルに含まれるデータのセマンティックネットを構築できる．
java SemanticNetTest [名前]
次に,”java SemanticNetTest masuda”の実行例を示す．

ソースコード 4　java SemanticNetTest masuda の実行例

```
1   ~/Programming2/Work3/semantic_net
2   ●java SemanticNetTest masuda【 fix/semantic_net 】
3   *** Links ***
4   soccer =is-a=> sports
5   Hiroki =is-a=> NIT-student
6   Hiroki =speciality=> AI
7   Chrono-Cross =is-a=> game
8   game =has-a=> story
9   ( Chrono-Cross =has-a=> story )
10  Hiroki =hobby=> soccer
11  Hiroki =plays=> Chrono-Cross
12  NIT-student =is-a=> student
13  ( Hiroki =is-a=> student )
14  student =donot=> study
15  ( NIT-student =donot=> study )
16  ( Hiroki =donot=> study )
17  Attack-On-Titan =is-a=> animation
18  Hiroki =watches=> Attack-On-Titan
19  Attack-On-Titan =is-a=> world
20  world =is=> cruel
21  ( Attack-On-Titan =is=> cruel )
22  *** Nodes ***
23  soccer
24  sports
25  Hiroki
26  NIT-student
27  AI
28  Chrono-Cross
29  game
30  story
31  student
32  study
33  Attack-On-Titan
34  animation
35  world
36  cruel
```

```
37      *** Query ***
38      ?y =plays=> Chrono-Cross
39      ?y =is-a=> student
40      ?y =hobby=> soccer
41      ?y =watches=> Attack-On-Titan
42      [{?y=Hiroki}]
```

2.4 考察

セマンティックネットを構築する際に特に注目したのが,is-a 関係による特性の継承である. `Chrono-Cross =is-a=> game` によって下位概念である「Chrono-Cross」から上位概念である「game」が関連付けられ, `game =has-a=> story` によって「game」に特性が紐づけられる. 結果として, `Chrono-Cross =has-a=> story` が導出される実装から, 知識表現とオブジェクト指向プログラミングの親和性の良さを体感した.
セマンティックネットを活用することによって, 単純なデータの列挙から新たな知識を導出できることができる.
しかし, 知識推論は一般から具体への方向にのみ働くことに注意が必要である.

3 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.

クラスフレーム設計担当者であるので,3.1 節にて設計したクラスフレームを示す.

3.1 手法

教科書 3.1 節のフレームのプログラムを参考にしてテキストファイルからデータを読み込み, フレームを構築する. semantic_net/下に以下のプログラムを配置した.

AIFrame.java　フレームのスーパークラス.

AIClassFrame.java　フレームの具体化クラス. クラスフレームを表す.

AIInstanceFrame.java　フレームの具体化クラス. インスタンスフレームを表し, いずれかのクラスフレームのインスタンスである.

AISlot.java　フレームが持つスロット.

AIDemonProc.java　デモン手続きのスーパークラス.

AIDemonProcReadTest.java　デモン手続きの具体化クラス. 読み込みデモン手続きを行う.

AIDemonProcWriteTest.java　デモン手続きの具体化クラス. 書き込みデモン手続きを行う.

AIWhenConstructedProc.java　クラスをデモン手続きとして利用するための抽象クラス.

AIFrameSystem.java　フレームシステムを表す.

FrameTest.java　テキストファイルのデータからフレームを構成する.

特に,FrameTest.java では, ゲームソフトを表すクラスフレームを作成し, 対応デバイス・本体価格・課金額・累計金額の四種類のスロットを作成する. その後, インスタンスフレームを作成し, グループメンバーそれぞれが提案したゲームソフトのフレームとする.
以下に, ゲームクラスフレームのスロットについて示す.

device　対応ソフト, すなわち, ゲームソフトのプラットフォームを指定する.String 型オブジェクトとし,"device" をデフォルトとする.

value　本体価格を指定する.Integer 型のオブジェクトとし,0 をデフォルトとする.

charges　課金額を指定する.Integer 型のオブジェクトとし,0 をデフォルトとする.

tribute　累計金額を指定する.Integer 型のオブジェクトとし,value+charges をデフォルトとする.

累計金額を計算するにあたって, デモン手続きを用いて本体価格と課金額の和をとる. スロット値を読み取る際に計算を行うので,AIDemonProcReadTest.java クラスに記述を追加した.
次に, クラスフレームの図を示す.

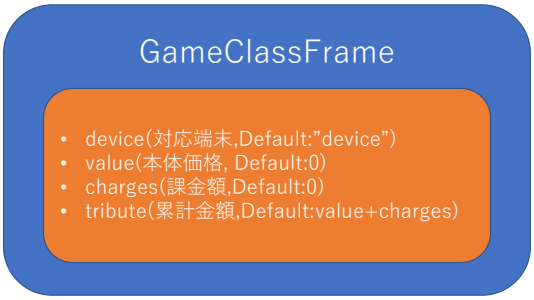

図 2　ゲームクラスフレームの概念図

以下に, 私個人が作成したインスタンスフレーム Monster-Hunter フレームの図を示す. ただし, 各スロット名の横の値は, 書き込みデモン手続きによって更新されるインスタンスフレームの各スロットの値である.

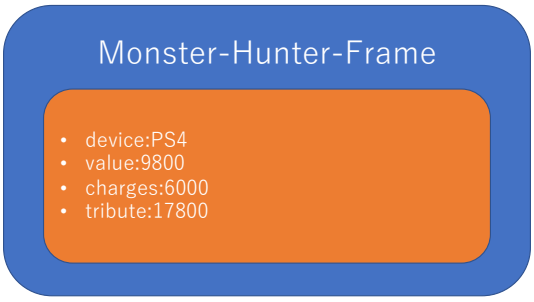

図 3　Monster-Hunter フレームの概念図

3.2 実装

この課題では, 教科書のフレームのプログラムを利用する FrameTest.java を作成した. このクラスでは, 実行時に第一引数として渡された文字列に該当する games/下のテキストファイルからデータを読み取って実行する.
以下で, 作成したメソッドとその役割について簡単に述べる.

readTextFile　テキストファイルファイルからデータを読み込み,List¡String¿として返却する.

defineClassFrame　クラスフレームとスロットを定義. また, デモン手続きを設定する.

outputSlots　スロット値の一覧を表示する.

また, インスタンスフレームのスロット値の更新と出力は以下のソースコードにある順で行う.

ソースコード 5　インスタンスフレームのスロット値の更新と出力プログラム

```
1   // device,value,charges,tribute はデフォルト値
2   System.out.println( "device,value,charges,tribute はデフォルト値"
        );
3   outputSlots(fs, instanceName, false);
4   System.out.println();
5
6   // tribute はデフォルト値
7   System.out.println( "tribute はデフォルト値" );
8   fs.writeSlotValue( instanceName, "device", new String(slotList.get
        (0)));
9   fs.writeSlotValue( instanceName, "value", new Integer(slotList.get
        (1)) );
10  fs.writeSlotValue( instanceName, "charges", new Integer(slotList.
        get(2)) );
11  outputSlots(fs, instanceName, false);
12  System.out.println();
13
14  // 再びデフォルト値
15  System.out.println( "再びデフォルト値" );
16  fs.writeSlotValue( instanceName, "tribute", new Integer(slotList.
        get(3)) );
```

```
17      outputSlots(fs, instanceName, true);
```

さらに, 以下の読み込みデモン手続きを用いて,$tribute = value + charges$ を実装する.

ソースコード 6　AIDemonProcReadTest.java における読み込みデモン手続き

```
1       Object value = inFrame.readSlotValue( inFrameSystem, "value",
            false );
2       Object charges = inFrame.readSlotValue( inFrameSystem, "charges",
            false );
3       if ( value instanceof Integer && charges instanceof Integer) {
4           int v = ((Integer) value).intValue();
5           int c = ((Integer) charges).intValue();
6           return AIFrame.makeEnum( new Integer( (int) v + c) );
7       }
```

3.3　実行例

フレームを構築するためのデータとして, 以下のデータを frame/games/下に用意した.

ソースコード 7　frame/games/Monster-Hunter.txt

```
1       PS4
2       9800
3       6000
4       17800
```

frame/下においてコンパイル後に以下のコマンドを実行することで対象のテキストファイルに含まれるデータのフレームを構築できる.

java FrameTest [ゲーム名]

次に,"java FrameTest Monster-Hunter"の実行例を示す.

ソースコード 8　java FrameTest Monster-Hunter の実行例

```
1       ~/Programming2/Work3/frame
2       ●java FrameTest Monster-Hunter【 fix/frame 】
3       クラスフレーム:game
4
5       インスタンスフレーム:Monster-Hunter
6
```

```
7     device,value,charges,tribute はデフォルト値
8     スロット値一覧
9     device:device
10    value:0
11    charges:0
12    tribute:0
13
14    tribute はデフォルト値
15    スロット値一覧
16    device:PS4
17    value:9800
18    charges:6000
19    tribute:15800
20
21    再びデフォルト値
22    スロット値一覧
23    device:device
24    value:0
25    charges:0
26    tribute:15800
```

3.4 考察

フレームは概念的にはセマンティックネットを拡張したものであり, クラスフレームからの継承の要素を残しつつも, それぞれのフレームが直接スロットを保持することによって, 特性の記述も可能である. また, セマンティックネットとの大きな違いとして上げられる点として, デモン手続きが存在することによって, クラスフレームのスロットデータをデフォルト値として保持しながらも, インスタンスフレームでスロットデータの書き換えが可能である. また, データ格納の性質上, セマンティックネットは自然言語からの構築に適していると考えられるが, フレームの場合はデータベースからの構築に適していると考えられる. それぞれの概念の得意なデータの形式を考えることで, より効果的な知識システムを構築できると考えられる.

参考文献

[1] Java による知能プログラミング入門 –著：新谷 虎松