新人研修報告書

平成 20 年 6 月 30 日
江藤　優樹

1.目的

Java でのアプリケーション開発ができるレベルまで Java 言語を理解することを目的とする。

2.方法

Web サイト、参考書を活用しながら、課題を行うことで Java 言語の基礎を習得する。習得した知識を応用し、Web アプリケーションの開発を行うことで、Java 言語の更なる理解を深める。

3.結果

学生時代に理解が足りなかった継承やインターフェース、抽象クラスについて理解することができた。また、Web アプリケーションの開発により、データーベースの操作、SQL について理解することができた。

4.結論

Java 言語の基礎の理解と Web アプリケーション開発に必要な知識を得ることができた。より高度なプログラムを組めるよう更なる努力をする。

5.コメント

3 ヶ月の研修期間で学生時代に習得できなかった多くの知識を得ることができた。
しかし、まだまだ習得しなければならない知識も多いので、日々の精進を忘れずにプログラミングのスキルアップを目指す。

目次

頁

[1] 目的

Java 言語をアプリケーション開発ができるレベルまで理解することを目的とする。

[2] 対象システム

研修で使用する開発環境を以下に示す

- PC：Hewlett-Packard HP dx2000 ST(PE677AV)
- メモリ：504MB
- JDK：jdk-1_5_0_14
- Struts：Struts1.3.8
- Tomcat：apache Tomcat 6.0
- MySQL：MySQL Server 5.0
- Eclipse：eclipse-SDK-3.3.1.1

[3] 方法

3.1 コーディング規約

コーディング規約を読み、その内容を理解することで見やすいプログラムを作成するよう心がける。特に、メソッド名、クラス名の命名に注意すること。

3.2 継承及びオーバーライドの習得

継承及びオーバーライドを習得するために Robocode を用いて以下のプログラム(ロボット)を作成する。

- 課題 1：Robot クラスを継承し、三角形と四角形の動きを交互にするロボット(動くだけで他のロボットに対して攻撃は行わない)
- 課題 2：AdvancedRobot クラスを継承し、様々な方向にスピンしながら移動するロボット(動くだけで他のロボットに対して攻撃は行わない)
- 課題 3：画面中央に移動し、360 ° 敵を詮索、敵を発見次第攻撃するロボット(ロボットは中央に移動後は動かない)

3.3 インターフェースの習得

インターフェース、Stratregy パターンを習得するために Robocode を用いて以下のプログラム(ロボット)を作成する。

- 課題 1 で作成した三角形と四角形の動きを交互にするロボットを拡張し、弾が当てられる度に三角形と四角形の動きを交互に変えるロボット。

3.4 抽象クラスの習得

抽象クラスを習得するために Robocode を用いて以下のプログラム(ロボット)を作成する。

- 3.3 章インターフェースの習得で作成した弾が当たる度に三角形と四角形の動きを交互に変えるロボット。(今回は抽象クラスを用いて作成)

3.5 Java 言語基礎習得状況の確認

Java 言語基礎習得状況の確認のために、オリジナルロボットを作成する。(インターフェースと抽象クラスは必須)

3.6 Web アプリケーションの作成

学校給食会システムのプログラムを分担し、割り当てられた以下のプログラムを作成する。

- 製品マスタ保守(マスタ保守)
- 製品分類マスタ保守(マスタ保守)
- 製品マスタリスト(マスタリスト)
- 製品構成マスタリスト(マスタリスト)
- 材料使用報告書(帳票)
- パン重量別合表(帳票)
- 受払報告書(帳票)

プログラム完成後、単体テスト仕様書を作成し、単体テストを行う。

[4] 結果

4.1 コーディング規約

コーディング規約を読むことで、プログラムを作成する際にどういったことに気を付けなければならないかを理解することができた。特に、メソッド名、クラス名の命名に注意しなければならないことを認識して、コーディングを行ったが、自分にしかわからない名前を付けていることが多々あった。
今後、わかりやすく、誰もが見やすいプログラムを作成していくためにコーディング規約をいつも意識する必要がある。

4.2 継承及びオーバーライドの習得

専門学校時代に概念は学んでいた。実際に組んでみることでより深く理解することができた。

・継承とは

あるクラスの実装を土台とし、別のクラスを実装するための方法。元のクラスをスーパークラス、継承するクラスをサブクラスと呼ぶ。

一つのクラスは任意の個数のサブクラスを持つことが出来るが、スーパークラスを複数持つことは出来ない。これを単一継承と呼び、クラスは継承されることで枝葉を広げていく木構造を成している。クラスの木構造の頂点(root)となるクラスは java.lang パッケージにある object クラスである。

スーパークラスを明示しない場合は自動的にこのクラスから派生されたこといなるが、スーパークラスを明示した場合もそのスーパークラスが object クラスを継承しているのですべてのクラスが object クラスをスーパークラスに持つことと変わりはない。

サブクラスは自身で実装したメソッドや変数に加え、スーパークラスのメソッドや変数も全て継承して使用できる。また、サブクラスをスーパークラスとして継承することも出来る。

・オーバーライドとは

継承時に、スーパークラスで定義されたメソッドと同じ名前、引数を持つメソッドをサブクラスでもう一度定義すること。

置き換えられたスーパークラスのメソッドは消えることはなく、オーバーライドメソッドからいつでも呼ぶことができる。

継承の実装例を以下に示す。

(例)クラスの継承

```
class　クラス名　extends　スーパークラス{
                内容
}
```

作成したロボットの説明を次項ページより示す。

- **課題 1 のロボット概要**

  三角形と四角形の動きを交互に行うロボット。

  独自のロボットを作成するために Robocode で用意されているロボットである Robot を継承し、各ロボットのメインメソッドとなる run メソッドをオーバーライドする。

  課題１のロボットの run メソッドの処理フローチャートを図 4.2.1 に示す。図 4.2.1 に示すように、run メソッドでは三角形と四角形の動きを交互に動作させるロジックを実装した。

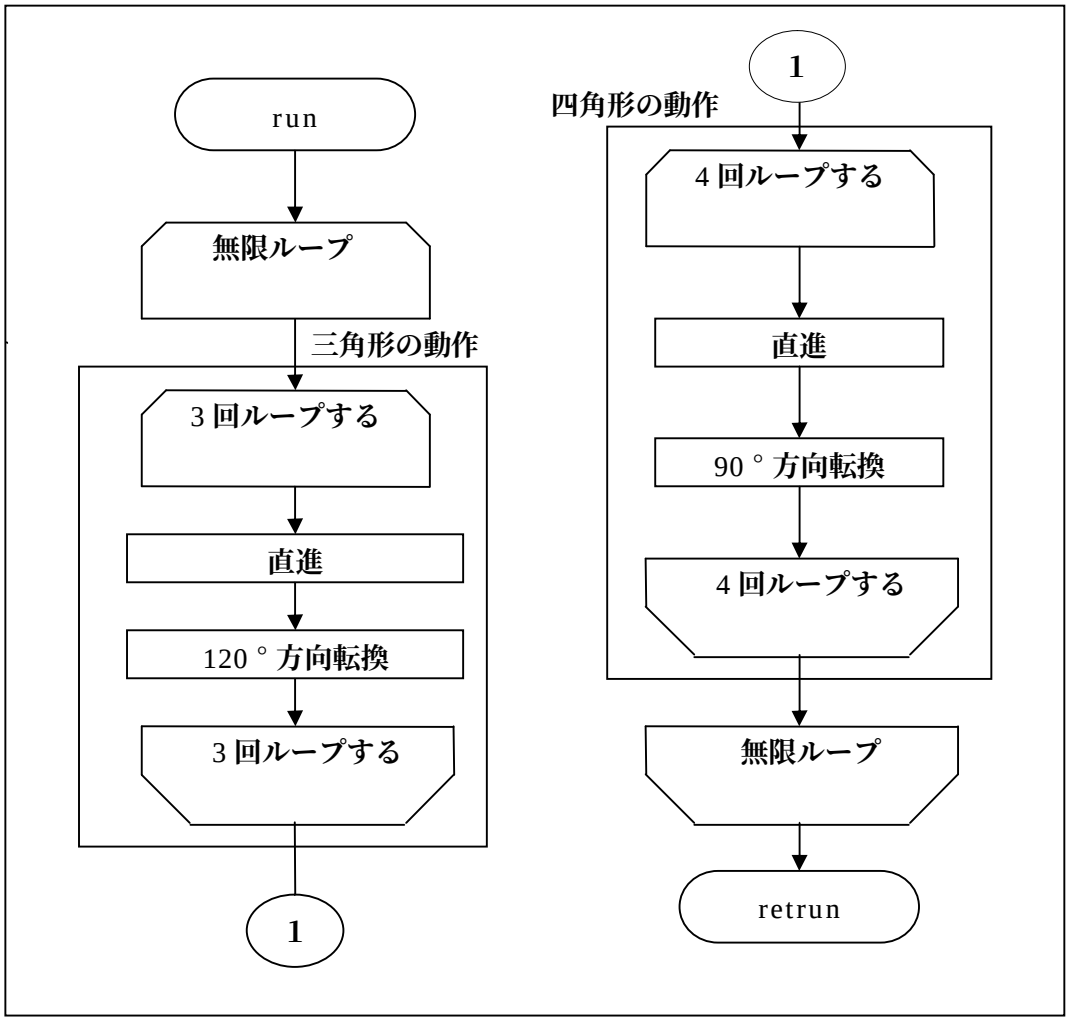

図 4.2.1　課題１のロボットの run メソッドの処理フローチャート

- **課題 2 のロボット概要**

  スピンをしながら移動するロボット。

  Robocode で用意されているロボットである AdvencedRobot を継承し、各ロボットのメインメソッドとなる run メソッドをオーバーライドする。課題２のロボットの run メソッドの処理フローチャートを図 4.2.2 に示す。図 4.2.2 に示すように、run メソッドではスピン移動のロジックを実装した。

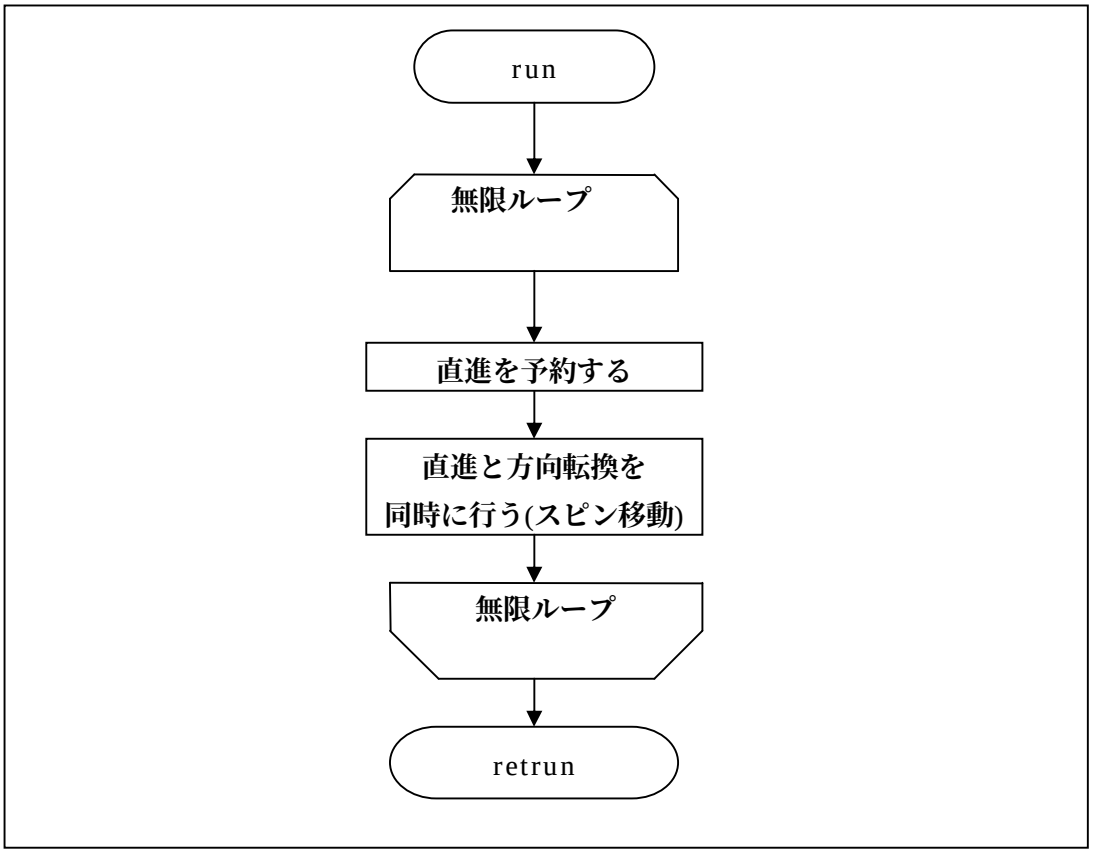

図 4.2.2　課題２のロボットの run メソッドの処理フローチャート

- **課題３のロボット概要**
  画面中央に移動後、360°敵を詮索、敵を発見次第攻撃するロボット。
  Robocode **で用意されているロボットである** AdvencedRobot **を継承し、各ロボットのメイン･メソッドとなる** run **メソッドをオーバーライドする。**
  課題 3 **のロボットの** run **メソッドの処理フローチャートを図** 4.2.3 **に示す。**
  図 4.2.3 **に示すように、画面の中央に移動するメソッドを** run **メソッドの最初に実行、敵を発見するためにレーダーを回すメソッドを実行する。**
  敵を発見すると敵を発見した時に呼ばれるメソッドで攻撃のメソッドを実行し、攻撃を行う。

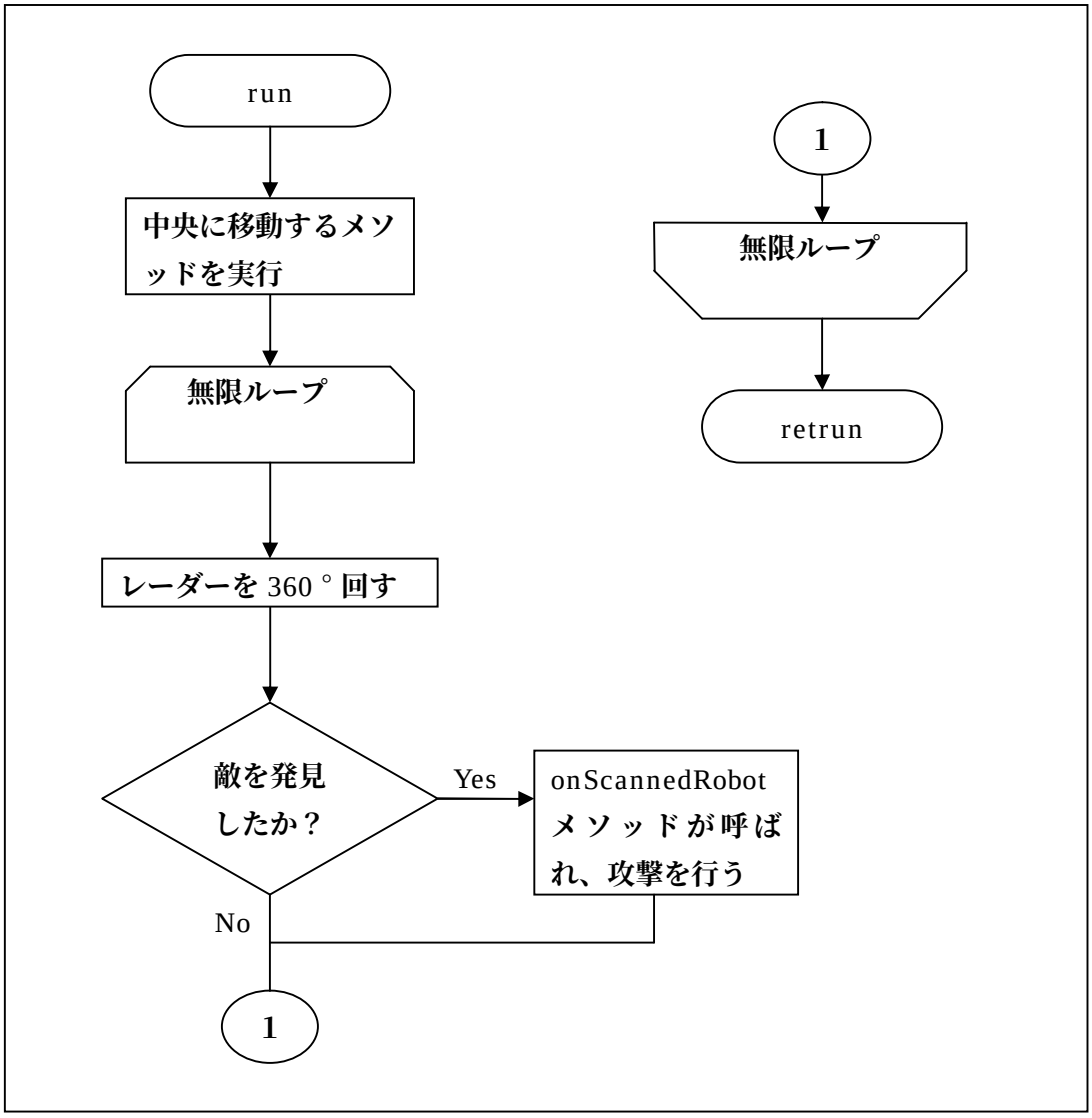

図 4.2.3　**課題** 3 **のロボットの** run **メソッドの処理フローチャート**

4.3　インターフェースの習得

インターフェースを使用したプログラムの作成を行うことによって、インターフェースについて理解した。

・インターフェースとは

定数とメソッドの宣言を一つにまとめたもので実装したクラスは同一のインターフェースを実装したクラス間で同等の機能を持つ。
インターフェースを実装したクラスはインターフェースで定義されているメソッドを全て実装しなければならない。
インターフェースの利点はクラスが複数のインターフェースを実装できる。
インターフェースのメソッドには自動的に public と abstract が付加される。
複数のインターフェースを実装するにはカンマ区切り実装するインターフェースを列挙する。
インターフェースの実装はクラスに依存しない。

インターフェースの実装例を以下に示す。

(例)インターフェース作成

```
public　interface　インターフェース名{
　　void　メソッド名()；
}
```

(例)クラスに実装

```
public　class　クラス名　implements　インターフェース名{
　　public　void　インターフェースのメソッド名{
　　　　内容
　　}
 }
```

作成したロボットの説明を次項ページより示す。

- **ロボットの概要**

  弾が当たる度に三角形と四角形の動きを交互に変えるロボット。

  メインロボットクラスの run メソッドではインターフェースを通じて動作を実行。

  敵の弾が当たった際に呼び出されるメソッドで動作の切り替えを行う。

  メインロボットでは、動作のインターフェースを実装したインスタンスを保持しており、このインスタンスは三角形の動作クラスもしくは、四角形の動作クラスのいずれかが入る。

  敵の弾が当たった時に、三角形を保持していたら、四角形を、四角形を保持していたら三角を保持しなおすことで動作の切り替えを行っている。

  ロボットのクラス図を図 4.3.1 に示す

  ロボットの run メソッドの処理フローチャートを次項ページの図 4.3.2 に示す。

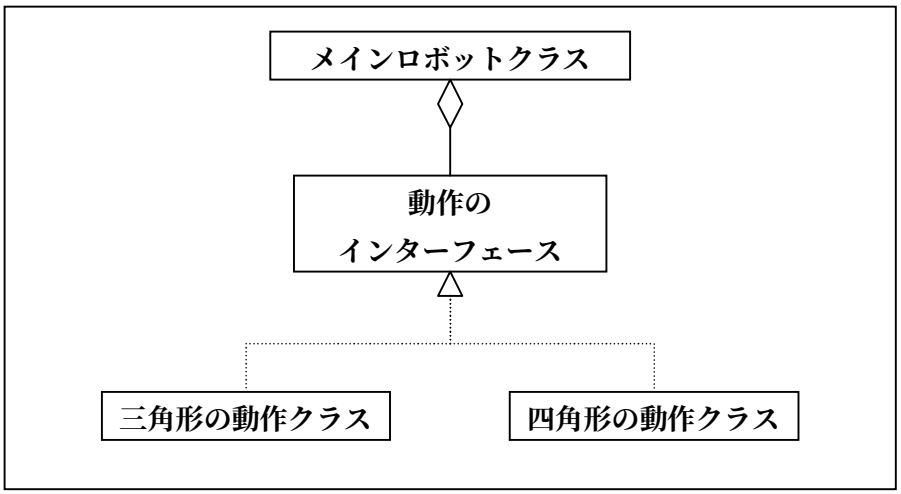

図 4.3.1 ロボットのクラス図

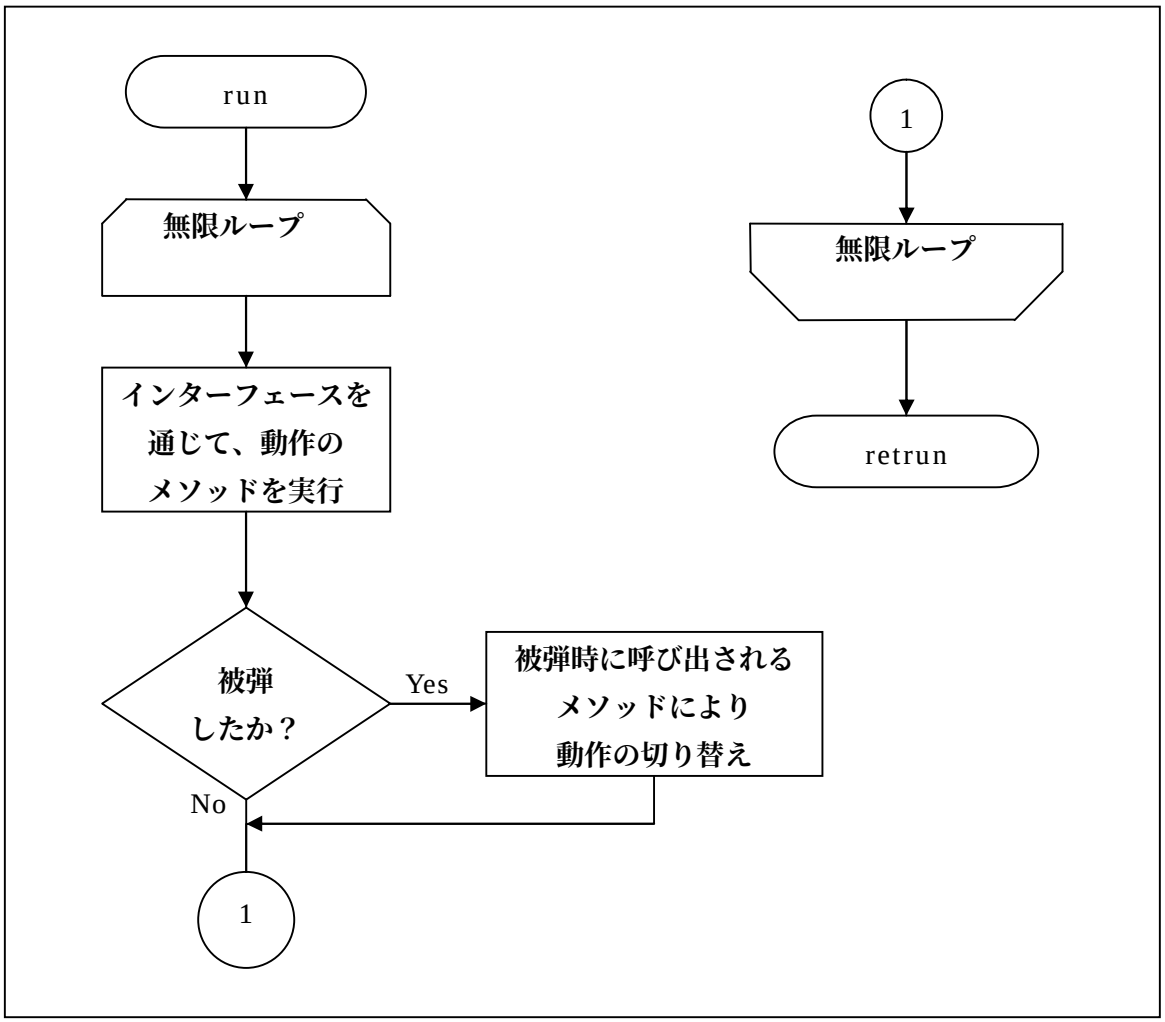

図 4.3.2 ロボットの run メソッドの処理フローチャート

4.4 抽象クラスの習得

抽象クラスについての理解とインターフェースとの相違点について理解した。

・抽象クラスとは

抽象メソッドを持ったクラスは抽象クラスとなり、抽象クラスを利用するのは抽象メソッドを実装したサブクラスをインスタンス化する必要がある、抽象メソッドが一つでもあるクラスはインスタンス化することができない。抽象クラスは複数のクラスを同じ型扱われたいときに有効である。

・インターフェースとの相違点

抽象クラスは中身が定義されたメソッドも持つことができる。
抽象クラスは、サブクラスに共通して必要なインスタンス変数を指定できるが、インターフェースはデータを変更することができない定数として扱われる。

抽象クラスの実装例を以下に示す。

(例)抽象クラスの作成

```
public  abstract  class  抽象クラス名{
        abstract  void  抽象メソッド名();
}
```

(例)サブクラスで実装

```
public  class  サブクラス名  extends  抽象クラス名{
        public  void  抽象メソッド名(){
                内容
        }
}
```

作成したロボットの説明を次項ページより示す。

- **ロボットの概要**

  弾が当たる度に三角形と四角形の動作を交互に変えるロボット。
  メインロボットクラスの run メソッドでオブジェクト生成抽象クラスを通じての動作を実行する。
  敵の弾が当たった際に呼び出されるメソッドでオブジェクト生成抽象クラスを通じて動作の切り替えを行う。
  メインロボットでは、動作のインターフェースを実装したインスタンスを保持しており、このインスタンスは三角形の動作クラスもしくは、四角形の動作クラスのいずれかが入る。敵の弾が当たった時に、三角形を保持していたら、四角形を、四角形を保持していたら三角を保持しなおすことで動作の切り替えを行っている。

  ロボットのクラス図を図 4.4.1 に示す。
  ロボットの run メソッドの処理フローチャートを図 4.4.2 に示す。

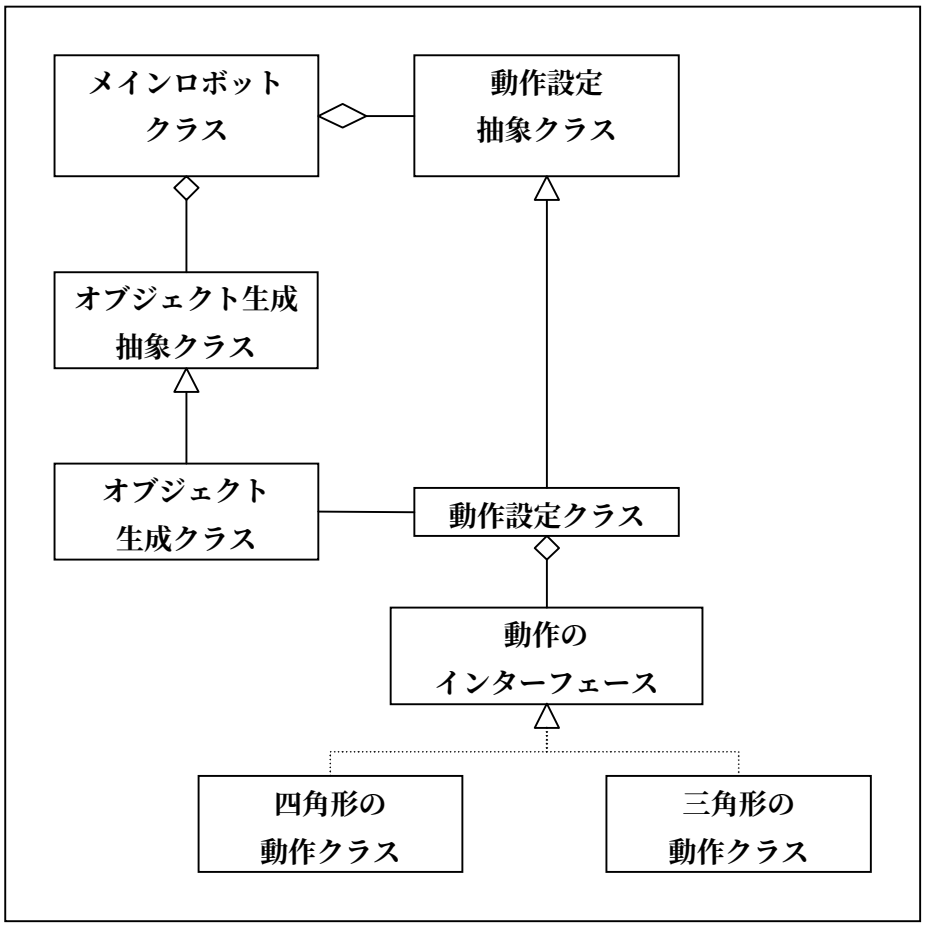

図 4.4.1　ロボットのクラス図

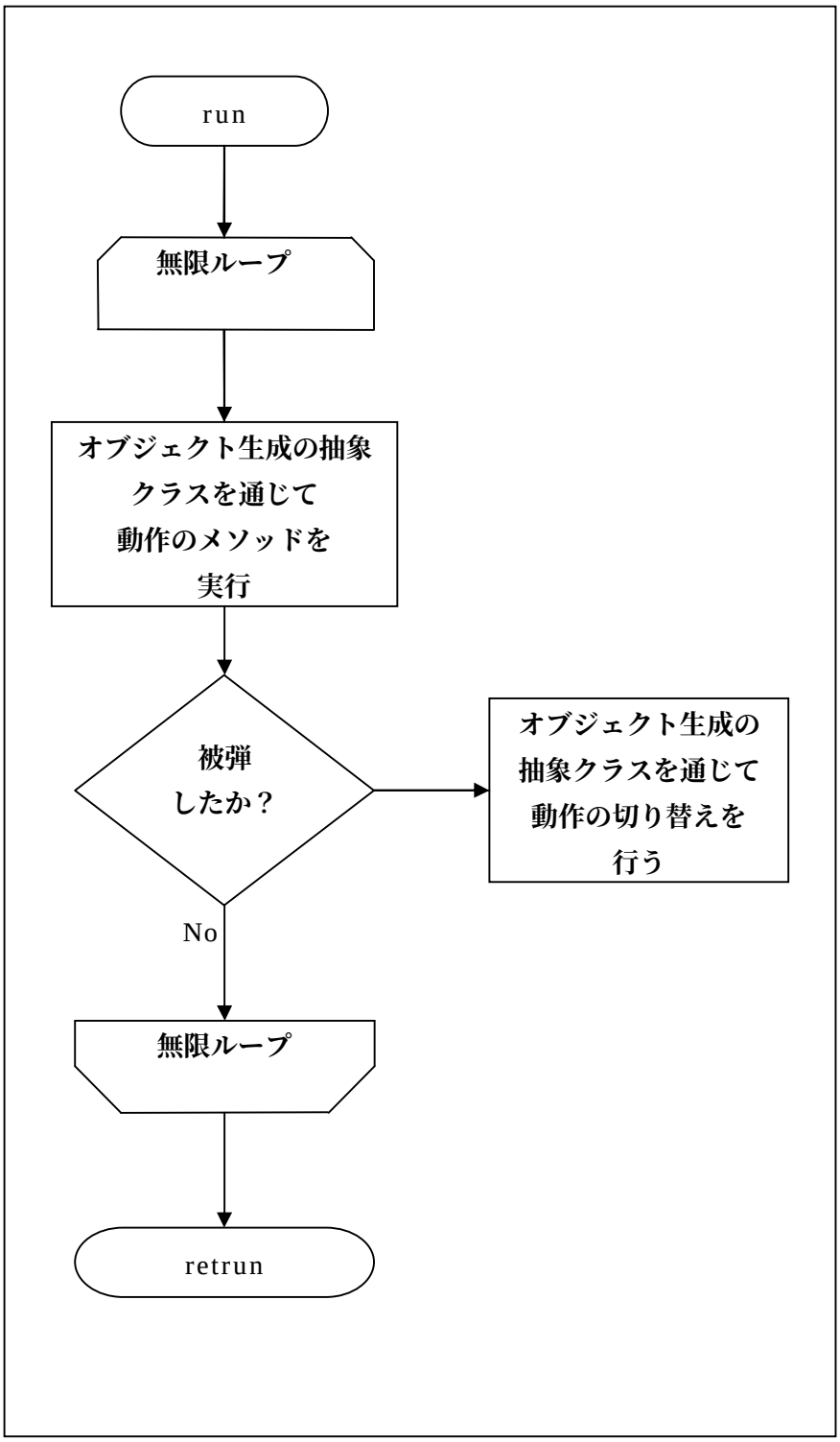

図 4.4.2　ロボットの run メソッドの処理フローチャート

4.5 Java 言語基礎習得状況の確認

どんなロボットを作成するかを自分で決め、また、ロボットをどういった手順で作っていけば完成させることができるかを考えることによって、 プログラムの作成をどういった手順で行っていけばいいかを理解することができた。

作成したロボットの説明を以下より示す。

- ロボットの概要

  ロボットの特性を以下に示す。

  ・壁に沿っての移動とスピンしながらの移動の二つの移動方法を持つ。

  ・敵の弾が当たった際に移動方法を変える。

  ・威力の低い弾で現在の向いている方向に真っ直ぐ移動しながら攻撃と威力の高い弾で敵に向かいながら攻撃の二つの攻撃方法を持つ。

  ・壁に当たった際に攻撃方法を変える。

  ・メインロボットでは、動作のインターフェースを実装したインスタンスを保持しており、このインスタンスはスピン移動クラスもしくは、壁沿い移動のクラスのいずれかが入る。敵の弾が当たった時に、スピンを保持していたら、壁沿いを、壁沿いを保持していたらスピンを保持しなおすことで移動方法の切り替えを行っている。

  ・メインロボットでは、攻撃動作のインターフェースを実装したインスタンスを保持しており、このインスタンスは威力の低い弾で攻撃クラスもしくは、威力の高い弾で攻撃クラスのいずれかが入る。壁にぶつかった時に、威力の低い弾を保持していたら、威力の高い弾を、威力の高い弾を保持していたら威力の低い弾を保持しなおすことで攻撃方法の切り替えを行っている。

ロボットのクラス図を次項ページより図 4.5.1 に示す

ロボットの run メソッドの処理フローチャートを次項ページより図 4.5.2 に示す

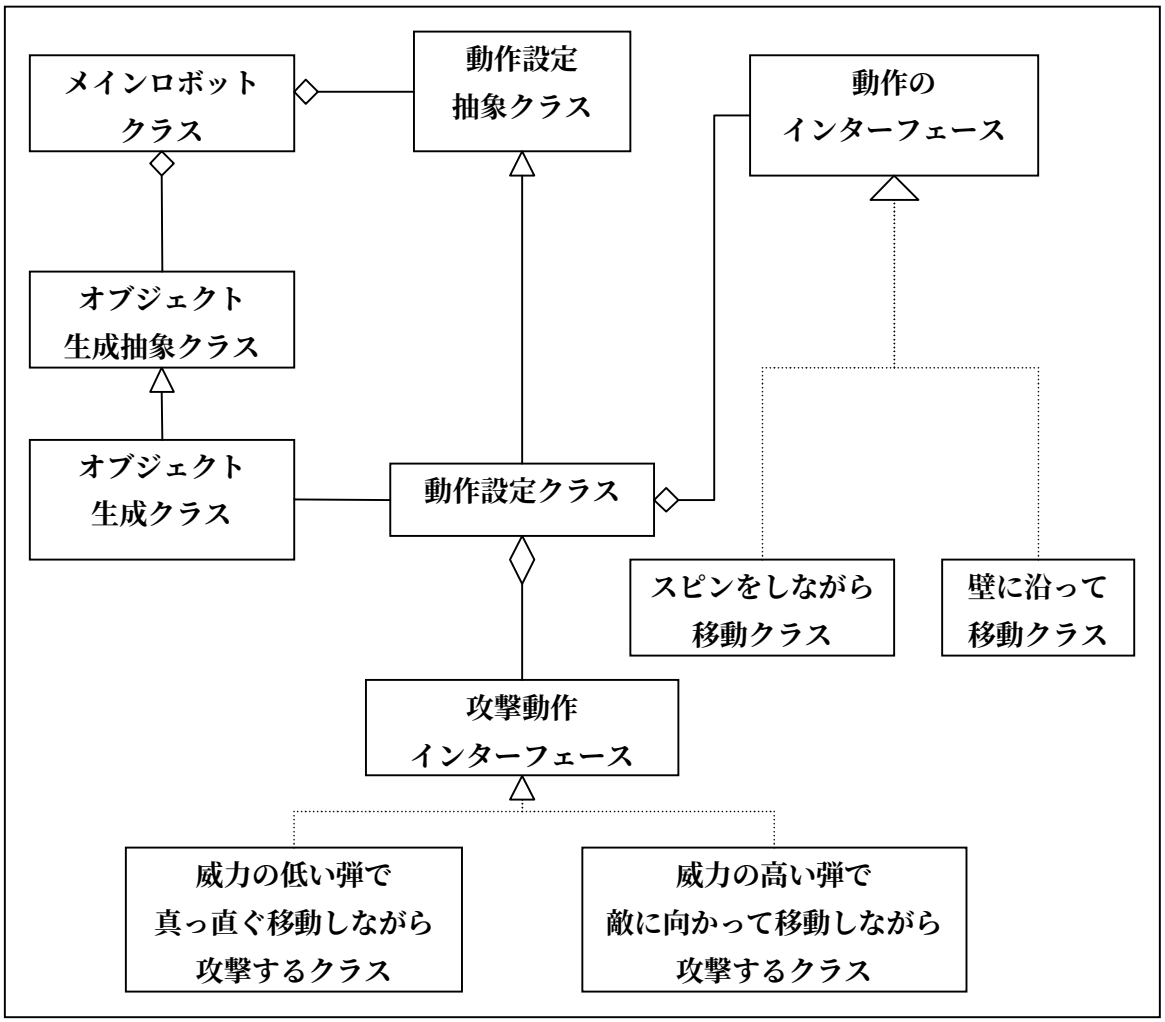

図 4.5.1　ロボットのクラス図

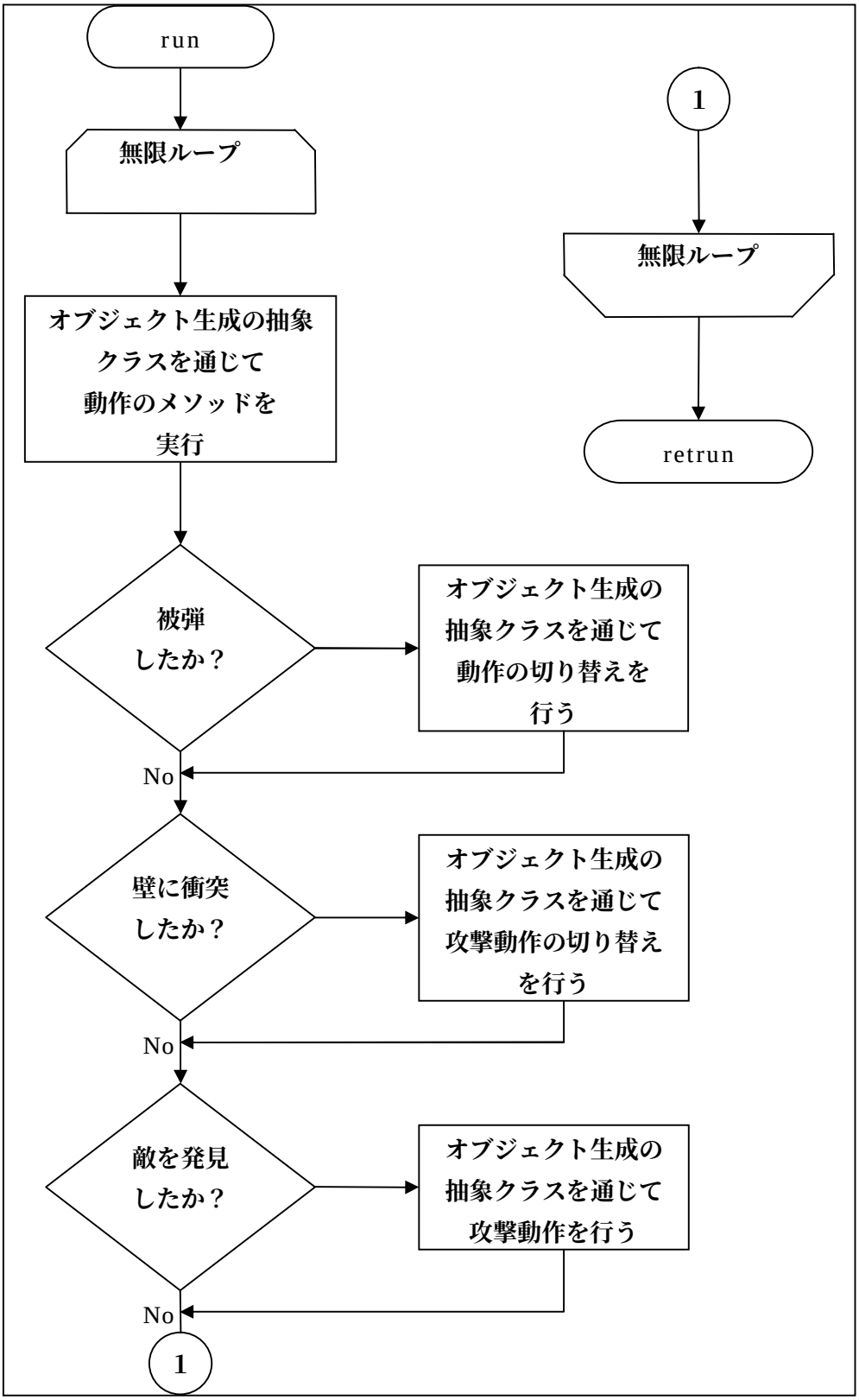

図 4.5.2　ロボットの run メソッドの処理フローチャート

4.6 Web アプリケーションの作成

学校給食会システムを作成していく中で以下の理解ができた。

１．Web アプリケーション開発の流れ。

２．JSP についての理解。

また、データベース、SQL について理解が乏しかったが、マスタ保守作成時の登録、削除、更新作業と、帳票作成(Excel に出力)を作成したことにより以下の理解ができた。

３．データベースの操作に SQL を用いれば良いことを理解した。尚、データベース操作にはデータ登録、更新、削除があること

以下、理解した内容について説明する。

１．Web アプリケーション(struts)開発の流れ(MVC モデル)

・入力、出力画面となる JSP の作成(View)

・JSP で入力されたデータの格納や入力値のチェックを行う From クラスの作成(Date,Document)

・データの登録や削除等を行う Action クラスの作成(Model)

※Control は struts-config-xml で行うため今回は未修得(他メンバが作成担当であったため)。

2. JSP について

Java 言語を利用して Web サーバで動的に Web ページを生成し、クライアントに送信する技術。

struts(前記の Action クラス)でセッションに登録したデータを strut タグで、取り扱うことができることを理解した。

以下、データの取り扱いについて理解した内容を説明する。

・表示方法

画面を表示する前に、あらかじめデータベースから該当するデータを取得し、セッションに登録し、JSP にタグを記述することでデータが表示される。

JSP に表示するまでの流れを以下の図 4.6.1 に示す。

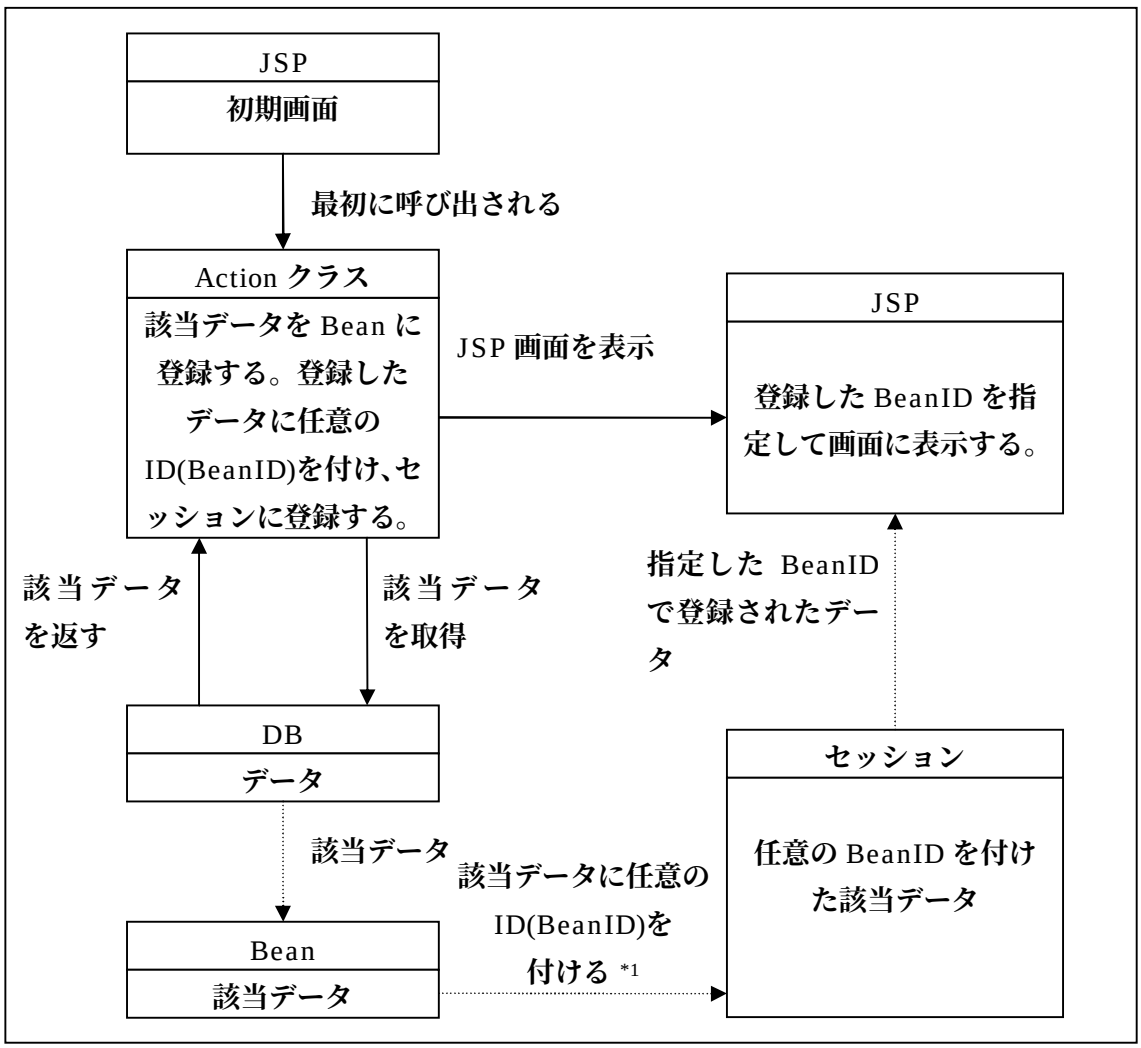

図 4.6.1 データを JSP に表示する流れ

以下に学校給食会システムで使用した JSP にデータを表示させるタグの使用例を示す。

・セレクトボックスで表示する場合

```
<html:form action=”/Action クラス名”>
  <html:select property="メンバ名 " >
  <html:optionsCollection name="BeanID "/>
  </html:select>
</html:form>
```

・テキストボックスで表示する場合

```
<html:form action=”/Action クラス名”>
  <html:text name=”BeanID” property="メンバ名"/ >
</html:form>
```

・画面にそのまま表示する場合

```
<html:form action=”/Action クラス名”>
  <bean:write name=”BeanID” property=”メンバ名”/>
</html:form>
```

セッションの説明を以下に示す。

Web アプリケーションで、各ユーザーの状態を保存し、各ユーザーを区別するための機能。セッション ID というランダムな ID を元にユーザーの識別を行うものである。

セッションへの登録方法(図 4.6.1 の[*1])を以下に示す。

以下の setAttribute メソッドを使ってセッションに登録する。

```
setAttribute("BeanID",データ);
```

結果は通常の HTML 形式となるため、Web ブラウザに特殊な機能を組み込むことなく Web アプリケーションを構築できる。

3. 1 データベース操作

データベース操作の SQL 文を以下に示す。

登録： INSERT INTO テーブル名(カラム名) VALUES(登録データ)

削除： DELETE FROM テーブル名 WHERE 条件

更新： UPDATE テーブル名 SET カラム名 = 更新データ

外部結合： SELECT テーブル名.カラム名,結合テーブル名.結合カラム名
FROM テーブル名 LEFT JOIN 結合テーブル名 ON
テーブル名.カラム名 = 結合テーブル名.結合カラム名

グループ化： SELECT カラム名,集約関数 FROM テーブル名
GROUP BY カラム名

データの取得： SELECT カラム名 From テーブル名

全データの取得： SELECT * From テーブル名

条件を選択して取得： SELECT カラム名 FROM テーブル名
WHERE 条件

3. 2 SQL についての知識

今回、学校給食会システムの開発過程で、実際に上記の SQL 文を用いて、データベースへの操作を行い、データが SQL 文の通りに処理されているのを確認し、SQL の理解を深めることができた。

[5] 結論

学生時代に理解が足りなかった Java 言語の基礎の理解と Web アプリケーション開発に必要な知識を得ることができた。また、より高度なプログラムを組めるよう更なる努力をする。

[6] 参考文献

6.1 参考書

・「やさしい Java　第三版」
・「Java 言語プログラミンレッスン　下」
・「改定版スタートアップオラクルマスターSilver SQL/Oracle 入門」
・「Visal Basic で学ぶ　アルゴリズム入門」

6.2 参考 Web サイト

・コンピュータプログラミング B 冬爪クラス
[http://software.ed.sie.dendai.ac.jp/b/java/wiki.cgi?page=FrontPage#p0]
・TECHSCORE　[http://www.techscore.com/index.html]
・MySQL クイック・リファレンス
[http://www.bitscope.co.jp/tep/MySQL/quickMySQL.html#doc1_id42]
・ASH multimedia lab　[http://ash.jp/]
・Java Drive　[http://www.javadrive.jp/index.html]
・SAK 図書館 [http://homepage2.nifty.com/sak/]
・MySQL のコマンド集
[http://kiyoeri.gotdns.org/~kiyoeri/pukiwiki/?MySQL%A4%CE%A5%B3%A5%DE%A5%F3%A5%C9%BD%B8]
・POI API Documentation [http://poi.apache.org/apidocs/overview-summary.html]
・site cooler [http://www.site-cooler.com/]
・サルでもわかる 逆引きデザインパターン
[http://www.nulab.co.jp/designPatterns/designPatterns1/designPatterns1-1.html]
・矢沢久雄の早わかり GoF デザインパターン
[http://itpro.nikkeibp.co.jp/article/COLUMN/20051123/225074/]
・Robocode(ロボコード)日本語ドキュメント
[http://www.geocities.co.jp/SiliconValley/9155/]
・ロボコード API
[http://www.solar-system.tuis.ac.jp/Java/robocode_api/index.html]