プログラム設計 後期期末試験

準備 プログラムを作る前に、以下の操作をしてファイルの準備をしておくこと。

1. 授業の配付資料を全てダウンロードする場合は、以下を実行する（既に実行済みの場合は不要）
   ```
   $  mygitclone-pd
   ```
2. GitHub から自分のリポジトリを clone しておく（既に実行済みの場合は不要）
   ```
   $ mygitclone4d 自分の GitHub ユーザ名
   ```
3. リポジトリのフォルダに移動して、設定用のスクリプトを実行する
   ```
   $ cd ~/prog4d-ユーザ名
   $ ./myconf
   ```
4. 今回の定期試験用のフォルダをコピーする
   ```
   $ cp  -r  /usr/local/common/kogai/pd/test2term  .  （←ここにピリオド）
   $ cd  test2term       （コピーしたフォルダに移動する）
   $ ls                  （フォルダ内のファイルを確認すると、以下のファイルがコピーされている）
   myanswer.asta
   ```
5. まず、astah を起動してから、コピーしたファイルを開く
   提出する全ての astah ファイルの図内に、**自分の学科の出席番号と氏名**を「ノート」を使って書くこと

問題

PBL 実験で、「教室のゴミ箱の状況を管理してメールで通知する」システムを開発することにした。この開発における設計と実装に関して、以下の **1** ～ **6** の問に答えなさい。

1 開発するシステムのシナリオを考えるために、システムの機能を以下のように整理した。

- 週番は、ゴミ箱の状況をメールで受け取ることができる
- 週番は、ゴミ箱の状況を Web で閲覧することができる
- 学生は、自分のメールアドレスを登録することができる
- 学生は、ゴミ箱の状況を Web で閲覧することができる
- 担任は、ゴミ箱の状況をメールで受け取ることができる
- 担任は、ゴミ箱の状況を Web で閲覧することができる
- 担任は、自分のメールアドレスを登録することができる
- 担任は、週番の情報を編集することができる
- 自分のメールアドレスを登録する、週番の情報を編集する際は、必ずユーザ認証する必要がある

この時、以下の要素をアクタ, ユースケースとしたユースケース図を astah で作成しなさい。

アクタ 学生, 週番, 担任
ユースケース 自分のメールアドレスを登録する, ゴミ箱の状況をメールで受け取る, ゴミ箱の状況を Web で閲覧する, 週番の情報を編集する, ユーザ認証する

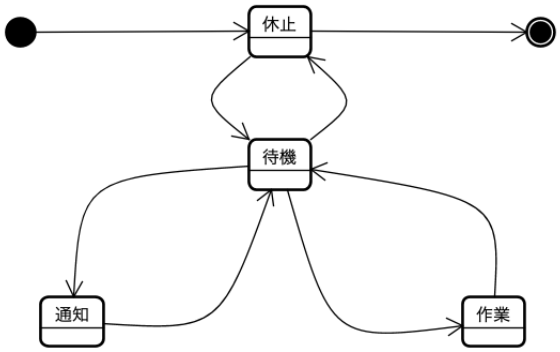
2　「測距センサによってゴミ箱の溜まり具合が変化した際に、ゴミの量に応じて週番または担任にメールで通知して、Web サイトを更新する」部分の機能を以下のような処理でまとめた。

対象者判断　測距センサから距離を受け取ると、送信対象者を「メール送信」へ渡す。
メール送信　「対象者判断」から送信対象者を受け取り、週番リストから週番を読み込み、メールアドレスリストからメールアドレスを読み込み、週番と担任にメールを渡し、状況を「Web 更新」へ渡す。
Web 更新　「メール送信」から状況を受け取り、Web サイトへ状況を渡す。

この時、以下のような要素を使ったデータフロー図を astah で作成しなさい。

外部エンティティ　測距センサ, 週番, 担任, Web サイト
プロセス　対象者判断, メール送信, Web 更新
データストア　週番リスト, メールアドレスリスト
データフロー　シナリオ中の下線部

3　「ゴミ箱の監視状況の変化」を、ステートマシン図で設計することを考える。次のような図のところまで作成した。



この時、上記の図に**以下の要素を追加した**ステートマシン図を astah で完成させなさい。

アクティビティ

- 状態「休止」の内部アクティビティとして、トリガーが「0 時」、ガードが「月曜日」、アクションが「週番更新」を追加
- 状態「通知」の entry アクティビティとして、「Web 更新」を追加
- 状態「通知」の do アクティビティとして、「30 分おきにメールを送る」を追加
- 状態「作業」（ゴミ袋の交換作業）の entry アクティビティとして、「LED 点灯」を追加
- 状態「作業」（ゴミ袋の交換作業）の exit アクティビティとして、「LED 消灯」を追加

遷移

- 状態「休止」から状態「待機」へは、トリガー「8 時」で遷移する
- 状態「待機」から状態「休止」へは、トリガー「19 時」で遷移する
- 状態「待機」から状態「通知」へは、トリガー「センサ変化あり」で遷移する
- 状態「通知」から状態「待機」へは、トリガー「センサ変化あり」で遷移する
- 状態「待機」から状態「作業」へは、トリガー「スイッチを押す」で遷移する
- 状態「作業」から状態「待機」へは、トリガー「スイッチを押す」で遷移する
- 状態「休止」から終了状態へは、トリガー「電源 OFF」で遷移する
- 上記以外の遷移は、トリガー, カードなしで遷移する

4 「測距センサから取得した距離からゴミ袋の状況を判断する」部分のみを、Java のクラス Trash, Controller, Main1 で作成した。(プログラムは問題の後に掲載している。)

astah を使って、クラス Main1 のメソッド main から呼び出されているメソッド changeStatus に関するシーケンス図を完成させなさい。以下の点に注意して作成すること。

- ライフラインは、クラス Main1, Controller, Trash のみとする
- ライフラインのオブジェクト名は必要ない（クラス名のみ）
- メッセージの引数は描く必要はない
- ライフラインの消滅の「×」印は描く必要はない
- 複合フラグメントを使う際は名前は付けなくてもよい（その処理内容がわかる名前を付けてもよい）
- 複合フラグメントのガードは省略せずに描くこと
- 複合フラグメントに含まれるメッセージは、含まれることが分かるように枠内に配置すること

5 「学生と教員のメールアドレスを管理する」部分の一部を、Java のクラス Member, MemberList, Main2 で作成した。(プログラムは問題の後に掲載している。)

astah を使って、クラス Main2 のメソッド main から呼び出されているメソッド clearList に関するシーケンス図を完成させなさい。以下の点に注意して作成すること。

- ライフラインは、クラス Main2, MemberList, Member のみとする
- ライフラインのオブジェクト名は必要ない（クラス名のみ）
- メッセージの引数は描く必要はない
- ライフラインの消滅の「×」印は描く必要はない
- 複合フラグメントを使う際は名前は付けなくてもよい（その処理内容がわかる名前を付けてもよい）
- 複合フラグメントのガードは省略せずに描くこと
- 複合フラグメントに含まれるメッセージは、含まれることが分かるように枠内に配置すること

6 astah を使って、クラス MemberList のメソッド show の処理を表すアクティビティ図を作成しなさい。

問題はここまで

4, **5** 各 20 点　**1**, **2**, **3**, **6** 各 15 点

定期試験の実施について

試験中に使用できるもの

- 筆記用具（メモ用紙は必要な人に配布）
- 演習室のコンピュータ一台（一つの机に一人の配置で、座る場所はどこでもよい）

試験中に参照できるもの

- 自分のホームディレクトリ（ホームフォルダ）以下に保存されているファイル
  (定期試験では紙媒体のものは参照不可)

* **上記以外の情報を参照することは不正行為とする**
(例：USB で接続された機器に保存されているファイルの参照、Web ブラウザやネットワークを介した情報の参照、自分の PC を使用する、など)

* 試験中（開始 5 分後〜開始 60 分後）は、演習室外へのネットワークアクセスは遮断される

* GitHub への提出のためのコマンドに限ってネットワーク利用が可能（それ以外は不正行為とする）

答案の提出

1. 提出する全ての astah ファイルの図内に、**自分の学科の出席番号と氏名**を「ノート」を使って書く
2. 端末内で、以下のコマンドで課題を提出

```
$ git add -A
$ git commit -m "後期期末提出"
$ git push origin master
```

（push が成功すると「done」が含まれたメッセージが表示される）

3. 提出が完了しているかを確認したい人は声をかけて下さい。（その場で教員側の画面で確認します）

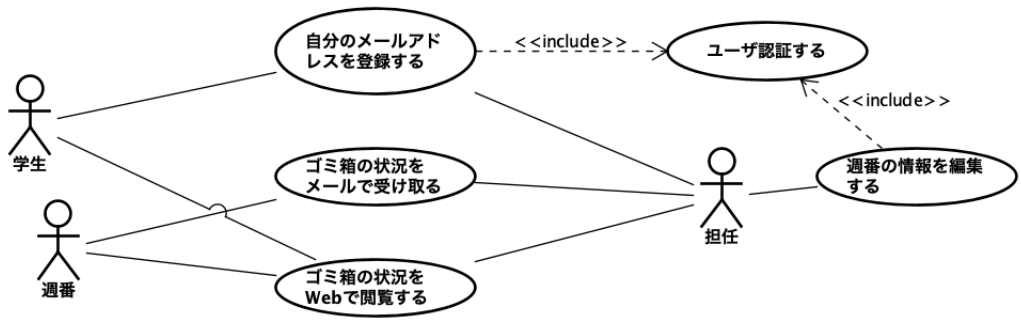
教室のゴミ袋の状況を管理するプログラムの一部

```
class Trash {
    private int depth;
    private String status;
    public int getDepth() {
        return depth;
    }
    public void setDepth(int depth) {
        this.depth = depth;
    }
    public String getStatus() {
        return status;
    }
    public void setStatus(String status) {
        this.status = status;
    }
    public void show() {
        System.out.println("depth: " + depth);
        System.out.println("status: " + status);
    }
}

class MemberList {
    private Member[] list;
    public MemberList() {
        list = new Member[50];
    }
    public void clearList() {
        int i;
        for(i=0; i<list.length; i++) {
            list[i] = new Member();
            list[i].setName("anonymous");
            list[i].setEmail("aaa@bbb.com");
        }
    }
    public void show() {
        int i;
        for(i=0; i<list.length; i++) {
            list[i].show();
        }
    }
}

class Member {
    private String name;
    private String email;
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
    public void show() {
        System.out.println("name: " + name);
        System.out.println("e-mail: " + email);
    }
}

class Controller {
    Trash trash;
    public void changeStatus() {
        int depth;
        depth = trash.getDepth();
        if(depth>=60) {
            trash.setStatus("少");
        }
        if(depth>=30 && depth<60) {
            trash.setStatus("中");
        }
        if(depth<30) {
            trash.setStatus("多");
        }
    }
    public Trash getTrash() {
        return trash;
    }
    public void setTrash(Trash trash) {
        this.trash = trash;
    }
}

class Main1 {
    public static void main(String[] args) {
        Controller c = new Controller();
        Trash t = new Trash();
        c.setTrash(t);
        t.setDepth(30);
        t.show();
        c.changeStatus();
    }
}

class Main2 {
    public static void main(String[] args) {
        MemberList ml;
        ml = new MemberList();
        ml.clearList();
        ml.show();
    }
}
```

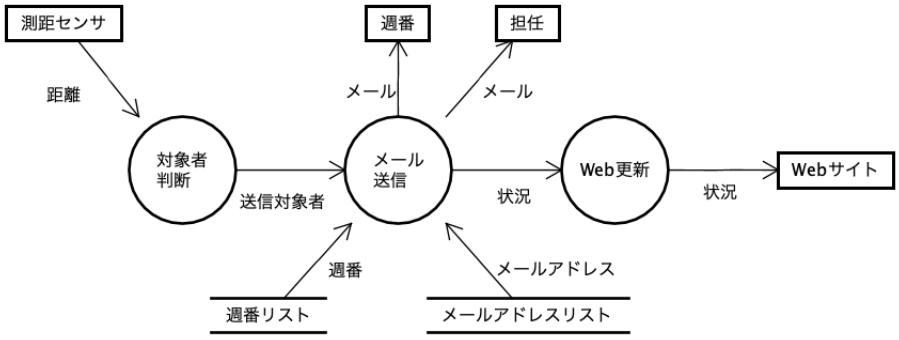
プログラム設計 後期期末試験模範解答（試験時間 90 分，平均 88.1 点）

採点について　不十分な箇所につき −2 点を基本とする。(部分点については各問を参照)
4, 5 各 20 点　1, 2, 3, 6 各 15 点

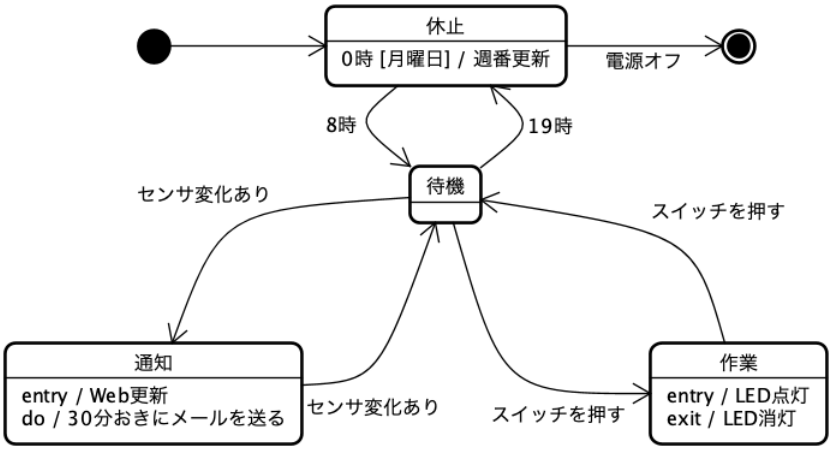
1



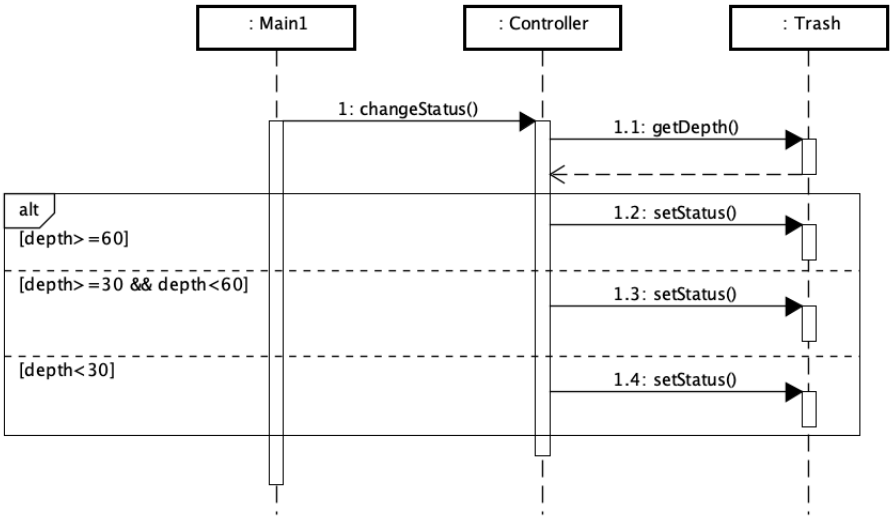
2



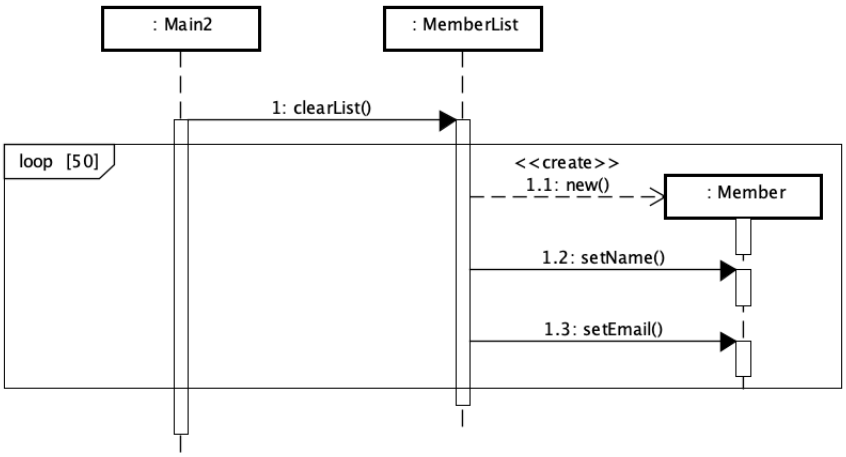
3



4 （ライフライン: 6 点, メッセージ・実行指定: 8 点, 複合フラグメント: 6 点）



5 （ライフライン: 6 点, メッセージ・実行指定: 8 点, 複合フラグメント: 6 点）



6