분석 설계 대상 시스템 (뷔페 POS기)

<20160391, 김민주, '20. 4. 11. (토)>

□ 주요 파일 소개

- admin.java
  주요 Class 및 method를 포함한 java파일로 Pos기 운용에 핵심이 된다.
  File I/o를 통해 menu.txt와 admin_log.txt를 접근하여 필요한 정보를 읽어오거나 기록할 수 있도록 만들었다.

- menu.txt
  뷔페에서 구매할 수 있는 음식 및 음료들을 모아놓은 메뉴 txt파일이다.
  admin.java에서 File I/o를 통해 메뉴들을 admin으로 전송할 수 있고,
  메뉴를 추가할 시 admin.java에서 정보를 받아와 menu.txt에 기록할 수 있다.

- admin_log.txt
  admin.java에서 실행되는 모습들을 기록하는 공간이다.
  admin.java에서 주문 및 계산 등등 여러 함수들이 실행될 때마다 log를
  File I/o를 통해 admin_log.txt에 저장한다.

□ Class 소개

- Customer
  고객이라는 객체의 속성이다.
  sales : 객체가 구매한 메뉴들의 총액
  count_minute : 뷔페에는 이용시간이 있으며
  이용시간이 끝날 때를 알리기 위해 객체마다
  이용시간을 가지고 있어야 한다.
  table_set : table은 총 10자리가 있고
  객체가 어떤 자리에 앉았는지 확인할 수 있도록
  한다.

```
class Customer{
    private int sales;
    private int count_minute;
    private int table_set;

    public Customer(){
        sales = 0;
        count_minute = 40;
        table_set = 0;
    }
    public int get_sales(){
        return sales;
    }
    public void set_sales(int sale){
        sales = sale;
    }
    public int get_table(){
        return table_set;
    }
    public void set_table(int table){
        table_set = table;
    }
    public int get_count_minute(){
        return count_minute;
    }
    public void set_count_minute(int count_minute){
        this.count_minute = count_minute;
    }
```

```
public void countdown(){
    Timer m_timer[] = new Timer[10];
    for(int i = 0; i < 10; i++){
        m_timer[i] = new Timer();
    }
    TimerTask m_task = new TimerTask(){
        public void run(){//나중에 고객 클래스에서 count_minute인자로 받아서 써야할듯...?
            if(count_minute > 0){
                if(count_minute % 10 ==0){//매초마다 초기화 하지만 표시는 10초에 한번씩 갱신, 그리고 calculate할 경우 이용시간 0이 바로 갱신될 수 있도록
                System.out.println(table_set +"고객의 이용시간이"+count_minute + "분 남았습니다.");
                }
                count_minute = count_minute-1;
            }else{
                System.out.println(table_set +"고객의 이용시간이 종료되었습니다.");
                m_timer[table_set].cancel();
            }
        }
    };
    m_timer[table_set].schedule(m_task,0,1000);
}
```

public void countdown() : 카운트다운을 할 수 있는 타이머 세팅이다.
기존 countdown의 목표는 분당 리셋될 수 있도록 제작하려 했지만
현재는 빠른 확인과 문제가 없는지를 확인하기 위해 1초당 reset되도록 고쳐놨고,
출력은 10초에 한번씩 알림이 될 수 있도록 만들었다.

- admin

admin.java의 메인 Class로써 admin.java에서 사용되는 거의 모든 함수들이
admin Class에 포함되어있다.
또한 main함수도 admin Class에 포함되어 있다.

□ 함수 소개

main함수 :

시스템 구동을 위한 가장 중요한 함수.

Customer class를 사용하여 테이블의 개수인 10개 크기의 class 배열을 만들고 초기화시켰다. hm이라는 String 변수를 만들어 종업원이 사용할 메뉴를 hm에 입력하면 메뉴들과 hm을 비교하면서 종업원이 원하는 메뉴를 실행할 수 있도록 한다.

```
public static void main(String[] args){
    init();
    Customer[] people = new Customer[10];

    for(int i = 0; i<10; i++){
        people[i] = new Customer();
        people[i].set_table(i);
    }//객체를 초기화 - 초기화 안해서 널포인트 에러 떴었음 신기하네

    String hm;
    int num = 0;
    int price = 0;
    String total_sale_log = "";
    while(true){
        //몇번 손님인지 확인해야함
        System.out.println("메뉴선택\norder    mor
        hm = sc.nextLine();
        if(hm.equals("order")){
            System.out.println("몇번손님(0~9)");
            num = sc.nextInt();
            sc.nextLine();
            people[num] = new Customer();
            people[num].set_table(num);
            order(people[num]);
            people[num].countdown();//order은 첫
        }else if(hm.equals("more_order")){
            System.out.println("몇번손님(0~9)");
            num = sc.nextInt();
            sc.nextLine();
            more_order(people[num]);
        }else if(hm.equals("calculate")){
            System.out.println("몇번손님(0~9)");
            num = sc.nextInt();
            sc.nextLine();
            System.out.print("고객이 지불한 금액 : ");
            price = sc.nextInt();
            sc.nextLine();
            calculate(people[num], price);//sale
            people[num].set_count_minute(0);//계
            //객체제거도 해야할거 같은데
        }else if(hm.equals("add_menu")){
            add_menu();
        }else if(hm.equals("sub_menu")){
            sub_menu();
        }else if(hm.equals("total_sale")){
            System.out.println("총 매출 : " + total_
            total_sale_log = "(total_sale)총 매출 :
            admin_log(total_sale_log);
        }else if(hm.equals("exit")){
            break;
        }else{
            System.out.println("오타확인 다시 입력해주십
        }
    }
    System.out.println("시스템이 종료되었습니다.");
    String logout = "";
    logout = "시스템이 종료되었습니다.\n";
    admin_log(logout);
```

order(), more_order()

음식을 주문한다는 것에 두 함수의 기능은 거의 같지만 다른 점이 있다.

order()함수는 고객이 첫 주문을 할 때만 사용되며 order()함수가 실행되는 즉시 해당 고객의 카운트다운이 시작된다. 그리고 order()함수를 실행하기 전 객체를 초기화해주면서 객체를 재사용 할 수 있도록 한다.

두 함수 모두 실행하기 전 어떤 테이블의 주문인지 확인하여 주문을 실행하고 주문을 받으면서 어떤 메뉴를 주문하였는지, 그리고 그 메뉴의 가격이 얼마인지를 확인할 수 있다.

그리고 주문을 마치면 주문한 메뉴들의 총 금액이 나오며, 이는 고객 객체마다 존재하는 sales에 저장된다.

```java
public static void order(Customer people){
    String order_log = "";
    while(true){
        System.out.println("무엇을 주문하시겠습니까");
        order_menu = sc.nextLine();

        if(order_menu.equals("exit")){
            break;
        }else if(menu.containsKey(order_menu) == false){
            System.out.println("없는 메뉴입니다 다시 시도하십시오\n");
            continue;
        }else{
            output(menu, order_menu);
            people.set_sales(people.get_sales() + menu.get(order_menu));
            order_log = "(order)"+people.get_table() + "번 고객이 " + order_menu + "를 주문하였습니다.(첫주문)\n";
            admin_log(order_log);
        }
    }
    System.out.println("총 금액 : " + people.get_sales() +"원\n");//sales를 고객 객체마다 1개씩
    //카운트다운 넣어야합니다. countdown(people); 객체로 받은 후
}
//처음주문
public static void more_order(Customer people){
    String more_order_log = "";
    System.out.println("추가주문 입니다.");
    while(true){
        System.out.println("무엇을 주문하시겠습니까");
        order_menu = sc.nextLine();

        if(order_menu.equals("exit")){
            break;
        }else if(menu.containsKey(order_menu) == false){
            System.out.println("없는 메뉴입니다 다시 시도하십시오");
            continue;
        }else{
            output(menu, order_menu);
            people.set_sales(people.get_sales() + menu.get(order_menu));
            more_order_log = "(more_order)"+people.get_table() + "번 고객이 " + order_menu + "를 주문하였습니다.\n";
            admin_log(more_order_log);
        }
    }
    System.out.println("총 금액 : " + people.get_sales()+"원\n");//sales를 고객 객체마다 1개씩
}
```

calculate()

계산하는 함수로써 고객마다 가지고있는 sales정보를 가져와 결제하는 시스템이다.
인자로는 어떤 객체인지 정보를 받고 그 객체에서 지불해야할 금액을 가져온다.
그리고 고객인 지불하는 금액은 int형으로 받아와 지불하는 금액에서 지불해야할 금액의 차액을 출력하여 종업원에게 보여준다.

```
public static void calculate(Customer people, int money){
    if(people.get_sales() > money){
        System.out.println("금액이 모자랍니다.");//돈이 모자랄때도 생각해야함
    }else{
        System.out.println("고객에게 " + money + "원을 받고 " + (money-people.get_sales()) + "원을 거슬러주었습니다.");
    }
    total_sales = total_sales + people.get_sales();

    String calculate_log = "";
    calculate_log = "(calculate)고객에게 " + money + "원을 받고 " + (money-people.get_sales()) + "원을 거슬러주었습니다.\n";
    admin_log(calculate_log);
    //계산하면 고객의 sales를 0으로 초기화해주고, countdown도 120으로 초기화해준다.
}
//계산
```

- add_menu(), sub_menu()

메뉴는 Hashmap의 형태로 key값에는 메뉴의 이름이, value값에는 그 메뉴의 금액이 저장되어 있다.
이것을 활용하여 add_menu에서는 추가할 메뉴를 사용자에게 입력받아 Hashmap에 추가하고 그 정보를 menu.txt에 저장할 수 있도록 한다.
sub_menu또한 Hashmap에서 제거해야할 메뉴를 찾은 후 해당메뉴를 map에서 제외시킨다. 그리고 menu.txt에서도 해당 메뉴를 제거해야하지만 그 부분만은
구현이 되어있지 않은 상태이다.

```
public static void add_menu(){
    System.out.println("어떤 메뉴를 추가하시겠습니까?");
    add_menu = sc.nextLine();
    System.out.println("가격은 얼마입니까?");
    menu_price = sc.nextInt();
    menu.put(add_menu, menu_price);
    sc.nextLine();

    File file = new File("./menu.txt");
    FileWriter writer = null;
    try{
        writer = new FileWriter(file, true);
        writer.write("\n" + add_menu + "\n" + menu_price);
        writer.flush();
    }catch(IOException e){
        e.printStackTrace();
    }finally{
        try{
            if(writer != null){
                writer.close();
            }
        }catch(IOException e){
            e.printStackTrace();
        }
    }
    String add_menu_log = "";
    add_menu_log = "(add_menu)"+add_menu + "를 메뉴에 추가하셨습니다.\n ";
    admin_log(add_menu_log);
}
//메뉴추가
```

```
public static void sub_menu(){//txt파일에서도 제거해야하는데 못함
    System.out.println("어떤 메뉴를 제거하시겠습니까?");
    sub_menu = sc.nextLine();
    menu.remove(sub_menu);
    String sub_menu_log = "";
    sub_menu_log ="(sub_menu)"+ sub_menu + "를 메뉴에서 제거하셨습니다.\n ";
    admin_log(sub_menu_log);
}
//메뉴제거
```

- total_sale
  총 매출액을 표시한다.
  main함수의 while문에서 메뉴선택을 통해 total_sale을 확인할 수 있다.

- admin_log
  실행되는 거의 모든 작업들을 admin_log.txt에 기록한다.
  String변수를 통해 저장할 문자들을 인자로 받아와 File I/o를 통해 admin_log.txt에 저장하는 형식이다.

- menu_find()
  메뉴판이고 menu.txt를 File I/o로 불러와 출력한다.

```
public static void admin_log(String a){
    File file = new File("./admin_log.txt");
    FileWriter writer = null;
    try{
        writer = new FileWriter(file, true);
        writer.write(a);
        writer.flush();
    }catch(IOException e){
        e.printStackTrace();
    }finally{
        try{
            if(writer != null){
                writer.close();
            }
        }catch(IOException e){
            e.printStackTrace();
        }
    }
}
```

```
public static void menu_find(){
    int count = 0;
    try{
        File file = new File("./menu.txt");
        FileReader filereader = new FileReader(file);
        BufferedReader bufReader = new BufferedReader(filereader);
        String line = "";
        while((line = bufReader.readLine()) != null){
            if(count %2 == 0){
                System.out.print(line + " : ");
            }else{
                System.out.println(line+"\n");
            }
            ++count;
        }
        bufReader.close();
    }catch (FileNotFoundException e) {
        // TODO: handle exception
    }catch(IOException e){
        System.out.println(e);
    }
}
```

□ 부족한 점

- 최종 프로젝트를 위해 구현해야할 Gui시스템과 메뉴별 매출, 그리고 sub_menu()함수에서 File I/o를 통해 menu.txt파일에 접근해 해당 메뉴를 제거하는 기능을 구현해야 할 것이다.