Assignment #1

2019006171 김은민

Activity.java

```java
public class Activity {

    private String name;
    private String location;
    private int price;

    // constructor
    public Activity(String name, String location, int price) {
        this.name = name;
        this.location = location;
        this.price = price;
    }

    // copy constructor
    public Activity(Activity activity) {
        this.name = activity.name;
        this.location = activity.location;
        this.price = activity.price;
    }

    // getter, setter
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getLocation() {
        return location;
    }
    public void setLocation(String location) {
        this.location = location;
    }
    public int getPrice() {
        return price;
    }
    public void setPrice(int price) {
        this.price = price;
    }

    // toString method
    public String toString() {
        return name + "(" + location + ", " + price + " won)";
    }

    // equals method
    public Boolean equals(Activity activity) {
        if (activity == null) return false;
        return name == activity.name
                && location == activity.location
                && price == activity.price;
    }
```

Schedule.java

```java
public class Schedule {
    private String name;
    private int days;
    private Activity[][] plan;
    private int expense;
    public static int scheduleNum = 0;

    // constructor
    public Schedule(String name, int days) {
        this.name = name;
        this.days = days;
        this.plan = new Activity[12][days];
        scheduleNum ++;
    }
```

// 행 개수를 시간대의 개수인 12개,
열 개수를 입력 받은 여행 일수로 하는 배열
// 총 스케줄 개수가 1 증가

Schedule.java

```
// copy constructor
public Schedule(Schedule schedule) {
    this.name = schedule.name;
    this.days = schedule.days;
    this.plan = new Activity[12][days];

    for (int i=0;i<12;i++) {
        for (int j=0;j<days;j++) {
            if (schedule.plan[i][j] == null)
                this.plan[i][j] = null;
            else
                this.plan[i][j] = new Activity(schedule.plan[i][j]);
        }
    }
    this.expense = schedule.expense;
    scheduleNum ++;
}

// getter, setter
public String getName() {
    return name;
}
public void setName(String name) {
    this.name = name;
}
```

// 복사할 스케줄의 plan의 각 날짜와 시간대를 전부 돌면서 복사

// 복사할 스케줄의 특정 날짜와 시간대의 activity가 비었다면 복사 받을 스케줄의 해당 날짜 시간대도 비어 있게 한다

// activity가 있는 날짜 시간대의 activity를 deep copy한다

// 총 스케줄 개수가 1 증가

Schedule.java

```java
public void addActivity(int day, int time, Activity activity) {     // 스케줄에 activity를 추가하는 메소드

    if (checkIfInPlan(activity) == true)
        System.out.println("Fail to add activity");    // 추가하려는 activity가 스케줄에 이미 있으면 추가하지 않는다

    else if (plan[time-9][day-1] != null)
        System.out.println("Fail to add activity");    // activity를 추가하려는 날짜와 시간대가 비어 있지 않으면
    else {                                                추가하지 않는다
        plan[time-9][day-1] = new Activity(activity);
        expense += activity.getPrice();                // Activity의 copy constructor를 사용해 deep copy 한다
    }
}
```

```java
public Boolean checkIfInPlan(Activity activity) {   // 추가하려는 activity가 스케줄에 이미 있는지 확인하는 메소드
    for (int i=0;i<12;i++) {
        for (int j=0;j<days;j++) {
            if (activity.equals(plan[i][j]))
                return true;
        }
    }
    return false;
}
```

// activity를 추가하려는 스케줄의 각 날짜와 시간대를 모두 돌면서 추가하려는 activity와 같은 activity가 있는지 확인한다

// 추가하려는 activity와 동일한 activity가 없음

Schedule.java

```java
public void printActivity() {
    System.out.println("----------------".repeat(6));
    System.out.printf("%-16s","");
    for (int i=0;i<days;i++)
        System.out.printf("%-16s","Day " + (int)(i+1));
    System.out.println();
    for (int i=0; i<12;i++) {
        System.out.printf("%-16s",i+9 +":00");
        for (int j=0;j<days;j++) {
            if (plan[i][j] == null)
                System.out.printf("%-16s","----");
            else
                System.out.printf("%-16s",plan[i][j].getName());

        }
        System.out.println();
    }
    System.out.println("----------------".repeat(6));
    System.out.println("Total expenses : " + expense);
    System.out.println("----------------".repeat(6));
}

public void deleteActivity(int day, int time) {
    expense -= plan[time-9][day-1].getPrice();
    plan[time-9][day-1] = null;
    System.out.println("Removed successfully");
}
```

// 스케줄의 모든 일정을 출력하는 메소드

// 출력할 스케줄의 여행일수만큼 1행의 날짜 출력

// 1열(시간대) 출력

// 해당 날짜 시간대에 activity가 없을 때 출력

// activity가 있으면 activity의 이름 출력

// 스케줄의 총 지출 출력

// 스케줄에서 activity를 삭제하는 메소드
// 지울 activity의 지출만큼 전체 지출에서 뺀다
// activity가 있던 날짜 시간대를 비운다

TravelScheduler.java

```
public class TravelScheduler {
    public static void main (String[] args) {

        Schedule[] scheduleList = new Schedule[5];                    // 최대 스케줄 5개를 저장하는 배열
        Activity[] activityList = new Activity[8];

        activityList[0] = new Activity("Hiking", "Mountain", 0);
        activityList[1] = new Activity("Horse Riding", "Hill", 3000);
        activityList[2] = new Activity("Visiting Museum", "Museum", 8000);
        activityList[3] = new Activity("Watching movie", "Theater", 11000);
        activityList[4] = new Activity("Fishing", "Sea", 15000);
        activityList[5] = new Activity("Surfing", "Beach", 20000);
        activityList[6] = new Activity("Camping", "Field", 30000);
        activityList[7] = new Activity("Paragliding", "Mountain", 50000);


        while (true) {                                                // 첫번째 선택지를 무한으로 반복한다
            System.out.println("1) Select schedule");
            System.out.println("2) Edit schedule");
            System.out.println("3) End Program");
            System.out.print("Select menu : ");
            Scanner scan = new Scanner(System.in);
            int choice = scan.nextInt();
            String dump = scan.nextLine();
```

TravelScheduler.java

```
switch (choice) {

// Select Schedule
case 1:
    for (int i=0;i<5;i++) {
        if (scheduleList[i] == null)
            System.out.println(i+1 + ") EMPTY SCHEDULE");
        else
            System.out.println(i+1 + ") " + scheduleList[i].getName());
    }

    System.out.print("Select a schedule : ");
    int scheduleChoice = scan.nextInt();
    dump = scan.nextLine();

    if (scheduleChoice == 0) break:
    else if (scheduleList[scheduleChoice-1] != null) {
        while(true) {
        System.out.println("1) Add Activity");
        System.out.println("2) Remove Activity");
        System.out.println("3) Print Schedule");
        System.out.print("Select menu : ");
        choice = scan.nextInt();
        dump = scan.nextLine();

        if (choice == 0) break;
```

// CASE 1 :
첫번째 선택지에서 1) Select schedule을 선택할 경우

// 모든 스케줄 리스트를 출력

// 조작할 schedule을 선택

// 0을 선택하면 첫번째 선택지로 돌아간다

// 선택한 스케줄이 empty schedule 이 아닐 경우 두번째 선택지를 반복

// 선택할 스케줄을 어떻게 할 건지 선택

// 0을 선택하면 두번째 선택지로 돌아간다

TravelScheduler.java

// CASE 1 :
첫번째 선택지에서 1) Select schedule을 선택할 경우

// 두번째 선택지에서 1) Add activity를 선택할 경우

```
// Add Activity
else if (choice == 1) {
    // activity list 출력
    for (int i=0;i<8;i++)
        System.out.println(i+1 + ") " + activityList[i].toString());          // 모든 activity 리스트를 출력

    System.out.print("Select activity to do : ");
    int activityChoice = scan.nextInt();
    dump = scan.nextLine();
    System.out.print("Enter the day to do activity : ");
    int day = scan.nextInt();
    dump = scan.nextLine();
    System.out.print("Enter the time to do activity(9~20): ");          // 추가할 activity와 날짜 시간대를 선택
    int time = scan.nextInt();
    dump = scan.nextLine();

    scheduleList[scheduleChoice-1].addActivity(day, time, activityList[activityChoice-1]);
                                                                        // 현재 스케줄의 addActivity 메소드를 invoke
}
```

TravelScheduler.java

// CASE 1 :
첫번째 선택지에서 1) Select schedule을 선택할 경우

// 두번째 선택지에서 2) Remove activity 를 선택할 경우

```
    // Remove Activity
    else if (choice == 2) {
        scheduleList[scheduleChoice-1].printActivity();
        System.out.print("Enter the day to remove activity : ");
        int day = scan.nextInt();
        dump = scan.nextLine();
        System.out.print("Enter the time to remove activity : ");
        int time = scan.nextInt();
        dump = scan.nextLine();

        scheduleList[scheduleChoice-1].deleteActivity(day,time);
    }
    // Print schedule
    else
        scheduleList[scheduleChoice-1].printActivity();

    }

}
else break;
break;
```

// 스케줄의 모든 일정을 출력

// 삭제할 activity가 있는 날짜 시간대 선택

// 현재 스케줄의 deleteActivity 메소드 invoke

// 두번째 선택지에서 3) Print schedule 를 선택할 경우

// 현재 스케줄의 printActivity 메소드 invoke

// 선택한 스케줄이 empty schedule 일 경우 첫번째 선택지로 돌아간다

TravelScheduler.java

```
// Edit Schedule
case 2:
    while(true) {
    System.out.println("1) Make a new schedule");
    System.out.println("2) Copy an existing schedule");
    System.out.print("Select menu : ");
    choice = scan.nextInt();
    dump = scan.nextLine();

    if (choice == 0) break;

    // Make a new schedule
    else if (choice == 1) {
        System.out.print("Enter a name for the schedule : ");
        String name = scan.nextLine();
        System.out.print("Enter travel days : ");
        int travelDays = scan.nextInt();
        dump = scan.nextLine();

        scheduleList[Schedule.scheduleNum] = new Schedule(name, travelDays);
    }
```

// CASE 2 :
첫번째 선택지에서 2) Edit schedule을 선택할 경우

// 두번째 선택지를 반복

// 0을 입력한 경우 첫번째 선택지로 돌아간다

// 두번째 선택지에서 1) Make a new schedule 을 선택할 경우

// 추가할 스케줄 이름과 여행일수 입력

// 스케줄 리스트에 새 스케줄 추가

TravelScheduler.java

// CASE 2 :
첫번째 선택지에서 2) Edit schedule을 선택할 경우
// 두번째 선택지에서 2) Copy an existing schedule 을 선택할 경우

```
// Copy an existing schedule
else {
    for (int i=0;i<5;i++) {
        if (scheduleList[i] == null)
            System.out.println(i+1 + ") EMPTY SCHEDULE");
        else
            System.out.println(i+1 + ") " + scheduleList[i].getName());   // 모든 스케줄 리스트 출력
    }
    System.out.print("Select the schedule to copy : ");
    choice = scan.nextInt();
    dump = scan.nextLine();
    System.out.print("Enter a schedule name : ");                          // 복사할 스케줄 선택
    String name = scan.nextLine();

    scheduleList[Schedule.scheduleNum] = new Schedule(scheduleList[choice-1]);  // 스케줄 리스트에 새 스케줄 추가
    scheduleList[Schedule.scheduleNum-1].setName(name);                  // 추가된 스케줄의 이름을 입력한 이름으로 변경
}
}
break;
```

TravelScheduler.java

```
// End Program
case 3:
    System.exit(0);

}
```

// CASE 3 : 첫번째 선택지에서 3) End Program 을 선택할 경우