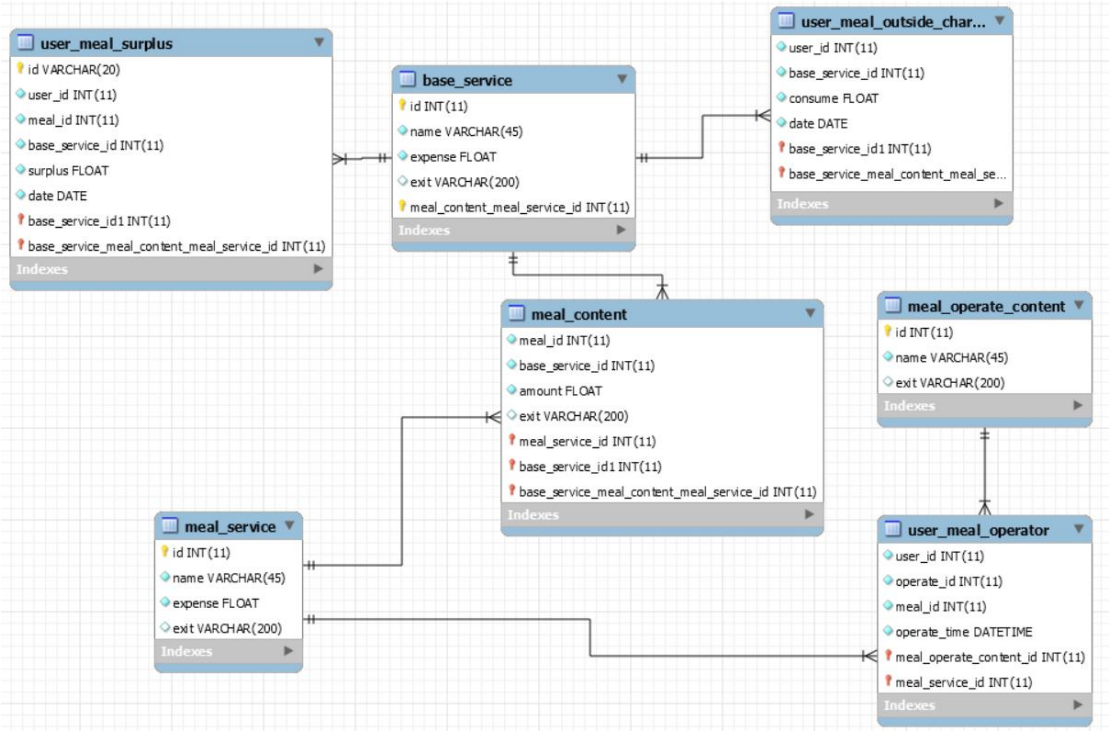
（1）：数据库的 ER 图：



（2）：操作的设计：

操作 1：

设计：

//对套餐的订购以及退订情况进行查询；

inquiryMeals(int userId, String date);

输入用户名和想要查询的套餐操作日期，就会看到历史记录，如果那天没有对套餐的操作，那么就没有关于套餐的输出；

从数据库获取数据，然后将数据分类打印；

运行截图：

//参数可以修改

```
//对套餐的订购以及退订情况进行查询；
//查询用户1在2018-01-01对套餐的操作（可以查询别的日期）
inquiryMeals( userId: 1,  date: "2018-01-01");
```

//输出的结果

```
对1进行套餐的查询开始的时间：  2018-10-29 18:08:47
用户1 于 2018-01-01 10:10:10.0 日，订购套餐 1
对1进行套餐的查询结束的时间：  2018-10-29 18:08:48
                                                    ;
```

对应操作所用的时间：

//打印了链接数据库之前的时间和 close 链接之后的时间

```
对1进行套餐的查询开始的时间：  2018-10-29 18:08:47
用户1 于 2018-01-01 10:10:10.0 日，订购套餐 1
对1进行套餐的查询结束的时间：  2018-10-29 18:08:48
                                                    ;
```

除代码之外其他需要说明的东西：

无；

操作 2：

设计：

//订购套餐；

orderMeals(int userId, int mealId);

输入用户 id 和想要订购的套餐的 id 号，就可以查询到套餐的信息；

往 table “user_meal_surplus”（存储订购套餐之后，可以使用的套餐内的基准服务的类型和数量）中插入对应数据，插入数据的基准服务 id 以及 surplus（基准服务剩余的数量）通过 mealId 从 table “meal_content”（套餐的 id，名称，基准服务内容和数量等等）中获得；

运行截图：

//方法导调用

```
//订购套餐；
//id为1的用户订购套餐4
orderMeals( userId: 1,  mealId: 4);
```

//控制台输出

```
对用户1进行套餐的订购开始的时间：  2018-10-29 18:21:26
对用户1进行套餐的订购结束的时间：  2018-10-29 18:21:27
```

//为 id 为 1 的用户添加了刚刚订购的套餐 4

Result Grid | Filter Rows: | Edit: | Exp

id	user_id	meal_id	base_se	surplus	date
20181010101010101010	1	1	1	90	2018-10-10
2018929182127258	1	4	4	2048	2018-10-29
NULL	NULL	NULL	NULL	NULL	NULL

//记录对套餐的操作---订购套餐的 operate_id 为 1

user_id	operate_id	meal_id	operate_time
1	1	1	2018-01-01 10:10:10
1	1	4	2018-10-29 18:21:27

；

对应操作所用的时间：

```
对用户1进行套餐的订购开始的时间：  2018-10-29 18:21:26
对用户1进行套餐的订购结束的时间：  2018-10-29 18:21:27
```

；

除代码之外其他需要说明的东西：

1：table “user_meal_surplus”的 id 号是当前的时间（含毫秒）的，数据类型为 varchar(20)；

操作 3：

设计：

//退订---立即生效；

orderMeals(int userId, int mealId);

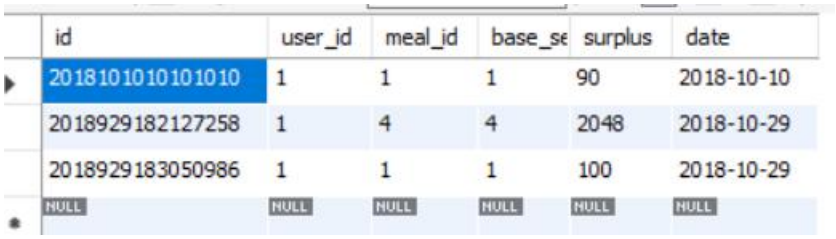
输入用户 id 和想要立即退订的套餐的 id，就会将 table “user_meal_surplus”中对应用户的套餐删除，然后将已经花费了的套餐内容存入 table “user_meal_outside_charge”，并且在 table

“user_meal_operator”中存储用户退订套餐的记录；

运行截图：

//先调用订购套餐的方法，orderMeals(1, 1);为用户 id 为 1 的，订购一份套餐 1

//原本数据库中套餐的余量

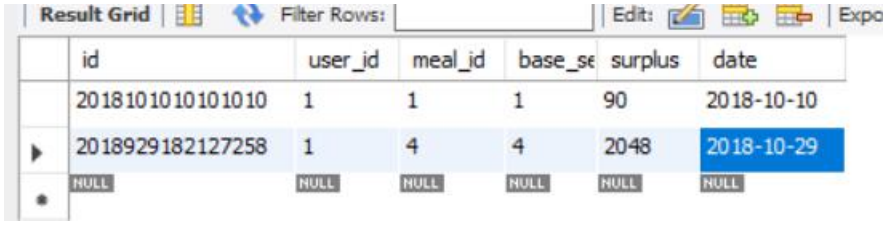


id	user_id	meal_id	base_se	surplus	date
20181010101010101010	1	1	1	90	2018-10-10
20189291821272258	1	4	4	2048	2018-10-29
20189291830509866	1	1	1	100	2018-10-29
NULL	NULL	NULL	NULL	NULL	NULL

//退订套餐 1

```
//退订---立即生效；
unsubscribeMealsFunctionImmediately( userId: 1,  mealId: 1);
```

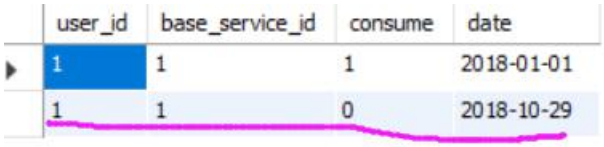
//退订了花费最少的那个（surplus---余量最多）



Result Grid | Filter Rows: | Edit: | Expo

id	user_id	meal_id	base_se	surplus	date
20181010101010101010	1	1	1	90	2018-10-10
20189291821272258	1	4	4	2048	2018-10-29
NULL	NULL	NULL	NULL	NULL	NULL

//在 table “user_meal_outside_charge”中存储退订的套餐花费了的内容，由于这里没有花费，于是存储的是基础服务 id=1 的数量 0



user_id	base_service_id	consume	date
1	1	1	2018-01-01
1	1	0	2018-10-29

//再次退订套餐 1

```
//退订---立即生效；
unsubscribeMealsFunctionImmediately( userId: 1,  mealId: 1);
```

//表格的内容

id	user_id	meal_id	base_se	surplus	date
20189291821272258	1	4	4	2048	2018-10-29
NULL	NULL	NULL	NULL	NULL	NULL

//在 table “user_meal_outside_charge”中存储退订的套餐花费了的内容，由于这里原本套餐中服务的数量为 100，余量为 90，于是存储的是基础服务 id=1 的数量 10

user_id	base_service_id	consume	date
1	1	1	2018-01-01
1	1	0	2018-10-29
1	1	10	2018-10-29

//用户对套餐的操作的记录---立即退订的 operate_id 为 2

user_id	operate_id	meal_id	operate_time
1	1	1	2018-01-01 10:10:10
1	1	4	2018-10-29 18:21:27
1	1	1	2018-10-29 18:30:51
1	2	1	2018-10-29 18:32:48
1	2	1	2018-10-29 18:36:15

；

对应操作所用的时间：

```
C:\Program Files\Java\jdk-9.0.4\bin\java.exe   -javaagent:D:
用户1对套餐1进行的立即退订的开始的时间：  2018-10-29 18:32:48
用户1对套餐1进行的立即退订的结束的时间：  2018-10-29 18:32:48
```

；

除代码之外其他需要说明的东西：

1：如果订购相同套餐，想退订一份，立即退订会帮助挑选套餐使用最小的那一份退订；

2：一份套餐中如果有多样基准服务，那么立即退订的时候每样基准服务的花费都会退订到位；

3：套餐中的余量不在本月也可以使用；

4：Main.java 代码的第 348 以及 368 行设置的内容，要求：一个套餐的基准服务内容不超过 20 项，否则就会出错---原本的基准服务类型只有 4 个，所以其实 int[5]就足够了，但是设置多一些，以防万一；

操作 4：

设计：

//退订---次月生效；

unsubscribeMealsFunctionNextMonth(int userId, int mealId);

仅向 table “user_meal_operator”中存入对套餐的操作，没有任何对套餐余量的影响等等；

运行截图：

//退订前---用户 1 下月退订套餐 3

id	user_id	meal_id	base_se	surplus	date
20189291847119 17	1	4	4	2048	2018-10-29
NULL	NULL	NULL	NULL	NULL	NULL

//退订

```
//退订---次月生效；
unsubscribeMealsFunctionNextMonth( userId: 1,  mealId: 3);
```

//控制台输出

```
用户1对套餐3进行的次月生效退订的开始的时间：  2018-10-29 18:49:30
用户1对套餐3进行的次月生效退订的结束的时间：  2018-10-29 18:49:30
```

//对套餐余量无影响

	id	user_id	meal_id	base_se	surplus	date
▶	2018929184711917	1	4	4	2048	2018-10-29
*	NULL	NULL	NULL	NULL	NULL	NULL

//记录最套餐的操作---本月退订的 operate_id 为 3

user_id	operate_id	meal_id	operate_time
1	1	1	2018-01-01 10:10:10
1	1	4	2018-10-29 18:21:27
1	1	1	2018-10-29 18:30:51
1	2	1	2018-10-29 18:32:48
1	2	1	2018-10-29 18:36:15
1	2	4	2018-10-29 18:46:04
1	1	4	2018-10-29 18:47:11
1	3	3	2018-10-29 18:49:30

对应操作所用的时间：

```
用户1对套餐3进行的次月生效退订的开始的时间：  2018-10-29 18:49:30
用户1对套餐3进行的次月生效退订的结束的时间：  2018-10-29 18:49:30
```

；

除代码之外其他需要说明的东西：

无；

操作 5：

设计：

call(int userId, String startTime, String endTime, String date);

先考虑套餐中是否有通话的余量：

如果有就抵扣（如果剩余多个通话套餐的余量，可以同时抵扣完一个再抵扣下一个）；

如果有余量，但是余量的总时间不够，那么抵扣完所有的套餐余量，然后再加上无法抵消的那部分；

如果没有余量，就使用基本资费；

运行截图：

//先订购 3 个套餐 1

```
//订购套餐；
//id为1的用户订购套餐4
orderMeals( userId: 1,  mealId: 1)
```

//剩余 300 分钟的通话余量

//跨套餐抵扣通话时间

//通话

```
//用户在通话情况下的资费生成；
call( userId: 1,  startTime: "01:01:01",  endTime: "05:01:01",  date: "2018-01-01");
```

//套餐余量被抵扣

201892919121438	1	1	1	0	2018-10-29
201892919123941	1	1	1	0	2018-10-29
201892919126334	1	1	1	60	2018-10-29

//套餐无法抵扣完通话时间

//通话

```
//用戶在通话情况下的资费生成;
call( userId: 1,  startTime: "01:01:01",  endTime: "05:01:01",  date: "2018-01-01");
```

//抵扣完套餐

2018929184711917	1	4	4	2048	2018-10-29
201892919121438	1	1	1	0	2018-10-29
201892919123941	1	1	1	0	2018-10-29
201892919126334	1	1	1	0	2018-10-29

//没抵扣完的通话时间，使用基准资费

user_id	base_service_id	consume	date
1	1	1	2018-01-01
1	1	0	2018-10-29
1	1	10	2018-10-29
1	4	0	2018-10-29
1	1	240	2018-10-29
1	1	180	2018-10-29

//原本无套餐的余量

id	user_id	meal_id	base_se	surplus	date
2018929184711917	1	4	4	2048	2018-10-29
NULL	NULL	NULL	NULL	NULL	NULL

//通话

```
//用戶在通话情况下的资费生成;
call( userId: 1,  startTime: "01:01:01",  endTime: "05:01:01",  date: "2018-01-01");
```

//使用基本资费

user_id	base_service_id	consume	date
1	1	1	2018-01-01
1	1	0	2018-10-29
1	1	10	2018-10-29
1	4	0	2018-10-29
1	1	240	2018-10-29

;

对应操作所用的时间：

```
用户1对通话操作记录开始的时间：  2018-10-29 19:04:38
用户1对通话操作记录结束的时间：  2018-10-29 19:04:38
```
；

除代码之外其他需要说明的东西：

通话情况下资费不足一分钟的按照一分钟计算；

操作 6：

设计：

sendMessage(int userId, String date);

如果有短信的套餐，就抵扣掉一条，如果没有，就增加基准资费；

运行截图：

//没有短信套餐/短信套餐无余量

//发短信

```
//用户在短信情况下的资费生成
//短信的日期：yyyy-mm-dd
sendMessage( userId: 1,  date: "2018-01-01");
```

//套餐外基准资费表增加一条发短信的记录

user_id	base_service_id	consume	date
1	1	1	2018-01-01
1	1	0	2018-10-29
1	1	10	2018-10-29
1	4	0	2018-10-29
1	1	240	2018-10-29
1	1	180	2018-10-29
1	2	1	2018-10-29

//订购一条短信的套餐

```
//订购套餐；
//id为1的用户订购套餐4
orderMeals( userId: 1,  mealId: 2);
```

//短信套餐有余量

//发短信

```
//用户在短信情况下的资费生成
//短信的日期：yyyy-mm-dd
sendMessage( userId: 1,  date: "2018-01-01");
```

//短信套餐中的数量从 200 减到 199

id	user_id	meal_id	base_se	surplus	date
2018929184711917	1	4	4	2048	2018-10-29
201892919111428	1	2	2	199	2018-10-29

；

对应操作所用的时间：

```
C:\Program Files\Java\jdk-9.0.4\bin\java.exe "ja
用户1对短信操作记录开始的时间：  2018-10-29 19:11:28
用户1对短信操作记录结束的时间：  2018-10-29 19:11:29
```
；

除代码之外其他需要说明的东西：

无；

操作 7：

设计：

useFlow(int userId, String date, float num, int isLocal);

isLocal 如果为 1，就是在本地使用流量，为 0 则是在全国使用流量；

如果是在本地使用流量，而本地流量套餐余量不够，就继续使用全国流量的套餐余量；

运行截图：

//没有套餐

//在全国使用 2050M 的流量

```
//用户在使用流量下的资费生成
//local为1代表用户是在本地使用的流量，否则，为非本地使用的流量
useFlow( userId: 1,  date: "2018-01-01",  num: 2050,  isLocal: 1);
```

user_id	base_service_id	consume	date
1	4	2050	2018-10-29
1	3	2050	2018-10-29

//在本地使用 2050M 的流量

```
//用户在使用流量下的资费生成
//local为1代表用户是在本地使用的流量，否则，为非本地使用的流量
useFlow( userId: 1,  date: "2018-01-01",  num: 2050,  isLocal: 0);
```

user_id	base_service_id	consume	date
1	4	2050	2018-10-29

//有套餐---先订购一份套餐 6（1G 本地流量，2G 全国流量）

```
//订购套餐；
//id为1的用户订购套餐4
orderMeals( userId: 1,  mealId: 6);
```

//在本地使用流量

//本地流量够使用---使用本地流量 100M

```
//用户在使用流量下的资费生成
//local为1代表用户是在本地使用的流量，否则，为非本地使用的流量
useFlow( userId: 1,  date: "2018-01-01",  num: 100,  isLocal: 1);
```

套餐原本 1024M 的本地流量减去 100M，变成 924M

2018929192452486	1	6	3	924	2018-10-29
2018929192452529	1	6	4	2048	2018-10-29

//本地流量不够使用，但是有全国流量够使用---使用 1024M 本地流量

```
//用户在使用流量下的资费生成
//local为1代表用户是在本地使用的流量，否则，为非本地使用的流量
useFlow( userId: 1,  date: "2018-01-01",  num: 1024,  isLocal: 1);
```

全国流量由原来的 2048M 减到 1948M（由于本地流量还剩余 924M，抵扣了这么多）

2018929192452486	1	6	3	0	2018-10-29
2018929192452529	1	6	4	1948	2018-10-29

//本地+全国的流量都不够使用，就将不够使用的部分增加到基准资费里面（如果没有剩余流量套餐就也是这个）---使用 1949M 本地流量

```
//用户在使用流量下的资费生成
//local为1代表用户是在本地使用的流量，否则，为非本地使用的流量
useFlow( userId: 1,  date: "2018-01-01",  num: 1949,  isLocal: 1);
```

本地，全国流量都抵扣完了，于是增加到额外的基准资费里面---base_service_id 为 3 代表使用的为本地流量，consume 为本地流量使用的数量 1（原本有 1948M 的全国流量，抵扣了这么多）

user_id	base_service_id	consume	date
1	1	10	2018-10-29
1	4	0	2018-10-29
1	1	240	2018-10-29
1	1	180	2018-10-29
1	2	1	2018-10-29
1	4	2	2018-10-29
1	4	2050	2018-10-29
1	3	2050	2018-10-29
1	3	1	2018-10-29

//在全国使用流量

先订购一份套餐 6（1G 本地流量，2G 全国流量）

```
//订购套餐；
//id为1的用户订购套餐4
orderMeals( userId: 1,  mealId: 6);
```

//注意下面对全国流量的使用，使用的时候，本地流量不变

//全国套餐余量足够使用

使用 1000M 全国流量，注意这里的 isLocal 的数值为 0（即：非本地流量）

```
//用户在使用流量下的资费生成
//local为1代表用户是在本地使用的流量，否则，为非本地使用的流量
useFlow( userId: 1,  date: "2018-01-01",  num: 1000,  isLocal: 0);
```

//之前数据库中可能有一条全国流量的记录仍旧有剩余，刚刚测试的时候没有注意到，麻烦把那条记录删掉，仅留下全国流量剩余 2048M 的那条记录就好了
//这是如果数据库中全国流量的套餐余量有 2048M，然后使用了 1000M 全国流量之后的结果（全国流量从 2048M 变成 1048M）

2018929193316611	1	6	3	1024	2018-10-29
2018929193316669	1	6	4	1048	2018-10-29
NULL	NULL	NULL	NULL	NULL	NULL

；

//全国套餐余量不够使用---增加基准资费

使用 1049M 全国流量，注意这里的 isLocal 的数值为 0（即：非本地流量）

```
//用户在使用流量下的资费生成
//local为1代表用户是在本地使用的流量，否则，为非本地使用的流量
useFlow( userId: 1,  date: "2018-01-01",  num: 1049,  isLocal: 0);
```

套餐中全国流量的余量变为 0

2018929193316611	1	6	3	1024	2018-10-29
2018929193316669	1	6	4	0	2018-10-29

增加没有抵扣掉的全国流量 1M（1049M 中抵扣掉了 1048M）

1	4	1	2018-10-29

；

对应操作所用的时间：

```
C:\Program Files\Java\jdk-9.0.4\bin\java.exe  -javaagen
用户1对使用流量操作记录开始的时间：  2018-10-29 19:25:35
用户1对使用流量操作记录结束的时间：  2018-10-29 19:25:36
```

；

除代码之外其他需要说明的东西：

无；

操作 8：

设计：

generateBillByMonth(int userId, String month);

从表格中对应获取数值，分类打印；

内容包括如下：

* （0）：打印：用户 userId 在 month 月的月账单；
* （1）：用户在套餐外的基准资费的使用情况以及费用（各项费用+总费）；
* （2）：用户订购套餐费用（各项费用+总费）；
* （3）：用户立即退订套餐费用（各项费用+总费）；
* （4）：用户下月退订套餐费用（各项费用+总费）；

* （5）：用户本月使用总费用；

运行截图：

//输入想要获取的月账单的用户的 id 和月份

```
//某个用户月账单的生成
//month的参数需要是"yyyy-mm"的形式
//在控制台输出
generateBillByMonth( userId: 1,  month: "2018-10");
```

//打印获取的月账单的结果

```
对1进行进行2018-10月账单的生成开始的时间：  2018-10-29 19:45:02
用户 1 在 2018-10 月的月账单
（1）：基准服务：
2018-10-29 使用 通话 基准服务数量 0.0，花费 0.0 元；
2018-10-29 使用 通话 基准服务数量 10.0，花费 5.0 元；
2018-10-29 使用 国内流量（4G） 基准服务数量 0.0，花费 0.0 元；
2018-10-29 使用 通话 基准服务数量 240.0，花费 120.0 元；
2018-10-29 使用 通话 基准服务数量 180.0，花费 90.0 元；
2018-10-29 使用 短信 基准服务数量 1.0，花费 0.1 元；
2018-10-29 使用 国内流量（4G） 基准服务数量 2.0，花费 10.0 元；
2018-10-29 使用 国内流量（4G） 基准服务数量 2050.0，花费 10250.0 元；
2018-10-29 使用 本地流量（4G） 基准服务数量 2050.0，花费 4100.0 元；
2018-10-29 使用 本地流量（4G） 基准服务数量 1.0，花费 2.0 元；
2018-10-29 使用 国内流量（4G） 基准服务数量 1.0，花费 5.0 元；
（2）：订购套餐：
2018-10-29 18:21:27.0 订购套餐 国内流量套餐 1份，花费 30.0 元；
2018-10-29 18:30:51.0 订购套餐 话费套餐 1份，花费 20.0 元；
2018-10-29 18:47:11.0 订购套餐 国内流量套餐 1份，花费 30.0 元；
2018-10-29 19:01:21.0 订购套餐 话费套餐 1份，花费 20.0 元；
2018-10-29 19:01:24.0 订购套餐 话费套餐 1份，花费 20.0 元；
2018-10-29 19:01:26.0 订购套餐 话费套餐 1份，花费 20.0 元；
2018-10-29 19:11:14.0 订购套餐 短信套餐 1份，花费 10.0 元；
2018-10-29 19:24:52.0 订购套餐 本地国内流量套餐 1份，花费 35.0 元；
2018-10-29 19:33:16.0 订购套餐 本地国内流量套餐 1份，花费 35.0 元；
（3）：立即退订套餐：
2018-10-29 18:32:48.0 立即退订套餐 话费套餐 1份，返回 20.0 元；
2018-10-29 18:36:15.0 立即退订套餐 话费套餐 1份，返回 20.0 元；
2018-10-29 18:46:04.0 立即退订套餐 国内流量套餐 1份，返回 30.0 元；
（4）：下月退订套餐：
2018-10-29 18:49:30.0 下月退订套餐 本地流量套餐 1份，返回 0 元；
（5）：月总花费：
2018-10 月总花费： 14732.1 元；
对1进行进行2018-10月账单的生成结束的时间：  2018-10-29 19:45:03
```

;

对应操作所用的时间：

//开始时间

```
对1进行进行2018-10月账单的生成开始的时间：  2018-10-29 19:45:02
```

//结束时间

对1进行进行2018-10月账单的生成结束的时间： 2018-10-29 19:45:03

；

除代码之外其他需要说明的东西：

无；

（3）：可能的优化方案：

1：在写代码的时候，明显感觉到有很多是重用的代码，之前没有考虑到将关于数据库的代码写成增，删，改，查，这样四个对表格的操作，很是可惜，害的自己之后写代码麻烦而且容易出错（没有写成函数可以调用）；

另外，没有使用 hibernate，hibernate 应该会更加好用；

2：没有将连接数据库和数据库关闭的代码写在多有方法的最外面，当时没想起来，还得自己不知道链接了多少回的数据库；

3：多了 user_operate_content 的表格（记录对套餐的操作类型），因为在对套餐操作的时候，只有 3 个值：“订购”，“立即退订”，“下月退订”，专门使用一个表格来记录很繁琐，应该将 user_meal_operator 表的 operate_id 直接换成 operate 值，然后改变值的数据类型为 enum；

4：在建立数据表的时候，最好添加一个 primary key，不然之后修改的时候，不好修改；因为或许两条记录的相似度很高，那就不知道应该要修改哪条记录了，后来我特地修改了 user_meal_surplus 表，为它添加了 primary key，但是我的数据库中的表还是有几张是没有 primary key，下次可以为表格添加一个自增的 primary key（如果不需要考虑插入指定 id 的记录）；

5：代码的函数的参数：通话的开始和结束时间默认是正常的，即：开始时间<结束时间，且不跨天（跨天打电话时，只输入小时，没有输入天数，就会出错）；

6：我所考虑的套餐没有包括不退订那么下个月再自动买一次，我所考虑的仅仅是以前买，没用完就可以继续使用的套餐，于是下月退订仅仅是在退订操作里面留下记录，于是后续系统如果更新，增加一种套餐仅在当月起效的时候，那么现在的通话，短信，流量等等套餐就无法延续到下个月，那么代码要改，数据库的 meal_service 表的套餐的属性设定中也需要增加一下 can_only_use_this_month（int），即：是否只可以在本月使用；

（4）：其余说明（对本代码）：

1：在使用套餐的时候，按照余量最少的那个开始使用，若最少的余量有多个一样多的就随机使用余量；