集成测试报告

1 引言

1.1 目的

本文档对“随手记”记账软件的集成测试结果进行总结，对bug 进行汇总和分析，为后续系统测试和回归提供参考。本文档的主要读者是测试经理、测试人员和开发人员。

1.2 测试策略

集成测试重点测试两方面：1）模块间接口的正确性；2）集成后的子系统的功能的正确性。决定集成测试次序的依据是《软件概要设计说明》中的软件层次架构图，采用自底向上的集成策略，从程序模块结构的最底层模块开始组装和测试，逐层向上，直到整个系统集成完毕。集成测试阶段采用黑盒测试的测试方法对子系统功能进行验证，采用白盒测试方法对模块间接口进行验证。

（1） 首先对中层模块内进行集成。

（2） 在完成中层模块集成测试后，对中层模块和界面模块进行集成测试。输入域和输出域覆盖上进行测试用例的设计，同时考虑接口功能的完整性。虽然仅仅是两个到三个模块的集成，但实际采用的是自顶向下集成的方法，并用selenium进行脚本记录。

（3） 对于界面接口，利用场景法设计测试用例，根据测试用例手动验证其功能，详见系统测试，同时关注其是否调用正确的接口函数，并且是否能够正确的传入参数，正确的获取返回值和输出参数。

（4） 参考等价类划分方法。

（5） 参考边界值划分方法。

（6） 参考使用错误猜测方法。

（7） 覆盖分析。

2 集成测试接口分析

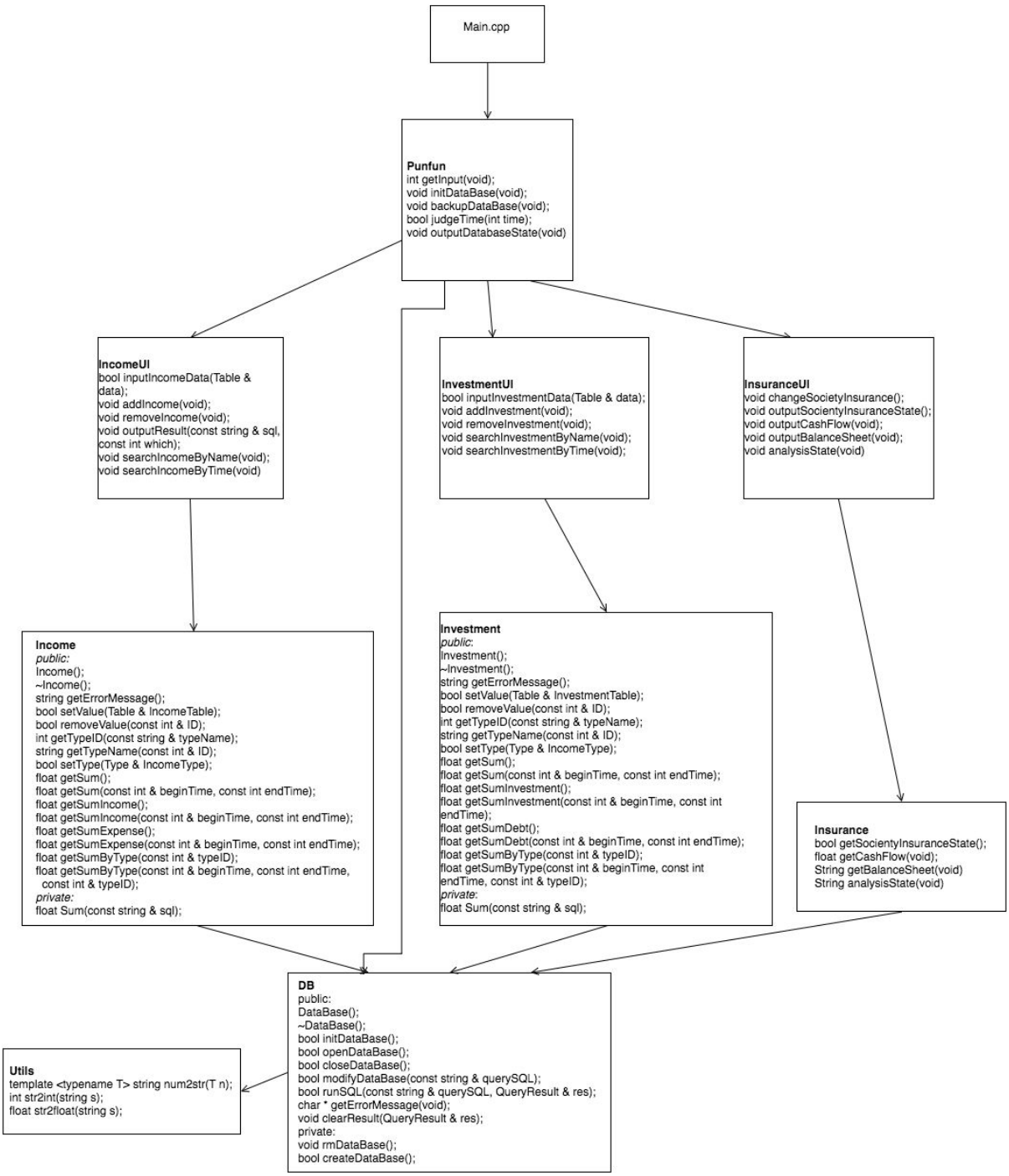
Main.cpp
Punfun
int getInput(void);
void initDataBase(void);
void backupDataBase(void);
bool judgeTime(int time);
void outputDatabaseState(void)
IncomeUI
bool inputIncomeData(Table & data);
void addIncome(void);
void removeIncome(void);
void outputResult(const string & sql, const int which);
void searchIncomeByName(void);
void searchIncomeByTime(void)
InvestmentUI
bool inputInvestmentData(Table & data);
void addInvestment(void);
void removeInvestment(void);
void searchInvestmentByName(void);
void searchInvestmentByTime(void);
InsuranceUI
void changeSocietyInsurance();
void outputSocientyInsuranceState();
void outputCashFlow(void);
void outputBalanceSheet(void);
void analysisState(void)
Income
public:
Income();
~Income();
string getErrorMessage();
bool setValue(Table & IncomeTable);
bool removeValue(const int & ID);
int getTypeID(const string & typeName);
string getTypeName(const int & ID);
bool setType(Type & IncomeType);
float getSum();
float getSum(const int & beginTime, const int endTime);
float getSumIncome();
float getSumIncome(const int & beginTime, const int endTime);
float getSumExpense();
float getSumExpense(const int & beginTime, const int endTime);
float getSumByType(const int & typeID);
float getSumByType(const int & beginTime, const int endTime, const int & typeID);
private:
float Sum(const string & sql);
Investment
public:
Investment();
~Investment();
string getErrorMessage();
bool setValue(Table & InvestmentTable);
bool removeValue(const int & ID);
int getTypeID(const string & typeName);
string getTypeName(const int & ID);
bool setType(Type & IncomeType);
float getSum();
float getSum(const int & beginTime, const int endTime);
float getSumInvestment();
float getSumInvestment(const int & beginTime, const int endTime);
float getSumDebt();
float getSumDebt(const int & beginTime, const int endTime);
float getSumByType(const int & typeID);
float getSumByType(const int & beginTime, const int endTime, const int & typeID);
private:
float Sum(const string & sql);
Insurance
bool getSocientyInsuranceState();
float getCashFlow(void);
String getBalanceSheet(void)
String analysisState(void)
DB
public:
DataBase();
~DataBase();
bool initDataBase();
bool openDataBase();
bool closeDataBase();
bool modifyDataBase(const string & querySQL);
bool runSQL(const string & querySQL, QueryResult & res);
char * getErrorMessage(void);
void clearResult(QueryResult & res);
private:
void rmDataBase();
bool createDataBase();
Utils
template <typename T> string num2str(T n);
int str2int(string s);
float str2float(string s);

图1 模块调用关系图

2.1 数据库模块接口分析

标识符	名称	调用层次	调用函数次数
HLD_001_INT_001	DataBase()	0	0
HLD_001_INT_002	~DataBase()	0	0
HLD_001_INT_003	openDataBase(void)	0	0
HLD_001_INT_004	closeDataBase(void)	0	0
HLD_001_INT_005	createDataBase(void)	1	2
HLD_001_INT_006	rmDataBase(void)	0	0
HLD_001_INT_007	modifyDataBase(const string & querySQL)	1	2
HLD_001_INT_008	runSQL(const string & querySQL,QueryResult & res)	1	2
HLD_001_INT_009	clearResult(QueryResult & res)	0	0

数据库模块是“随手记”系统最底层的被调用模块，通过分析，其对外接口均只有一层的调用函数，或者不调用，因此这里不对其进行测试。

2.2 收入支出模块接口分析

标识符	名称	调用层次	调用函数次数
HLD_002_INT_001	Income()	1	1
HLD_002_INT_002	~Income()	1	1
HLD_002_INT_003	string getErrorMessage()	0	0
HLD_002_INT_004	float getSum()	2	3
HLD_002_INT_005	bool setValue(Table & IncomeTable);	1	2
HLD_002_INT_006	bool removeValue(const int & ID);	1	2

HLD_002_INT_007	int getTypeID(const string & typeName);	1	2
HLD_002_INT_008	string getTypeName(const int & ID);	1	2
HLD_002_INT_009	bool setType(Type & IncomeType);	1	2
HLD_002_INT_011	float getSum(const int & beginTime, const int endTime);	2	3
HLD_002_INT_012	float getSumIncome();	2	3
HLD_002_INT_013	float getSumIncome(const int & beginTime, const int endTime);	2	3
HLD_002_INT_014	float getSumExpense();	2	3
HLD_002_INT_015	float getSumExpense(const int & beginTime, const int endTime);	2	3
HLD_002_INT_016	float getSumByType(const int & typeID);	2	3
HLD_002_INT_017	float getSumByType(const int & beginTime, const int endTime, const int & typeID);	2	3
HLD_002_INT_018	float Sum(const string & sql);	1	2

通过分析收入支出模块的每个对外的接口函数，其中部分函数只有一层函数调用，甚至不调用，集成测试阶段只对调用层次为2层及以上的接口进行测试。因此只对getSum, getSumIncome, getSumExpense及getSumByType进行测试。

2.3 资产负债模块接口分析

标识符	名称	调用层次	调用函数次数
HLD_003_INT_001	Investment();	1	1
HLD_003_INT_002	~Investment();	1	1
HLD_003_INT_003	string getErrorMessage();	0	0
HLD_003_INT_004	bool setValue(Table & InvestmentTable);	1	2
HLD_003_INT_005	bool removeValue(const int & ID);	1	2

HLD_003_INT_006	int getTypeID(const string & typeName);	1	2
HLD_003_INT_007	string getTypeName(const int & ID);	1	2
HLD_003_INT_008	bool setType(Type & IncomeType);	1	2
HLD_003_INT_009	float getSum();	2	3
HLD_003_INT_010	float getSum(const int & beginTime, const int endTime);	2	3
HLD_003_INT_011	float getSumInvestment();	2	3
HLD_003_INT_012	float getSumInvestment(const int & beginTime, const int endTime);	2	3
HLD_003_INT_013	float getSumDebt();	2	3
HLD_003_INT_014	float getSumDebt(const int & beginTime, const int endTime);	2	3
HLD_003_INT_015	float getSumByType(const int & typeID);	2	3
HLD_003_INT_016	float getSumByType(const int & beginTime, const int endTime, const int & typeID);	2	3
HLD_003_INT_017	float Sum(const string & sql);	1	2

通过分析资产负债模块的每个对外的接口函数，其中部分函数只有一层函数调用，甚至不调用，集成测试阶段只对调用层次为2层及以上的接口进行测试。类似于收入支出模块，只对getSum, getSumIncome, getSumExpense及getSumByType进行测试。

2.4 程序处理模块接口分析

标识符	名称	调用层次	调用函数次数
HLD_004_INT_001	void initDataBase(void)	1	1
HLD_004_INT_002	void backupDataBase(void)	0	0
HLD_004_INT_003	bool judgeTime(int time)	0	0
HLD_004_INT_004	bool inputIncomeData(Table & data)	1	5
HLD_004_INT_005	void addIncome(void)	2	7

HLD_004_INT_006	void removeIncome(void)	1	1
HLD_004_INT_007	void outputResult(const string & sql, const int which)	1	1
HLD_004_INT_008	void searchIncomeByName(void)	2	2
HLD_004_INT_009	void searchIncomeByTime(void)	2	2
HLD_004_INT_010	bool inputInvestmentData(Table & data)	1	5
HLD_004_INT_011	void addInvestment(void)	2	7
HLD_004_INT_012	void removeInvestment(void)	1	1
HLD_004_INT_013	void searchInvestmentByName(void)	2	2
HLD_004_INT_014	void searchInvestmentByTime(void)	2	2
HLD_004_INT_015	void changeSocietyInsurance()	1	5
HLD_004_INT_016	void outputSocientyInsuranceState()	1	3
HLD_004_INT_017	void outputCashFlow(void)	1	7
HLD_004_INT_018	void outputBalanceSheet(void)	1	7
HLD_004_INT_019	void analysisState(void)	1	8
HLD_004_INT_020	void outputDatabaseState(void)	1	1

通过分析程序处理模块的每个对外接口函数，其中initDatabase, backupDatabase, judgeTime 及 outputDatabaseState调用函数层次为一或没有调用其他函数，这里不对其进行测试，其他接口均进行集成测试。

2.5 界面模块（主函数）接口分析

接口序号	接口描述
GUN_INT_001	初始化数据库
GUN_INT_002	备份/恢复数据库
GUN_INT_003	增加一项收入支出
GUN_INT_004	删除一项收入支出
GUN_INT_005	用项目名称检索收入支出记录
GUN_INT_006	用项目日期检索收入支出记录

GUN_INT_007	增加一项资产负债
GUN_INT_008	删除一项资产负债
GUN_INT_009	用项目名称检索资产负债记录
GUN_INT_010	用项目日期检索资产负债记录
GUN_INT_011	增加社保记录
GUN_INT_012	输出社保状态
GUN_INT_013	输出指定日期范围的现金流量表
GUN_INT_014	输出指定日期范围的资产负债表
GUN_INT_015	分析财务状况
GUN_INT_016	输出当前数据库存储的记录总数

界面模块直接调用程序处理模块的接口，不包含逻辑处理，故将界面模块与程序处理模块视作一个整体，在集成测试阶段对表2-5中所列接口进行基本功能验证。

综上，集成测试阶段的被测接口如下：

标识符	名称
HLD_002_INT_011	float getSum(const int & beginTime, const int endTime);
HLD_002_INT_013	float getSumIncome(const int & beginTime, const int endTime);
HLD_002_INT_015	float getSumExpense(const int & beginTime, const int endTime);
HLD_002_INT_017	float getSumByType(const int & beginTime, const int endTime, const int & typeID);
HLD_003_INT_011	float getSum(const int & beginTime, const int endTime);
HLD_003_INT_013	float getSumIncome(const int & beginTime, const int endTime);
HLD_003_INT_015	float getSumExpense(const int & beginTime, const int endTime);
HLD_003_INT_017	float getSumByType(const int & beginTime, const int endTime, const int & typeID);
GUN_INT_001	初始化数据库

GUN_INT_002	备份/恢复数据库
GUN_INT_003	增加一项收入支出
GUN_INT_004	删除一项收入支出
GUN_INT_005	用项目名称检索收入支出记录
GUN_INT_006	用项目日期检索收入支出记录
GUN_INT_007	增加一项资产负债
GUN_INT_008	删除一项资产负债
GUN_INT_009	用项目名称检索资产负债记录
GUN_INT_010	用项目日期检索资产负债记录
GUN_INT_011	增加社保记录
GUN_INT_012	输出社保状态
GUN_INT_013	输出指定日期范围的现金流量表
GUN_INT_014	输出指定日期范围的资产负债表
GUN_INT_015	分析财务状况
GUN_INT_016	输出当前数据库存储的记录总数

3 采用的测试方法和技术

（1）集成测试阶段，采用自底向上的测试策略。优于本系统的底层模块为数据库操作模块，模块接口和行为相对稳定，且尽早对底层模块行为进行验证对于上层模块的开发有着至关重要的作用，因此采用自底向上的集成测试策略。
（2）参考等价类测试方法。
（3）参考边界值测试方法。
（4）参考场景法测试方法。
（5）覆盖分析。

4 测试环境

（1）硬件：HP Xeon520
（2）软件：

- sqlite 3
- 操作系统：Windows7及以上

5 测试通过／失败标准

测试通过的标准如下：

- 所有的接口用例都被执行并通过；
- 所有发现的缺陷都被修正并回归测试；
- 被测接口的输入域被100%覆盖；
- 被测接口的函数调用路径被100%覆盖。

6 用例分析与设计

6.1 收入支出模块的集成测试

1）HLD_002_INT_011测试用例设计与分析

（1）设计标识符：IT_TD_001

（2）被测特性：

- 开始时间不合法，查询失败；
- 开始时间不在1980/01/01～2099/12/31范围内，查询失败；
- 结束时间不合法，查询失败；
- 结束时间不在1980/01/01～2099/12/31范围内，查询失败；
- 开始时间晚于结束时间，查询失败；
- 输入参数合法，查询成功。

（3）测试方法：

分析 getSum 的函数调用关系图，从 getSum 的单元测试可知，用桩对 Sum 函数及 Sum 中调用的 runSQL，getErrorMessage 进行了替代。对于 getSum 的集成来说，关键是验证 getSum 与 Sum 的接口是否与预计的效果相同。采用决策表分析的方法设计用例。

（4）测试用例设计：

条件桩：

- c1: 开始时间 < 结束时间？
- c2: 开始时间合法？
- c3: 1980/01/01 <= 开始时间 <= 2099/12/31？
- c4: 结束时间合法？
- c5: 1980/01/01 <= 结束时间 <= 2099/12/31？

行动桩

- a1: 查询成功
- a1: 查询失败

由此，得到决策表如下：

c1:开始时间<结束时间？	Y	Y	Y	Y	Y	N
c2:开始时间合法？	Y	Y	Y	Y	N	-
c3:1980/01/01 <= 开始时间 <= 2099/12/31？	Y	Y	Y	N	-	-
c4:结束时间合法？	Y	Y	N	-	-	-
c5:1980/01/01 <= 结束时间 <= 2099/12/31？	Y	N	-	-	-	-
a1:查询成功	X					
a2:查询失败		X	X	X	X	X

（5）测试用例：参见IT_TD_001.xls
（6）测试通过／失败的标准：所有用例都必须被执行，且没有发现缺陷。

2）HLD_002_INT_013测试用例设计与分析

（1）设计标识符：IT_TD_002
（2）getSumIncome 的输入变量为查询的开始时间、结束时间，与 getSum 相同；程序的处理逻辑亦与 getSum 相同，唯一的不同在于进行SQL语句查询时，getSumIncome 限定金额为正数，但这并不影响测试用例的设计与分析。因此，getSumIncome 的测试用例设计思路同 getSum 所分析，得到的测试用例参加 IT_TD_002.xls。

3）HLD_002_INT_015测试用例设计与分析

（1）设计标识符：IT_TD_003
（2）getSumExpense 的输入变量为查询的开始时间、结束时间，与 getSum 相同；程序的处理逻辑亦与 getSum 相同，唯一的不同在于进行SQL语句查询时，getSumExpense 限定金额为负数，但这并不影响测试用例的设计与分析。因此，getSumExpense 的测试用例设计思路同getSum 所分析，得到的测试用例参加 IT_TD_003.xls。

4）HLD_002_INT_017测试用例设计与分析

（1）标识符定义：IT_TD_004
（2）被测特性：

- 开始时间不合法，查询失败；
- 开始时间不在1980/01/01～2099/12/31范围内，查询失败；
- 结束时间不合法，查询失败；

- 结束时间不在1980/01/01～2099/12/31范围内，查询失败；
- 开始时间晚于结束时间，查询失败；
- 输入项目类型代码为小数，查询失败；
- 输入项目类型代码为负整数，查询失败；
- 输入项目类型代码为正整数，且不在已有TypeID范围内，查询失败；
- 输入参数合法，查询成功。

（3）测试方法：

分析 getSumByType 的函数调用关系图，从 getSumByType 的单元测试可知，用桩对 Sum 函数及 Sum 中调用的 runSQL，getErrorMessage 进行了替代。对于 getSumByType 的集成来说，关键是验证 getSumByType 与 Sum 的接口是否与预计的效果相同。采用正交实验法设计用例。

（4）测试用例的设计

- 分析因素数：getSumByType 共有3个输入变量：查询开始时间、结束时间、项目类型代码，故因素数为3。
- 分析每个因素对应的水平数：

表1 因素状态（水平）

状态／因素	开始时间	结束时间	项目类型代码
0	不合法时间	不合法时间	小数
1	小于1980/01/01	小于1980/01/01	负整数
2	大于1980/01/01且小于结束时间	大于开始时间且小于2099/12/31	正整数且超出项目类型范围
3	大于2099/12/31	大于2099/12/31	正整数且在项目类型范围内

将中文转化成字母，得到新的因素状态如表2所示，

状态／因素	开始时间	结束时间	项目类型代码
0	A_1	B_1	C_1
1	A_2	B_2	C_2
2	A_3	B_3	C_3
3	A_4	B_4	C_4

- 从正交表中开始查找，选择正交表：
  1. 表中的因素数>=3；
  2. 表中至少有3个因素的水平数>=4；
  3. 行数取最少的一个，即满足（4^3）的最少行数 3 x （ 4-1）＋1＝10.

  最后选中正交表公式$L_{16}(4^5)$，对应的正交矩阵如表2所示，

	1	**2**	**3**	**4**	**5**
1	0	0	0	0	0
2	0	1	1	1	1
3	0	2	2	2	2
4	0	3	3	3	3
5	1	0	1	2	3
6	1	1	0	3	2
7	1	2	3	0	1
8	1	3	2	1	0
9	2	0	2	3	1
10	2	1	3	2	0
11	2	2	0	1	3
12	2	3	1	0	2
13	3	0	3	1	2
14	3	1	2	0	3
15	3	2	1	3	0
16	3	3	0	2	1

用字母代替正交表如表4,

	1	**2**	**3**	**4**	**5**
1	A_1	B_1	C_1	0	0
2	A_1	B_2	C_2	1	1
3	A_1	B_3	C_3	2	2
4	A_1	B_4	C_4	3	3
5	A_2	B_1	C_2	2	3
6	A_2	B_2	C_1	3	2
7	A_2	B_3	C_4	0	1
8	A_2	B_4	C_3	1	0

9	A_3	B_1	C_3	3	1
10	A_3	B_2	C_4	2	0
11	A_3	B_3	C_1	1	3
12	A_3	B_4	C_2	0	2
13	A_4	B_1	C_4	1	2
14	A_4	B_2	C_3	0	3
15	A_4	B_3	C_2	3	0
16	A_4	B_4	C_1	2	1

第四五列没有意义，可以去掉。由此得到16个测试用例。

（5）测试用例：参见IT_TD_005.xls

（6）测试通过标准：所有用例都被执行，且没有发现缺陷。

5）HLD_003_INT_011～017测试用例设计与分析

资产负债模块与收入支出模块输入变量、处理逻辑类似，其对外接口的测试用例的设计与分析参见 1）～4），这里不再详述。

6）GUN_INT_001测试用例设计与分析

（1）设计标识符：IT_TD_005

（2）被测特性：初始化数据库界面接口

（3）测试方法：界面接口的测试采用手动测试的方法。

（4）测试用例的设计与分析：

- 检测接口输入的正确性：
  输入的合法字符串能否正确传递给 initDatabase接口函数；
  输入不合法字符串时，能否提出“输入错误”信息。
- 检测接口输出的正确性：
  初始化数据库成功时，能否正确提示；
  输入参数不合法时，能否提示“输入错误”信息。

（5）测试用例如下：

测试项标识符	测试项描述	预期结果
IT_TD_005_001	选择“1”－输入“yes”	“数据库初始完毕！”
IT_TD_005_002	选择“1”－输入“no”	返回选择菜单

IT_TD_005_003	选择“1”－输入“y”	错误提示信息

7）GUN_INT_002测试用例设计与分析

（1）设计标识符：IT_TD_006
（2）被测特性：备份／恢复数据库
（3）测试方法：界面接口的测试采用手动测试的方法。
（4）测试用例的设计与分析：

- 检测接口输入的正确性：
  输入的合法字符串能否正确传递给 backupDatabase接口函数；
  输入文件名不存在时，能否提出“输入错误”信息。
- 检测接口输出的正确性：
  备份／恢复数据库成功时，能否正确提示；
  输入参数不合法时，能否提示“输入错误”信息。
- 能否在任意步骤退出，并回滚该步骤所有操作。

（5）测试用例如下：

测试项标识符	测试项	预期结果
IT_TD_006_001	选择“2”－选择“1”（备份数据库文件）－输入正确文件路径	“操作完成！”
IT_TD_006_002	选择“2”－选择“2”（恢复数据库文件）－输入正确文件路径	“操作完成！”
IT_TD_006_003	选择“2”－选择“1”（备份数据库文件）－输入错误文件路径	提示备份数据库失败
IT_TD_006_004	选择“2”－选择“2”（恢复数据库文件）－输入错误文件路径	提示恢复数据库失败
IT_TD_006_005	选择“2”－退出	返回选择菜单
IT_TD_006_006	选择“2”－选择“1”－退出	返回选择菜单

8）GUN_INT_003测试用例设计与分析

（1）设计标识符：IT_TD_007
（2）被测特性：增加收入－支出项目
（3）测试方法：界面接口的测试采用手动测试的方法。
（4）测试用例的设计与分析：

- 检测接口输入的正确性：
  输入的合法字符串能否正确传递给 inputIncomeData接口函数；
  输入字符串不合法时，能否提出“输入错误”信息。
- 检测接口输出的正确性：
  收入－支出项目添加成功时，能否正确提示；

输入参数不合法时，能否提示“输入错误”信息。

- 能否在任意步骤退出，并回滚该步骤所有操作。

（5）测试用例：

测试项标识符	测试项描述	预期结果
IT_TD_007_001	项目发生时间不合法	“输入时间错误”并要求重新输入时间
IT_TD_007_002	项目发生时间不在1980/01/01~2099/12/31之间	“输入时间错误！”并要求重新输入时间
IT_TD_007_003	输入项目名称为空	回车无法进入下一项输入，必须输入项目名称。
IT_TD_007_004	输入项目金额为字符	提示错误信息，并要求重新输入。
IT_TD_007_005	输入项目类型代码为小数	提示错误信息，并要求重新输入。
IT_TD_007_006	输入项目类型代码为负数	提示错误信息，并要求重新输入。
IT_TD_007_007	输入项目类型代码超出范围	提示错误信息，并要求重新输入。
IT_TD_007_008	输入项目备注为空	回车无法进入下一项的输入，必须输入项目备注。
IT_TD_007_009	在任意步骤退出	返回选择菜单

9）GUN_INT_004测试用例设计与分析

（1）设计标识符：IT_TD_008

（2）被测特性：删除收入－支出项目

（3）测试方法：界面接口的测试采用手动测试的方法。

（4）测试用例的设计与分析：

- 检测接口输入的正确性：
  输入的合法字符串能否正确传递给 removeIncome接口函数；
- 检测接口输出的正确性：
  收入－支出项目删除成功时，能否正确提示；
  输入ID不合法或不存在时，能否提示“输入错误”信息。
- 能否在任意步骤退出，并回滚该步骤所有操作。

（5）测试用例如下：

测试项标识符	测试项	预期结果

IT_TD_008_001	选择“4”－输入ID为1.1	“删除失败！”
IT_TD_008_002	选择“4”－输入ID为－1	“删除失败！”
IT_TD_008_003	选择“4”－输入ID为100	“删除失败！”
IT_TD_008_004	选择“4”－输入ID为20	“删除成功！”

10）GUN_INT_005测试用例设计与分析

（1）设计标识符：IT_TD_009
（2）被测特性：按项目名称查询收入－支出项目
（3）测试方法：界面接口的测试采用手动测试的方法。
（4）测试用例的设计与分析：

- 检测接口输入的正确性：
  输入的合法字符串能否正确传递给 searchIncomeByName 接口函数；
- 检测接口输出的正确性：
  收入－支出项目查询成功时，能否正确提示；
  收入－支出项目查询失败时，能否提示错误信息。
- 能否在任意步骤退出，并回滚该步骤所有操作。

（5）测试用例如下：

测试项标识符	测试项	预期结果
IT_TD_009_001	选择“5”－“1月工资”	显示收入信息
IT_TD_009_002	选择“5”－“1月分红”	“查询失败！”

11）GUN_INT_006测试用例设计与分析

（1）设计标识符：IT_TD_010
（2）被测特性：按项目时间查询收入－支出项目
（3）测试方法：界面接口的测试采用手动测试的方法。
（4）测试用例的设计与分析：

- 检测接口输入的正确性：
  输入的合法字符串能否正确传递给 searchIncomeByTime 接口函数；
- 检测接口输出的正确性：
  收入－支出项目查询成功时，能否正确提示；
  收入－支出项目查询失败时，能否提示错误信息。
- 能否在任意步骤退出，并回滚该步骤所有操作。

（5）测试用例如下：

测试项标识符	测试项	预期结果
IT_TD_010_001	选择“5”－开始时间不合法	提示错误信息

IT_TD_010_002	选择“5”－开始时间不在1980/01/01～2099/12/31范围内	提示错误信息
IT_TD_010_003	选择“5”－结束时间不合法	提示错误信息
IT_TD_010_004	选择“5”－结束时间不在1980/01/01～2099/12/31范围内	提示错误信息
IT_TD_010_005	选择“5”－开始时间晚于结束时间	提示错误信息
IT_TD_010_006	选择“5”－开始时间结束时间均合法，且开始时间早于结束时间	输出正确查询结果

7. 缺陷汇总

测试项编号	**IT_TD_005_003**			
优先级	中			
测试项描述	初始化数据库界面接口			
采用测试方法	手动测试			
用例序号	输入	期望结果	实际结果	判定
001	选择“1”-输入“y”	错误提示信息	未显示错误信息	true

测试项编号	**IT_TD_006_003**			
优先级	中			
测试项描述	备份/恢复数据库			
采用测试方法	手动测试			
用例序号	输入	期望结果	实际结果	判定
001	选择“2”-选择“1”(备份数据库文 件)一输入错误文件路径	提示备份数据库错误	无错误信息	true

测试项编号	**IT_TD_006_004**			
优先级	中			
测试项描述	备份/恢复数据库			
采用测试方法	手动测试			
用例序号	输入	期望结果	实际结果	判定
001	选择“2”-选择“2”(恢复数据库文 件)-输入错误文件路径	提示恢复数据库错误	无错误信息	true

测试项编号	**IT_TD_006_005**			
优先级	中			
测试项描述	备份/恢复数据库			
采用测试方法	手动测试			
用例序号	输入	期望结果	实际结果	判定
001	选择“2”-退出	返回选择	直接输出	true

		菜单	0	

测试项编号	**IT_TD_007_004**			
优先级	中			
测试项描述	增加收入-支出项目			
采用测试方法	手动测试			
用例序号	输入	期望结果	实际结果	判定
001	输入项目金额为字符	提示错误信息,并要求重新输 入。	无错误信息	true

测试项编号	**IT_TD_007_005**			
优先级	中			
测试项描述	增加收入-支出项目			
采用测试方法	手动测试			
用例序号	输入	期望结果	实际结果	判定
001	输入项目类型代码为小数	提示错误信息,并要求重新输 入。	无错误信息	true

测试项编号	**IT_TD_007_006**
优先级	中
测试项描述	增加收入-支出项目
采用测试方	手动测试

法				
用例序号	输入	期望结果	实际结果	判定
001	输入项目类型代码为负数	提示错误信息,并要求重新输 入。	无错误信息	true

测试项编号	**IT_TD_007_009**			
优先级	中			
测试项描述	增加收入-支出项目			
采用测试方法	手动测试			
用例序号	输入	期望结果	实际结果	判定
001	在任意步骤退出	返回选择菜单	仍停留在原界面	true