大規模ソフトウェア論　開発演習２　課題

「酒屋在庫管理システム」

（ソースコード・テストレポート）

作成者：チーム生ハム

192X221X 矢吹直也
190X214X 平山孝輔
192X218X 三浦稚咲

2019/08/02 Version 1.0

１．ソースコード

酒屋在庫管理システムは 7 つのクラスから構成される．クラスはそれぞれ，酒屋在庫管理システムを実行するクラス **LiquorSystem**，キーボード入力のためのクラス **KeyBord**，酒屋の操作を管理するクラス **LiquorShop**，出荷伝票のクラス **ShippedInformation**，入荷依頼のクラス **ArrivalRequest**，銘柄と本数による酒のクラス **Liquor**，および注文情報のクラス **OrderInformation** である．

なお，これらのソースコードは，本レポートの末尾にまとめて掲載している．

２．テストレポート

本レポートにおいては，内部設計に基づいた**単体テスト**，外部設計に基づいた**結合テスト**および要求定義に基づいた**システムテスト**を行う．

２.１．単体テスト

実装したクラスのメソッドを対象に，単体テストを行う．クラス数が多く，すべてのメソッドに対する単体テストを実施するのは煩雑だったため，コンストラクタ，フィールド群の setter/getter はテストの対象外とした．したがって，コンストラクタおよび setter/getter のみメソッドに持つ ArrivalRequest，OrderInformation，Shippedinformation クラスについての単体テストは省略する．

本節で扱うテストは，**Liquor**，**Stock**，**LiquorShop** クラスについての単体テストである．テストは，開発環境である eclipse（version 2018-09）と JUnit3 を用いている．

２.１.１．　Liquor クラス

Liquor クラスでテストするメソッドは以下のとおりである．

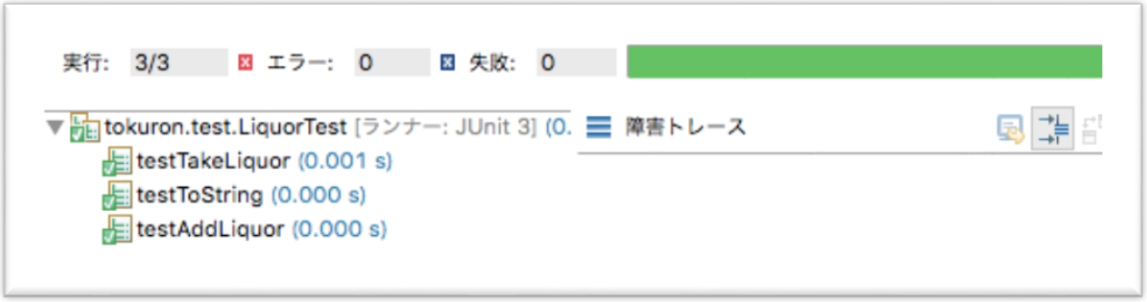
- **public String toString()**
  Liquor クラスを文字列に変換する．
- **public void takeLiquor(int count)**
  本数を引数にとり，酒の本数を減らす．
- **public void addLiquor(int count)**
  本数を引数にとり，酒の本数を増やす．

上記のメソッドの仕様をテストするために，以下のテストクラスを作成した．

LiquorTest.java

```
public class LiquorTest extends TestCase {
	private Liquor liquor1,liquor2;　//テスト用のインスタンス

	protected void setUp() throws Exception {
		this.liquor1=new Liquor("福寿",10);
		this.liquor2=new Liquor("祝米",6);
	}
	protected void tearDown() throws Exception {
		//後処理は特になし
	}

	public void testToString() {
		//toString では「銘柄名(tab)本数」という文字列が返る
		assertEquals("福寿　10",liquor1.toString());
	}

	public void testTakeLiquor() {
		//2 本出荷すれば，残りの本数は 8 本
		liquor1.takeLiquor(2);
		int count=liquor1.getCount();
		assertEquals(8,count);

		//在庫＜注文本数なら，本数に変化はない
		liquor1.takeLiquor(15);
		int count2=liquor1.getCount();
		assertEquals(8,count2);

		//境界値テスト　在庫＝注文本数の場合，残りの本数は 0 本
		liquor2.takeLiquor(6);
		int count3=liquor2.getCount();
		assertEquals(0,count3);
	}

	public void testAddLiquor() {
		//5 本入荷すれば，本数は 15 本
		liquor1.addLiquor(5);
		int count=liquor1.getCount();
		assertEquals(15,count);

		//0 本追加しても本数は変わらない
		liquor2.addLiquor(0);
		int count2=liquor2.getCount();
		assertEquals(6,count2);
	}
}
```

JUnit テストを行うと以下のような結果が得られた．すべてのテストメソッドが成功した．



２．１．２．Stock クラス

Stock クラスでテストするメソッドは以下のとおりである．

- **public boolean checkStock(OrderInformation order)**

  注文を引数にとり，在庫に銘柄が存在し，本数が十分にあれば
  注文で指定された分だけ本数を減らして ture を返す．
  そうでなければ false を返す．

- **public void addStock(String name, int count)**

  銘柄と本数を引数にとり，在庫に銘柄が存在すれば本数を追加する．
  在庫に銘柄が存在しなければ，新しく Liquor オブジェクトを生成し，
  在庫に追加する．

- **public void takeStock(String name, int count)**

  銘柄と本数を引数にとり，在庫に銘柄が存在すれば本数を減らす．
  在庫に銘柄が存在しなければ，何もしない．

- **public Liquor searchStock(String name)**

  銘柄を引数にとり，在庫にその銘柄が存在すれば，
  在庫の酒のリストから，その Liquor オブジェクトを返す．
  在庫にその銘柄が存在しなければ，Null を返す．

- **public String toString()**

  Stock クラスを文字列に変換する．

上記のメソッドの仕様をテストするために，以下のテストクラスを作成した．

StockTest.java

```
public class StockTest extends TestCase {
	Liquor yaegaki,banshu;   //テスト用の酒
	ArrayList<Liquor> stocklist;   //テスト用の酒のリスト
	OrderInformation order1,order2,order3,order4;   //テスト用の注文
	Stock stock;   //テスト用の Stock クラスのインスタンス

	protected void setUp() throws Exception {
		//酒の中身を設定しておく
		this.yaegaki=new Liquor("八重垣",12);
		this.banshu=new Liquor("播州一献",5);

		//在庫のリストを作成する．　在庫には八重垣と播州一献だけある
		this.stocklist=new ArrayList<Liquor>();
		this.stocklist.add(yaegaki);
		this.stocklist.add(banshu);

		//出荷できる注文
		this.order1=new OrderInformation("A さん","八重垣",5);
		//在庫数が足りない注文
		this.order2=new OrderInformation("B さん","播州一献",10);
		//銘柄が存在しない注文
		this.order3=new OrderInformation("A さん","奥播磨",5);
		//在庫＝注文本数の注文
		this.order4=new OrderInformation("B さん","播州一献",5);
```

```
		//stock が持つ酒の在庫リストに,上記の stocklist を設定する.
		this.stock=new Stock();
		stock.setStock(stocklist);
	}

	protected void tearDown() throws Exception {
		//後処理は特になし
	}

	/*銘柄があって在庫数がたりるときのみ true
	  銘柄があっても在庫数が足りなければ false
	  在庫に指定した銘柄がなければ false */
	public void testCheckStock() {
		assertTrue(stock.checkStock(order1));  //出荷できる注文
		assertFalse(stock.checkStock(order2)); //在庫不足で出荷できない注文
		assertFalse(stock.checkStock(order3));  //銘柄がなく出荷できない注文
		assertTrue(stock.checkStock(order4));  //在庫＝注文本数の注文
	}

	/* 在庫リストに存在しない酒の入荷と,
	   在庫リストにすでに存在する酒の入荷をテストする */
	public void testAddStock() {
		//在庫リストにすでにある酒を入荷する.
		stock.addStock("播州一献",10);
		//播州一献を 10 本追加したので，本数は 15 本
		assertEquals(15,(stock.searchStock("播州一献")).getCount());

		//在庫リストに存在しない酒を入荷する
		stock.addStock("龍力",20);
		assertNotNull(stock.searchStock("龍力"));  //null ではない
		assertEquals(20,(stock.searchStock("龍力")).getCount()); //本数は 20 本

		//0 本追加するなら，本数に変化はない
		stock.addStock("播州一献",0);
		assertEquals(15,(stock.searchStock("播州一献")).getCount());
	}

	/* 在庫リストに酒の銘柄が存在する場合について,
	   本数を指定して，その分だけ在庫の本数が減っていることを確認する */
	public void testTakeStock() {
		//銘柄が在庫リストに存在するものから酒を出荷する
		//八重垣から 5 本出荷すると，残りは 7 本
		stock.takeStock("八重垣",5);
		assertEquals(7,(stock.searchStock("八重垣")).getCount());

		//残りの本数すべてを出荷すると在庫は 0 本（境界値テスト）
		stock.takeStock("八重垣",7);
		assertEquals(0,(stock.searchStock("八重垣")).getCount());
	}


	/* 在庫リストに酒が存在すれば，その酒の Liquor オブジェクトを返す
	   在庫リストに酒が存在しなければ，null を返す */
	public void testSearchStock() {
		//八重垣は在庫リストに存在するので，Liquor オブジェクトが返る
		assertSame(yaegaki,stock.searchStock("八重垣"));

		//奥播磨という銘柄の酒は在庫リストに存在しないので，null が返る
		assertNull(stock.searchStock("奥播磨"));
	}

	//酒の在庫リストの内容を文字列で返せているか確認する
	public void testToString() {
		String line="酒銘柄¥t 本数(本)¥n"+"八重垣¥t12¥n"+"播州一献¥t5¥n";
```

```
                assertEquals(line,stock.toString());
        }
}
```

JUnit テストを行うと以下のような結果が得られた．すべてのテストメソッドが成功した．

2.1.3．LiquorShop クラス

LiquorShop クラスでテストするメソッドは以下のとおりである．

- **public boolean checkStock(OrderInformation order)**
  注文依頼を引数にとり，注文された酒の銘柄が在庫に登録されており，
  かつ，注文本数より多く存在するならば出荷処理を行い true を返す．
  そうでなければ false を返す．
- **public void addShippedInformation(ShippedInformation si)**
  出荷伝票を引数にとり，LiquorShop が持つ出荷伝票のリストに追加する．
- **public void addStock(ArrivalRequesti request)**
  入荷依頼を引数にとり，在庫に依頼の銘柄が存在すれば，
  指定された本数分を在庫に追加する．
  在庫に銘柄が存在しなければ，新しく Liquor オブジェクトを生成し，
  在庫に追加する．

ファイルの読み込みおよびファイルの保存を行う loadFile()，saveFile()メソッドについては，システムテストにてテストすることとした．

上記のメソッドの仕様をテストするために，以下のテストクラスを作成した

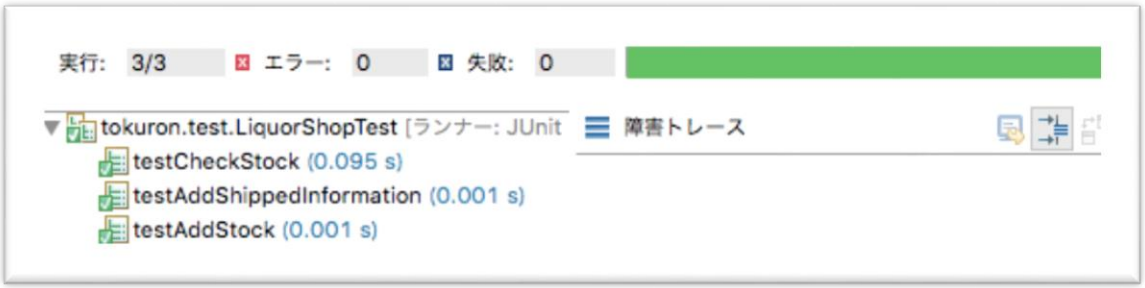
LiquorShopTest.java

```
public class LiquorShopTest extends TestCase {
	OrderInformation order1,order2,order3,order4;  //テスト用の注文
	ArrivalRequest request1,request2;  //テスト用の入荷依頼
	Stock stock;  //テスト用の在庫
	ArrayList<ShippedInformation> siList;  //テスト用の出荷伝票リスト
	ShippedInformation si;  //テスト用の出荷伝票
	LiquorShop ls; //テスト用の LiquorShop クラスのインスタンス

	protected void setUp() throws Exception {
		//LiquorShop を new して，ファイルを読み込む
		//在庫には「八重垣 12 本」「播州一献 5 本」のみが登録されている
		//出荷伝票には何も書かれていない
		this.ls=new LiquorShop();
		ls.loadFile();

		//LiquorShop の stock(在庫)と siLst(出荷伝票のリスト)取得
		this.stock=ls.getStock();
		this.siList=ls.getShippedInformation();

		//出荷できる注文
		this.order1=new OrderInformation("A さん","八重垣",5);
		//在庫数が足りない注文
		this.order2=new OrderInformation("B さん","播州一献",10);
		//銘柄が存在しない注文
		this.order3=new OrderInformation("A さん","奥播磨",5);
		//在庫＝注文本数の注文
		this.order4=new OrderInformation("B さん","播州一献",5);

		//在庫リストに存在する銘柄の入荷
		this.request1=new ArrivalRequest("播州一献",10);
		//在庫リストに存在しない銘柄の入荷
		this.request2=new ArrivalRequest("奥播磨",20);

		//出荷伝票の作成
		//銘柄:八重垣 本数:5 顧客名:A さん 注文番号:1
		this.si=new ShippedInformation("八重垣",5,"A さん",1);
	}

	protected void tearDown() throws Exception {
		//後処理は特になし
	}

	public void testCheckStock() {
		assertTrue(ls.checkStock(order1)); //出荷できる注文
		assertFalse(ls.checkStock(order2));  //在庫不足で出荷できない注文
		assertFalse(ls.checkStock(order3));  //銘柄がなく出荷できない注文
		assertTrue(ls.checkStock(order4)); //在庫＝注文本数の注文
	}

	public void testAddStock() {
		//在庫リストに存在する銘柄の入荷
		//播州一献の本数が 5+10=15 本になる
		ls.addStock(request1);
		assertEquals(15,(stock.searchStock("播州一献")).getCount());

		//在庫リストに存在しない銘柄の入荷
		//在庫リストに「奥播磨」という銘柄の酒が存在し，その本数は 20 本
		ls.addStock(request2);
		assertNotNull(stock.searchStock("奥播磨"));  //null でない
		assertEquals(20,(stock.searchStock("奥播磨")).getCount());
	}

	public void testAddShippedInformation() {
```

```
                    //出荷伝票を追加
                    //出荷伝票のリストにそれが含まれているか確かめる
                    ls.addShippedInformation(si);
                    assertTrue(siList.contains(si));
            }
}
```

LiquorShop の loadFile()では，stock.csv，shipped.csv というファイルから在庫および出荷伝票の読み込みが行われている．今回の単体テストでは，stock.csv に「八重垣 12 本」「播州一献 5 本」という二つの酒のみを予め登録しておき，テストに利用した．

JUnit テストを行うと以下のような結果が得られた．すべてのテストメソッドが成功した．



2.2．結合テスト

外部設計に基づいた結合テストを行う．酒屋在庫管理システムを運用するクラスである LiquorSystem クラスのメソッドごとに，外部設計を満たすことを確認する．

2.2.1．メニュー画面

LiquorSystem クラスのメソッド public static void menu()の実行画面は以下のようになる．外部設計と同じ画面出力であることが確認できた．

```
行いたい業務いずれかの数字を選択して下さい
システムを終了する場合は0を選択してください
0:システムを終了する
1:出荷する
2:入荷する
3:在庫表示を行う
4:出荷実績の表示を行う
```

2.2.2．出荷業務の画面

LiquorSystem クラスのメソッド public static void ship()の実行画面は以下のようになる．

```
顧客の名前を入力してください:鈴木
酒銘柄を入力してください:八重垣
本数を入力してください:2
```

在庫が存在する場合，次のように出荷伝票が表示される．

```
～出荷伝票～
注文番号  :1
名前      :鈴木
酒銘柄          :八重垣
本数      :2本


-------------------------
Enterキーでメニューに戻る
```

外部設計では，出荷伝票の下に「クリックでメインメニューに戻る」という指示があり，クリックによって出荷伝票発行画面からメニュー画面に戻ることが想定されていた．しかしながら，酒屋在庫管理システムはターミナル上のコンソールアプリケーションとして実現されていたため，クリックによる遷移ではなく，Enter キーを押すことによる遷移に変更している．

また，在庫が存在しない場合は，次のように表示される．

```
在庫がありません。申し訳ございません。
```

２.２.３．入荷業務の画面

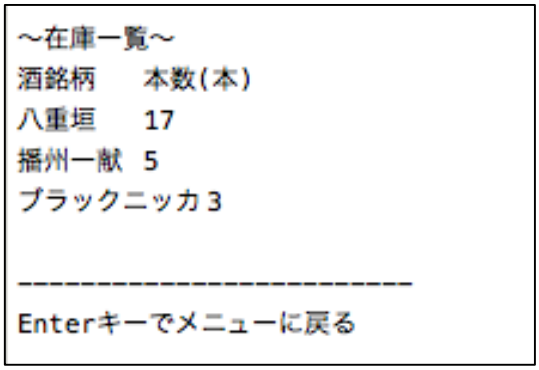
LiquorSystem クラスのメソッド public static void arrive()の実行画面は以下のようになる．外部設計と同じ画面出力であることが確認できた．

```
酒銘柄を入力してください:八重垣
本数を入力してください:5
```

```
酒銘柄を入力してください:ブラックニッカ
本数を入力してください:3
```

２.２.４．在庫表示の画面

LiquorSystem クラスのメソッド public static void viewStock()の実行画面は以下のようになる．

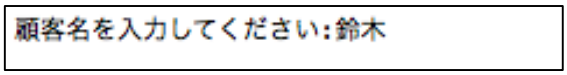


出荷伝票の表示画面と同様に，在庫表示画面からメニュー画面への遷移は，Enter キーを押すことでできるように変更している．また，２.２.３で入荷した酒の情報も反映されていることがわかる．

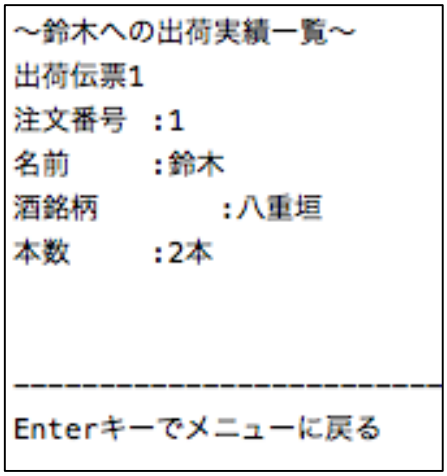
２.２.５．出荷実績の表示画面

LiquorSystem クラスのメソッド public viewShippedInformation()の実行画面は以下のようになる．

まず，システムから誰の出荷伝票の表示を行うか聞かれるため，顧客名を入力する．



顧客名を入力すると，その顧客の出荷伝票の一覧が表示される．



出荷伝票の表示画面と同様に，出荷実績表示画面からメニュー画面への遷移は，Enter キーを押すことでできるように変更している．

２.３．システムテスト

要求定義に基づいて，システムテストを行う．

～要求定義～

【酒の出荷】

・顧客名，酒の銘柄，本数を入力

・在庫があるか確認，ある場合出荷を行い，在庫を更新．出荷伝票が発行される

・在庫がない場合,「在庫不足」を表示して謝罪する

【酒の入荷】

・酒の銘柄，本数を入力

・同じ銘柄の酒が在庫にある場合，該当する銘柄の在庫本数に加算する

・ない場合，新たに銘柄と本数を在庫に追加する

【在庫表示】

・在庫内にある全ての酒の銘柄とその本数をリストにして表示する．

【出荷実績の表示】

・顧客名を入力

・指定の顧客に対して行ったすべての出荷の出荷伝票リストを表示する．

【システムを起動】

・ファイルからデータを読み込む．

【システムを終了】

・ファイルに在庫状況と出荷伝票を保存する．

【システムを起動】／【在庫表示】

まず，システムの起動テストを行った．事前に，在庫状況を格納するファイルとして図１に示すものを用意した． Ａ列が銘柄， Ｂ列が本数を表す．

	A	B
1	八重垣	10
2	播州一献	10
3	龍力	20
4	福寿	10
5		

図１　テスト前の在庫状況ファイル

出荷伝票用のファイルは空のファイルとした．この状態でシステムを起動させ，在庫表示を行うと次のように表示された．

～在庫一覧～

酒銘柄　本数(本)

八重垣　10

播州一献　　10

龍力　　20

福寿　　10

正しくファイルから在庫状況の読み込みができていることがわかる．

【酒の出荷】

　次に，出荷処理のテストを行った．出荷できる場合とできない場合についてテストを行うため，次の３つの入力を行った．

（１）顧客『平山』が『八重垣』を５本注文する

（２）顧客『平山』が『いいちこ』を５本注文する（銘柄の登録がない）

（３）顧客『平山』が『福寿』を 30 本注文する（数量が足りない）

　（１）の場合は，以下のような出荷伝票が出力される．

～出荷伝票～

注文番号　　:1

名前　　:平山

酒銘柄　　:八重垣

本数　　:5 本

この状態で在庫状況を確認すると次のように表示された．

～在庫一覧～
酒銘柄　本数(本)
八重垣　5
播州一献　　10
龍力　　20
福寿　　10

『八重垣』の本数が初期状態から 5 本減っており，正常に出荷処理が完了していることがわかる．

（２），（３）の場合，

在庫がありません。申し訳ございません。

というエラーメッセージが表示され，出荷処理が行われなかったことが確認できる．

【酒の入荷】

ここまでの出荷処理が終わった後に，入荷処理のテストを行った．

まず，在庫に登録のある銘柄の酒を追加する．『八重垣』を 10 本入荷するよう入力し，在庫状況の表示を行うと次のように表示された．

～在庫一覧～
酒銘柄　本数(本)
八重垣　15
播州一献　　10
龍力　　20
福寿　　10

『八重垣』が 10 本追加され，もとの 5 本から 15 本になった．

次に，在庫にない銘柄の酒を追加する．『ブラックニッカ』を 5 本入荷するよう入力し，在庫状況の表示を行うと次のように表示される．

～在庫一覧～
酒銘柄　本数(本)
八重垣　15
播州一献　　10
龍力　　20
福寿　　10
ブラックニッカ　5

新たに『ブラックニッカ』が在庫に追加され，本数が 5 本に設定された．これらから，正しく入荷処理が行われていることがわかる．

【出荷実績の表示】
次に，出荷実績の表示をテストした．『平山』への出荷実績を表示させると，次のように表示された．

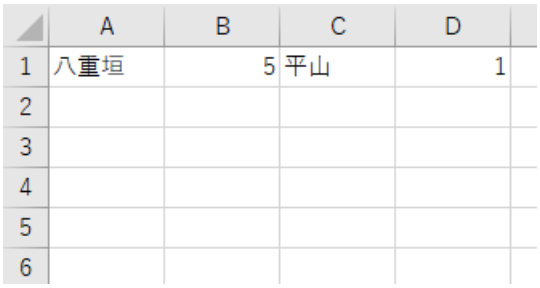
～平山への出荷実績一覧～
出荷伝票 1
注文番号　　:1
名前　　:平山
酒銘柄　　:八重垣
本数　　:5 本

先ほどの注文に対応した出荷伝票が正しく保存されていることがわかる．

【システムを終了】
最後に，システムを終了し，現在の在庫状況及び出荷伝票一覧が正しくファイルに保存されているかを確認した．システム終了後，在庫状況のファイルは図２，出荷伝票のファイルは図３のような状態となっていた．

	A	B
1	八重垣	15
2	播州一献	10
3	龍力	20
4	福寿	10
5	ブラックニッカ	5
6		

図２　終了時の在庫状況ファイル



	A	B	C	D
1	八重垣	5	平山	1
2				
3				
4				
5				
6				

図３　終了時の出荷伝票ファイル

図３は右から『酒銘柄』,『本数』,『(顧客の）名前』,『注文番号』に対応している．これらより，ファイルに正しく在庫状況と出荷伝票を保存できていることがわかる．この後，再びシステムを起動し，在庫表示および出荷実績の表示が正しく行われることも確認した．

以上のテストにより，要求定義によって必要とされた機能がすべて正しく実装できていることが確認できた．

３．コンパイル・実行に必要なドキュメント

今回作成した酒屋在庫管理システムは，統合開発環境の Eclipse 上で実装・テストしている．酒屋在庫管理システムをコマンドプロンプトで実行するには，作成したソースコードを実行可能な jar ファイルとしてまとめ，実行すればよい．本レポートには jar ファイルの添付をしていないが，jar ファイルに固めたものをコマンドプロンプト上で実行できることは確認済みである．

jar ファイル名を liquorSystem.jar とすれば，jar ファイルの実行は，当該ファイルが置いてあるディレクトリまで移動したのち，以下のコマンドを入力すればよい．

```
$  java -jar liquorSystem.jar
```

なお，外部設計仕様書には「システムの実行形式は liquorSystem という名前で，ターミナルで ./liquorSystem と入力することで実行する」と記載されていたが，我々はこれを jar ファイルでの実行と考えた．

システム上で利用される在庫のファイルおよび出荷伝票のファイルは，システムの起動時に存在しなかった場合，システムが自動的に生成することになっているため，特に用意する必要はない．予め在庫や出荷伝票のデータが存在しており，それらをシステムに反映させたい場合は，jar ファイルと同じディレクトリ内に「stock.csv」および「shipped.csv」というファイル名で，在庫ファイルと出荷伝票ファイルを作成しておけばよい．

４．上流工程評価アンケート

このアンケートはソフトウェア工学特論Ⅰのミニ酒屋問題における**内部設計と実装演習**の後に記述して提出してください．このアンケートは**グループで一枚**必ず提出してください．また，このグループ対象アンケートは，皆さんの**評価に影響を与えます**ので，正しく記述してください．

グループ名：生ハム　　グループリーダ名：三浦　稚咲

要求定義書・外部設計書のグループ名：A チーム

内部設計および実装では，顧客から与えられた要求にもとづいて作成された要求定義書・外部設計書に忠実であることが第一に求められます．本演習においても，内部設計・実装が**それまでの工程で作成された要求定義書・外部設計書の内容に準拠しているかどうか**が評価の対象となります．

そこで，今回の演習で利用した要求定義書・外部設計書を評価し，内部設計・実装を行う際に問題がなかったかどうかを記述してください．ここで問題がある等の記述が無ければ，要求定義書・外部設計書と内部設計・実装の間に**ずれ**が存在した場合に，それは**実装の際のグループに問題があったもの**と判断します．

(a)要求定義書・外部設計書の内容は理解しやすかったですか？

全く分からない　一部しか分からない　半分程度　ほぼ理解できた　完全に理解できた

5　4　3　②　1

(a') 選択肢が 1 以外である場合，その部位を指摘し，不明だった点をどのように解釈したのかを併せて明記しておくこと．この件に関しては，要求定義書・外部設計書の修正および修正箇所を指摘する資料を用意すること(アンケートとは別にメール等で提出してもよい)．

・システム起動時にファイルが存在しない場合の処理

→ファイルがなければ無視するものとし，存在しない旨を表示すると解釈した．

・酒の出荷時，在庫を減らす処理を行うタイミングが不明

→在庫を確認時，在庫が足りていれば在庫を減らすものとした．

・出荷実績の表示時，シーケンス図で顧客名の入力タイミングが不明

→出荷伝票リストを要求する前に入力するものとした．

・注文番号の決め方が一意であるとのみ書かれており，決め方が不明

→出荷伝票の数に応じた通し番号であるとした．

今回の要求定義書は……

(b)正しかったですか？　間違った要件が含まれていませんでしたか？

1. 正しい

2. 正しくない（もしくは恐らく正しくないと思われる項目が含まれている）

(b') 選択肢が 1 以外である場合，その部位を指摘し，正しくなかった点をどのように解釈したのかを併せて明記しておくこと．この件に関しては，要求定義書・外部設計書の修正および修正箇所を指摘する資料を用意すること(アンケートとは別にメール等で提出してもよい).

・外部設計では出荷伝票表示時などにクリックでメインメニューに戻る旨が記載されているが，c か java のコンソールアプリで実装する場合クリックのみで画面遷移をするのは不可能だと考えられる．

・論理データ設計ではデータ関連はないと記載されているが，酒と在庫など関連があるものは存在すると考えられる．

(c)一貫性がありましたか？　複数の要件の間に矛盾はありませんでしたか？

1. 一貫性があった

2. 一貫性がなかった（もしくはない部分があった）

(c') 選択肢が 1 以外である場合，その部位を指摘し，一貫性がなかった点をどのように解釈したのかを併せて明記しておくこと．この件に関しては，要求定義書・外部設計書の修正および修正箇所を指摘する資料を用意すること(アンケートとは別にメール等で提出してもよい).

(d)実現可能な事項のみが書かれていますか？

1. すべて実現可能である

2. 実現不可能である（もしくは実現可能だがコスト的，技術的に非常に困難である要件が含まれている）

(d') 選択肢が 1 以外である場合，その部位を指摘し，実現不可能な点をどのように解釈したのかを併せて明記しておくこと．この件に関しては，要求定義書・外部設計書の修正および修正箇所を指摘する資料を用意すること(アンケートとは別にメール等で提出してもよい).

・外部設計では，酒屋在庫管理システムをターミナル上で ./liquorSystem と入力することで実行すると記載されている

→jar ファイルによる実行であると解釈した．

・外部設計では出荷伝票表示時などにクリックでメインメニューに戻る旨が記載されている

→エンターキーを押すとメニューに戻ると解釈した．

(e)外部設計書の内容は要求定義書と対応が取れていましたか？

対応が全く取れていない　漏れが非常に多い　一部漏れが残ったまま　完全に過不足なく要求定義書と対応している

4　3　2　1

(e') 選択肢が 1 以外である場合，その部位を指摘し，対応の取れていなかった点をどのように解釈したのかを併せて明記しておくこと．この件に関しては，要求定義書・外部設計書の修正および修正箇所を指摘する資料を用意すること(アンケートとは別にメール等で提出してもよい).

(f)遷移情報と入出力情報を含めた画面設計は十分にできていますか？

1. 画面遷移も入出力情報も完全である

2. 画面遷移は記述したが，入出力情報に漏れがある（もしくはその逆）

3. 画面遷移も入出力情報も不十分

(f’）選択肢が 1 以外である場合，その部位を指摘し，不十分であった点をどのように解釈したのかを併せて明記しておくこと．この件に関しては，要求定義書・外部設計書の修正および修正箇所を指摘する資料を用意すること(アンケートとは別にメール等で提出してもよい).

(g)システム内で利用されるデータ構造について，論理データ設計やコード設計は十分でしたか？

1. 全てのデータ項目の定義とデータ間の関係も含めて十分に設計ができていた

2. データ項目の洗い出しはできていたがデータ間の関係が定義できていなかった

3. データ項目の洗い出しが不十分だった

(g’）選択肢が 1 以外である場合，その部位を指摘し，内部設計・実装の際に修正した点を併せて明記しておくこと．この件に関しては，要求定義書・外部設計書の修正および修正箇所を指摘する資料を用意すること(アンケートとは別にメール等で提出してもよい).

・論理データ設計では，データ関連が存在しない旨が記載されている

→クラス図に従って実装を行った

ソースコード　一覧

LiquorSystem.java

```
package tokuron;
import java.util.ArrayList;

import tokuron.liquor.ArrivalRequest;
import tokuron.liquor.OrderInformation;
import tokuron.liquorShop.LiquorShop;
import tokuron.liquorShop.ShippedInformation;;

public class LiquorSystem {

	private static boolean t = true;
	private static LiquorShop shop;

	public static void main(String[] args) {
		shop = new LiquorShop();
		shop.loadFile();
		while(t){
			menu();
		}
	}

	public static void menu(){
		System.out.println("-----------------------------------------------------");
		System.out.println("行いたい業務いずれかの数字を選択して下さい");
		System.out.println("システムを終了する場合は 0 を選択してください");
		System.out.println("0:システムを終了する");
		System.out.println("1:出荷する");
		System.out.println("2:入荷する");
		System.out.println("3:在庫表示を行う");
		System.out.println("4:出荷実績の表示を行う");
		System.out.println("-----------------------------------------------------");
		int input=-1;
		while(input<0||input>4){
			input = KeyBoard.inputNumber();
		}
		switch(input){
			case 1:
				ship();
				break;
			case 2:
				arrive();
				break;
			case 3:
				viewStock();
				break;
			case 4:
				viewShippedInformation();
				break;
			case 0:
```

```
                                        shutdown();
                                        break;
                        }
        }

        public static void ship(){
                        String customerName=new String(),liquorName = new String();
                        int count=-1,number;
                        while(count<=0){
                                System.out.println("-----------------------------------------------------");
                                System.out.print("顧客の名前を入力してください:");
                                customerName = KeyBoard.inputString();
                                //System.out.println();
                                System.out.print("酒銘柄を入力してください:");
                                liquorName = KeyBoard.inputString();
                                //System.out.println();
                                System.out.print("本数を入力してください:");
                                count = KeyBoard.inputNumber();
                                //System.out.println();
                                System.out.println("-----------------------------------------------------");
                                if(count<=0){
                                        System.out.println("本数は自然数で入力してください");
                                }
                        }
                        number=shop.getShippedInformation().size()+1;
                        OrderInformation order = new OrderInformation(customerName,liquorName,count);
                        if(shop.checkStock(order)){
                                ShippedInformation info = new
ShippedInformation(liquorName,count,customerName,number);
                                System.out.println("-----------------------------------------------------");
                                System.out.println("-------------------------");
                                System.out.println("～出荷伝票～");
                                System.out.println(info);
                                System.out.println("-------------------------");
                                shop.addShippedInformation(info);
                                System.out.println("Enter キーでメニューに戻る");
                                System.out.println("-----------------------------------------------------");
                                KeyBoard.inputString();
                        }else{
                                System.out.println("-----------------------------------------------------");
                                System.out.println("在庫がありません。申し訳ございません。");
                                System.out.println("-----------------------------------------------------");
                        }
        }

        public static void arrive(){
                        String name=new String();
                        int count=-1;
                        while(count<=0){
                                System.out.println("-----------------------------------------------------");
                                System.out.print("酒銘柄を入力してください:");
                                name = KeyBoard.inputString();
                                //System.out.println();
                                System.out.print("本数を入力してください:");
                                count = KeyBoard.inputNumber();
                                //System.out.println();
                                System.out.println("-----------------------------------------------------");
                                if(count<=0){
                                        System.out.println("本数は自然数で入力してください");
```

```
                        }
                }
                ArrivalRequest request = new ArrivalRequest(name,count);
                shop.addStock(request);
        }

        public static void viewStock(){
                System.out.println("----------------------------------------------------------");
                System.out.println("-------------------------");
                System.out.println("～在庫一覧～");
                System.out.println(shop.getStock());
                System.out.println("-------------------------");
                System.out.println("Enter キーでメニューに戻る");
                System.out.println("----------------------------------------------------------");
                KeyBoard.inputString();
        }

        public static void viewShippedInformation(){
                System.out.println("----------------------------------------------------------");
                System.out.print("顧客名を入力してください:");
                String name =KeyBoard.inputString();
                System.out.println();
                System.out.println("----------------------------------------------------------");

                ArrayList<ShippedInformation> list = shop.getShippedInformation();
                System.out.println("-------------------------");
                System.out.println("～"+name+"への出荷実績一覧～");
                int count =1;
                for(ShippedInformation info:list){
                        if(info.getCustomerName().equals(name)){
                                System.out.println("出荷伝票"+count);
                                System.out.println(info);
                                System.out.println();
                                count++;
                        }
                }
                System.out.println("-------------------------");
                System.out.println("Enter キーでメニューに戻る");
                KeyBoard.inputString();
        }

        public static void shutdown(){
                shop.saveFile();
                t=false;
        }
}
```

KeyBoard.java

```
package tokuron;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class KeyBoard {
        /**
         * 数字を入力する。
         */
```

```
	public static int inputNumber() {
		BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
		int number = -1;
		try {
			String line = br.readLine();
			number = Integer.parseInt(line);
		} catch (NumberFormatException e) {
			System.err.println("フォーマット例外です．もう一度．");
		} catch (IOException e) {
			System.err.println("入出力例外です．もう一度．");
		}
		return number;
	}

	/**
	 * 文字を入力する。
	 */
	public static String inputString() {
		BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
		String name = "";
		try {
			while(name == "") {
				String line = br.readLine();
				name = line;
			}
		} catch (NumberFormatException e) {
			System.err.println("フォーマット例外です．もう一度．");
			name = inputString();
		} catch (IOException e) {
			System.err.println("入出力例外です．もう一度．");
			name = inputString();
		}

		return name;
	}
}
```

LiquorShop.java

```
package tokuron.liquorShop;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.util.ArrayList;

import tokuron.liquor.ArrivalRequest;
import tokuron.liquor.Liquor;
import tokuron.liquor.OrderInformation;
import tokuron.liquor.Stock;

/**
 * 酒屋クラス
 * 在庫状況と出荷伝票のリストをフィールドに持つ
 * フィールドへのゲッター，在庫状況の更新や出荷伝票への追加，
 * ファイルから保存状況を読み出し，書き込みする機能を持つ
 * @author Kosuke HIRAYAMA
```

```
 *
 */
public class LiquorShop {
	/**
	 * 出荷伝票のリスト
	 */
	private ArrayList<ShippedInformation> shippedInformation = new ArrayList<ShippedInformation>();
	/**
	 * 在庫状況
	 */
	private Stock stock = new Stock();

	// 各種ファイル名
	private final String stockFileName = "stock.csv";
	private final String shipFileName = "shipped.csv";

	/**
	 * CSV ファイルから在庫状況を読み込む
	 */
	public void loadFile() {
		// 在庫状況を読み込み
		try {
			BufferedReader br = new BufferedReader(new InputStreamReader(new FileInputStream(stockFileName), "Shift-JIS"));

			String line;
			// 1 行ずつ CSV ファイルを読み込み
			while((line = br.readLine()) != null) {
				// コンマで分割する
				String[] elements = line.split(",");
				String name = elements[0];
				int count = Integer.parseInt(elements[1]);
				// 在庫状況に追加する
				this.stock.addStock(name, count);
			}
			br.close();
		} catch (FileNotFoundException e) {
			e.printStackTrace();
		} catch (IOException e) {
			e.printStackTrace();
		} catch (NumberFormatException e) {
			e.printStackTrace();
		}

		// 出荷伝票を読み込み
		try {
			BufferedReader br = new BufferedReader(new InputStreamReader(new FileInputStream(shipFileName), "Shift-JIS"));

			String line;
			// 1 行ずつ CSV ファイルを読み込み
			while((line = br.readLine()) != null) {
				String[] elements = line.split(",");
				String liquorName = elements[0];
				int count = Integer.parseInt(elements[1]);
				String customerName = elements[2];
				int orderNumber = Integer.parseInt(elements[3]);
				// 出荷伝票のリストに追加する
				ShippedInformation si = new ShippedInformation(liquorName, count, customerName, orderNumber);
				shippedInformation.add(si);
			}
			br.close();
		} catch (FileNotFoundException e) {
```

```
                        e.printStackTrace();
                } catch (IOException e) {
                        e.printStackTrace();
                } catch (NumberFormatException e) {
                        e.printStackTrace();
                }
        }

        /**
         * 現在の在庫状況および出荷伝票のリストをファイルに書き出す.
         * ファイルの内容は上書きされる.
         */
        public void saveFile() {
                // 在庫状況ファイルに書き出し
                // 在庫状況をリストに起こす
                ArrayList<Liquor> temp1 = this.stock.getStock();
                try {
                        // Excel で読み込めるように文字コードを Shift-JIS に指定
                        PrintWriter pw = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(new FileOutputStream(stockFileName), "Shift-JIS")));

                        // コンマ区切りで 1 行ずつ内容を保存
                        for(int i=0; i < temp1.size(); i++){
                                pw.print(temp1.get(i).getLiquorName());
                                pw.print(",");
                                pw.print(temp1.get(i).getCount());
                                pw.println();
                        }

                        pw.close();
                } catch (IOException e) {
                        // TODO 自動生成された catch ブロック
                        e.printStackTrace();
                }

                // 出荷伝票ファイルに書き出し
                try {
                        PrintWriter pw = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(new FileOutputStream(shipFileName), "Shift-JIS")));

                        for(int i=0; i < this.shippedInformation.size(); i++){
                                pw.print(this.shippedInformation.get(i).getLiquorName());
                                pw.print(",");
                                pw.print(this.shippedInformation.get(i).getCount());
                                pw.print(",");
                                pw.print(this.shippedInformation.get(i).getCustomerName());
                                pw.print(",");
                                pw.print(this.shippedInformation.get(i).getOrderNumber());
                                pw.println();
                        }

                        pw.close();
                } catch (IOException e) {
                        // TODO 自動生成された catch ブロック
                        e.printStackTrace();
                }
        }

        /**
         * 注文依頼を受け取り，出荷処理を行い，自身の持つ在庫状況を更新する.
         * 注文された酒の銘柄が，在庫に登録されており，かつ，注文本数よりも多く存在するならば,
出荷処理が行われる.
         * @param order 注文依頼
         * @return 正しく出荷処理が完了した場合は true
         */
```

```
	public boolean checkStock(OrderInformation order) {
		return this.stock.checkStock(order);
	}

	/**
	 * 出荷伝票を受け取り，自身の持つ出荷伝票のリストに追加する．
	 * @param si 出荷伝票
	 */
	public void addShippedInformation(ShippedInformation si){
		this.shippedInformation.add(si);
	}

	/**
	 * 入荷依頼を受け取り，入荷処理を行い，自身の持つ在庫状況を更新する．
	 * 依頼の銘柄の酒を指定された本数分，在庫に追加する．
	 * @param request 入荷依頼
	 */
	public void addStock(ArrivalRequest request) {
		String name = request.getLiquorName();
		int count = request.getCount();
		this.stock.addStock(name, count);
	}

	/**
	 * 自身の持つ在庫状況を取得させる．
	 * @return 在庫状況
	 */
	public Stock getStock() {
		return this.stock;
	}

	/**
	 * 自身の持つ出荷伝票のリストを取得させる．
	 * @return 出荷伝票
	 */
	public ArrayList<ShippedInformation> getShippedInformation() {
		return this.shippedInformation;
	}
}
```

ShippedInformation.java

```
package tokuron.liquorShop;

/**
 * 出荷伝票のクラス
 * 銘柄，本数，顧客名，注文番号をフィールドに持つ
 * @author yabuki
 *
 */
public class ShippedInformation {
	/**
	 * 銘柄
	 */
	private String liquorName;
	/**
	 * 本数
	 */
	private int count;
	/**
	 * 顧客名
	 */
```

```
	private String customerName;
	/**
	 * 注文番号
	 */
	private int orderNumber;

	/**
	 * コンストラクタ
	 * @param liquor
	 * @param count
	 * @param customer
	 * @param orderNumber
	 */
	public ShippedInformation(String liquor,int count,String customer,int orderNumber) {
		this.liquorName = liquor;
		this.count = count;
		this.customerName = customer;
		this.orderNumber = orderNumber;
	}

	/**
	 * 出荷伝票を文字列にする関数
	 */
	@Override
	public String toString(){
		return "注文番号¥t:"+this.orderNumber+"¥n 名前¥t¥t:"+this.customerName+"¥n 酒
銘柄¥t¥t:"+this.liquorName+"¥n 本数¥t¥t:"+this.count+"本¥n";
	}

	/**
	 * 銘柄の getter
	 * @return 銘柄
	 */
	public String getLiquorName() {
		return liquorName;
	}

	/**
	 * 銘柄の setter
	 * @param liquorName
	 */
	public void setLiquorName(String liquorName) {
		this.liquorName = liquorName;
	}

	/**
	 * 本数の getter
	 * @return 本数
	 */
	public int getCount() {
		return count;
	}

	/**
	 * 本数の setter
	 * @param count
	 */
	public void setCount(int count) {
		this.count = count;
	}

	/**
	 * 顧客名の getter
	 * @return 顧客名
	 */
```

```
	public String getCustomerName() {
		return customerName;
	}

	/**
	 * 顧客名の setter
	 * @param customerName
	 */
	public void setCustomerName(String customerName) {
		this.customerName = customerName;
	}

	/**
	 * 注文番号の getter
	 * @return 注文番号
	 */
	public int getOrderNumber() {
		return orderNumber;
	}

	/**
	 * 注文番号の setter
	 * @param orderNumber
	 */
	public void setOrderNumber(int orderNumber) {
		this.orderNumber = orderNumber;
	}
}
```

ArrivalRequest.java

```
package tokuron.liquor;

/**
 * 入荷依頼のクラス
 * 銘柄と本数をフィールドに持つ
 * @author yabuki
 *
 */
public class ArrivalRequest {
	/**
	 * 銘柄
	 */
	private String liquorName;
	/**
	 * 本数
	 */
	private int count;

	/**
	 * コンストラクタ
	 * @param name
	 * @param count
	 */
	public ArrivalRequest(String name,int count) {
		this.liquorName=name;
		this.count=count;
	}

	/**
	 * 銘柄の getter
	 * @return 銘柄
```

```
	 */
	public String getLiquorName() {
		return liquorName;
	}

	/**
	 * 銘柄の setter
	 * @param liquorName
	 */
	public void setLiquorName(String liquorName) {
		this.liquorName = liquorName;
	}

	/**
	 * 本数の getter
	 * @return 本数
	 */
	public int getCount() {
		return count;
	}

	/**
	 * 本数の setter
	 * @param count
	 */
	public void setCount(int count) {
		this.count = count;
	}
}
```

Liquor.java

```
package tokuron.liquor;

/**
 * 酒の銘柄と本数を持つクラス
 * 本数を増減するメソッドを持つ
 * @author yabuki
 *
 */
public class Liquor {
	/**
	 * 銘柄
	 */
	private String liquorName;
	/**
	 * 本数
	 */
	private int count;

	/**
	 * 銘柄と本数を引数にしたコンストラクタ
	 * @param name
	 * @param count
	 */
	public Liquor(String name,int count) {
		this.liquorName=name;
		this.count=count;
	}

	/**
	 * Liquor クラスを文字列に変換する
```

```
	 */
	public String toString() {
		return this.liquorName+"¥t"+count;
	}

	/**
	 * 本数を引数に，酒の本数を減らす関数
	 * @param count
	 */
	public void takeLiquor(int count){
		if(this.count>=count){
			this.count-=count;
		}
	}

	/**
	 * 本数を引数に，酒の本数を加える関数
	 * @param count
	 */
	public void addLiquor(int count){
		this.count+=count;
	}

	/**
	 * 銘柄の getter
	 * @return 銘柄
	 */
	public String getLiquorName() {
		return liquorName;
	}

	/**
	 * 銘柄の setter
	 * @param liquorName
	 */
	public void setLiquorName(String liquorName) {
		this.liquorName = liquorName;
	}

	/**
	 * 本数の getter
	 * @return 本数
	 */
	public int getCount() {
		return count;
	}

	/**
	 * 本数の setter
	 * @param count
	 */
	public void setCount(int count) {
		this.count = count;
	}
}
```

OrderInformation.java

```
package tokuron.liquor;

/**
 * 注文情報のクラス
```

```
 * 顧客名，銘柄，本数をフィールドに持つ
 * @author yabuki
 *
 */
public class OrderInformation {
	/**
	 * 顧客名
	 */
	private String customerName;
	/**
	 * 銘柄
	 */
	private String liquorName;
	/**
	 * 本数
	 */
	private int count;

	/**
	 * コンストラクタ
	 * @param customer
	 * @param liquor
	 * @param count
	 */
	public OrderInformation(String customer,String liquor,int count) {
		this.customerName=customer;
		this.liquorName = liquor;
		this.count = count;
	}

	/**
	 * 顧客名の getter
	 * @return 顧客名
	 */
	public String getCustomerName() {
		return customerName;
	}

	/**
	 * 顧客名の setter
	 * @param customerName
	 */
	public void setCustomerName(String customerName) {
		this.customerName = customerName;
	}

	/**
	 * 銘柄の getter
	 * @return 銘柄
	 */
	public String getLiquorName() {
		return liquorName;
	}

	/**
	 * 銘柄の setter
	 * @param liquorName
	 */
	public void setLiquorName(String liquorName) {
		this.liquorName = liquorName;
	}

	/**
	 * 本数の getter
	 * @return 本数
```

```
	 */
	public int getCount() {
		return count;
	}

	/**
	 * 本数の setter
	 * @param count
	 */
	 public void setCount(int count) {
		this.count = count;
	}
}
```