1.Реализовать указанный класс, включив в него вспомогательный внутренний класс или классы.

Создать класс Payment (покупка) с внутренним классом, с помощью объектов которого можно сформировать покупку из нескольких товаров.

Код программы

Main:

```
package com.company;

public class 0Main {


    public static void main(String[] args) {
        Menu menu = new Menu();
        menu.start();
    }
}
```

Menu:

```
package com.company;


import java.util.Scanner;


public class Menu {
    public void start(){
        Payment payment = new Payment();
        payment.fill();


        Scanner scanner = new Scanner(System.in);


        boolean key = true;
```

```
int code;
int id;

while(key){
    for(int i=0; i<15; i++)
        System.out.println();

    System.out.println("0: список товаров");
    System.out.println("1: моя покупка");
    System.out.println("2: добавить товар в покупку");
    System.out.println("3: удалить товар из покупки");
    System.out.println("4 - выйти");

    code = Integer.parseInt(scanner.nextLine());

    if(code==0) {
        payment.showTovar(false);
    }
    else if(code==1) {
        payment.showTovar(true);
    }
    else if(code==2) {
        System.out.println("id товара:");
        id = Integer.parseInt(scanner.nextLine());
        System.out.println("Количество товара:");
        int price = Integer.parseInt(scanner.nextLine());
        payment.addTovar(id, price);
    }
```

```
            else if(code==3) {
                System.out.println("id товара:");
                id = Integer.parseInt(scanner.nextLine());
                payment.deleteTovar(id);
            }
            else {
                key = false;
            }

            System.out.println("Нажмите Enter");
            scanner.nextLine();
        }
    }
}
```

Payment:

```
package com.company;

import java.util.ArrayList;

public class Payment{

    private ArrayList<Tovar> tovarList;
    private ArrayList<Tovar> myTovarList;

    class Tovar{
        private int id;
        private String name;
```

```
    private double price;
    private Integer count=null;

    public Tovar(int id, String name, double price){
        this.id=id;
        this.name=name;
        this.price=price;
    }

    public void show(){
        System.out.print("id="+id+"; name="+name+"; cost="+price);

        if(count!=null)
            System.out.println(", count="+count+", sum: "+price*count);
        else
            System.out.println();
    }

    public Tovar copy(){
        Tovar tovar = new Tovar(id, name, price);
        tovar.count = count;
        return tovar;
    }
}

public Payment(){
    tovarList = new ArrayList<Tovar>();
    myTovarList = new ArrayList<Tovar>();
```

```
}

public void fill(){
    tovarList.add(new Tovar(15, "Масло сливочное", 3.1));
    tovarList.add(new Tovar(68, "Шампиньоны", 5.7));
    tovarList.add(new Tovar(23, "Морковь 1 кг", 2.4));
    tovarList.add(new Tovar(14, "Конфеты Микс 500г", 3.83));
    tovarList.add(new Tovar(10, "Мороженое Юкки", 1.76));
}

public void addTovar(int id, int count){
    if(count<1)
        return;

    for(int i=0; i<myTovarList.size(); i++)
        if(myTovarList.get(i).id == id)
        {
            myTovarList.get(i).count += count;
            return;
        }

    for(int i=0; i<tovarList.size(); i++)
        if(tovarList.get(i).id == id)
        {
            Tovar tovar = tovarList.get(i).copy();
            tovar.count = count;
            myTovarList.add(tovar);
            return;
```

```
        }
    }

    public void deleteTovar(int id){
        for(int i=0; i<myTovarList.size(); i++)
            if(myTovarList.get(i).id == id) {
                myTovarList.remove(i);
                return;
            }
    }

    public void showTovar(boolean my){
        if(my)
            for(int i=0; i<myTovarList.size(); i++)
                myTovarList.get(i).show();
        else
            for(int i=0; i<tovarList.size(); i++)
                tovarList.get(i).show();
    }

}
```

Результат

```
0: список товаров
1: моя покупка
2: добавить товар в покупку
3: удалить товар из покупки
4 - выйти
0
id=15; name=Масло сливочное; cost=3.1
id=68; name=Шампиньоны; cost=5.7
id=23; name=Морковь 1 кг; cost=2.4
id=14; name=Конфеты Микс 500г; cost=3.83
id=10; name=Мороженое Юкки; cost=1.76
Нажмите Enter
```

```
0: список товаров
1: моя покупка
2: добавить товар в покупку
3: удалить товар из покупки
4 - выйти
2
id товара:
14
Количество товара:
3
Нажмите Enter
```

```
0: список товаров
1: моя покупка
2: добавить товар в покупку
3: удалить товар из покупки
4 - выйти
2
id товара:
15
Количество товара:
2
Нажмите Enter
```

```
0: список товаров
1: моя покупка
2: добавить товар в покупку
3: удалить товар из покупки
4 - выйти
1
id=15; name=Масло сливочное; cost=3.1, count=2, sum: 6.2
id=14; name=Конфеты Микс 500г; cost=3.83, count=3, sum: 11.49
Нажмите Enter
```

2. Реализовать агрегирование. При создании класса агрегируемый класс объявляется как атрибут

(локальная переменная, параметр метода). Включить в каждый класс 2-3 метода на выбор. Продемонстрировать использование разработанных классов.

Код программы

Planet:

```
package com.company;


import java.util.ArrayList;


class Planet{
    String name;
    int size;
    String color;


    public String getName() {
        return name;
    }


    public void setName(String name) {
        this.name = name;
    }


    public int getSize() {
        return size;
    }
```

```
    public void setSize(int size) {
        this.size = size;
    }

    public String getColor() {
        return color;
    }

    public void setColor(String color) {
        this.color = color;
    }

    public Planet(String name, int size, String color){
        this.name = name;
        this.size = size;
        this.color = color;
    }

    public void show(){
        System.out.println("Планета "+name+", цвет: "+color+", размер: "+size);
    }
}
```

Star:

```
class Star{
    int size;
    int age;
```

```
    public int getSize() {
        return size;
    }

    public void setSize(int size) {
        this.size = size;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    public Star(int size, int age){
        this.size = size;
        this.age = age;
    }
    public void show(){
        System.out.println("Звезда размером "+size+" и возрастом "+age);
    }
}
```

StarSystem:

```
class StarSystem{
    ArrayList<Planet> planets;
    ArrayList<Star> stars;

    public StarSystem(){
```

```
    planets = new ArrayList<Planet>();
    stars = new ArrayList<Star>();
}

public void addPlanet(String name, int size, String color){
    planets.add(new Planet(name, size, color));
}

public void addStar(int size, int age){
    stars.add(new Star(size, age));
}

public void removePlanets(){
    planets.clear();
}

public void removeStars(){
    stars.clear();
}

public void show(){
    System.out.println("Состав звёздной системы:");
    if(planets.size()==0&&stars.size()==0)
        System.out.println("пусто");
    else
    {
        for(int i=0; i<planets.size(); i++)
            planets.get(i).show();
```

```
            for(int i=0; i<stars.size(); i++)
                stars.get(i).show();
        }
    }
}
```

Main:

```
public class Main {

    public static void main(String[] args) {
        StarSystem system = new StarSystem();

        system.addPlanet("Меркурий", 250000, "оранжевый");
        system.removePlanets();
        system.show();

        system.addPlanet("Венера", 400000, "красный");
        system.addPlanet("Сатурн", 735002, "серебристый");

        system.addStar(500, 1000000);
        system.addStar(350, 780000);
        system.addStar(800, 5450000);

        system.show();

    }
}
```

Результат

```
Состав звёздной системы:
пусто
Состав звёздной системы:
Планета Венера, цвет: красный, размер: 400000
Планета Сатурн, цвет: серебристый, размер: 735002
Звезда размером 500 и возрастом 1000000
Звезда размером 350 и возрастом 780000
Звезда размером 800 и возрастом 5450000
```

Создать класс Звездная система, используя классы Планета, Звезда.

3. Построить модель программной системы с применением отношений (обобщения, агрегации, ассоциации, реализации) между классами. Задать атрибуты и методы классов. Реализовать (если необходимо) дополнительные классы. Продемонстрировать работу разработанной системы.

Система Аэрофлот. Администратор формирует летную Бригаду (пилоты, штурман, радист, стюардессы) на Рейс. Каждый Рейс выполняется Самолетом с определенной вместимостью и дальностью полета. Рейс может быть отменен из-за погодных условий в Аэропорту отлета или назначения. Аэропорт назначения может быть изменен в полете из-за технических неисправностей, о которых сообщил командир.

Код программы

Administrator:

```
package com.company;

import java.util.ArrayList;

public class Administrator extends Worker {

    public Brigade createBrigade(ArrayList<Pilot> pilots, Navigator navigator, Radioman
radioman, Stewardess stewardess){
        return new Brigade(pilots, navigator, radioman, stewardess);
    }

    public Cruise createCruise(Brigade brigade, Plane plane, Aeroport originalAeroport, Aeroport
destinationAeroport){
        return new Cruise(brigade, plane, originalAeroport, destinationAeroport);
    }
```

```
    public Cruise askChangeDestinationAeroport(Cruise cruise, Aeroport destinationAeroport){
        cruise.setDestinationAeroport(destinationAeroport);
        return cruise;
    }

    public boolean cancelCruise(Cruise cruise){
        if(cruise.getStatus()==0) {
            cruise.setStatus(2);
            return true;
        }
        else{
            return false;
        }
    }

    public void work(){
        System.out.println("Администратор работает!");
    }

    public void show(){
        System.out.println("Администратор:");
        super.show();
    }

    public Administrator(int id, String fio) {
        super(id, fio);
    }
}
```

Aeroport:

```
package com.company;

public class Aeroport {
    private String country;
    private String name;

    public String getCountry() {
        return country;
    }

    public void setCountry(String country) {
        this.country = country;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void show(){
```

```
        System.out.println("country: "+country);
        System.out.println("name: "+name);
    }

    public Aeroport(String country, String name) {
        this.country = country;
        this.name = name;
    }
}
```

AeroSystem:

```
package com.company;

import java.util.ArrayList;

public class AeroSystem {

    private ArrayList<Cruise> cruises = new ArrayList<Cruise>();


    public void addCruise(Cruise cruise){
        cruises.add(cruise);
    }

    public void showCruises(){
        for(int i=0; i<cruises.size(); i++)
            cruises.get(i).show();
    }
}
```

Brigade:

```
package com.company;

import java.util.ArrayList;

public class Brigade {
    private ArrayList<Pilot> pilots = new ArrayList<Pilot>();
    private Navigator navigator;
    private Radioman radioman;
    private Stewardess stewardess;

    public void show(){
        for(int i=0; i<pilots.size(); i++)
            pilots.get(i).show();
        navigator.show();
        radioman.show();
        stewardess.show();
    }

    public ArrayList<Pilot> getPilots() {
        return pilots;
    }

    public void setPilots(ArrayList<Pilot> pilots) {
```

```
        this.pilots = pilots;
    }

    public Navigator getNavigator() {
        return navigator;
    }

    public void setNavigator(Navigator navigator) {
        this.navigator = navigator;
    }

    public Radioman getRadioman() {
        return radioman;
    }

    public void setRadioman(Radioman radioman) {
        this.radioman = radioman;
    }

    public Stewardess getStewardess() {
        return stewardess;
    }

    public void setStewardess(Stewardess stewardess) {
        this.stewardess = stewardess;
    }

    public Brigade(ArrayList<Pilot> pilots, Navigator navigator, Radioman radioman, Stewardess
stewardess) {
        this.pilots = pilots;
        this.navigator = navigator;
        this.radioman = radioman;
        this.stewardess = stewardess;
    }
}
```

Chassis:

```
package com.company;

public class Chassis {
    private String producer;
    private int size;

    public String getProducer() {
        return producer;
    }

    public void setProducer(String producer) {
        this.producer = producer;
    }

    public int getSize() {
        return size;
```

```
    }

    public void setSize(int size) {
        this.size = size;
    }

    public void show(){
        System.out.println("Шасси: ");
        System.out.println("producer: "+producer);
        System.out.println("size: "+size);
    }

    public Chassis(String producer, int size) {
        this.producer = producer;
        this.size = size;
    }
}
```

Cruise:

```
package com.company;

public class Cruise {

    private Brigade brigade;
    private Plane plane;
    private Aeroport originalAeroport;
    private Aeroport destinationAeroport;
    private int status; //0 - готовится, 1 - в полете, 2 - отменен, 3 - завершен

    public void show(){
        System.out.println("Рейс: ");
        System.out.println("Бригада: ");
        brigade.show();
        System.out.println("Самолет: ");
        plane.show();
        System.out.println("Аэропорт отлета: ");
        originalAeroport.show();
        System.out.println("Аэропорт назначения: ");
        destinationAeroport.show();
        System.out.print("Статус: ");

        if(status==0)
            System.out.println("готовится к вылету");
        else if(status==1)
            System.out.println("в полете");
        else if(status==2)
            System.out.println("отменен");
        else if(status==3)
            System.out.println("завершен");
    }

    public Brigade getBrigade() {
        return brigade;
```

```
    }

    public void setBrigade(Brigade brigade) {
        this.brigade = brigade;
    }

    public Plane getPlane() {
        return plane;
    }

    public void setPlane(Plane plane) {
        this.plane = plane;
    }

    public Aeroport getOriginalAeroport() {
        return originalAeroport;
    }

    public void setOriginalAeroport(Aeroport originalAeroport) {
        this.originalAeroport = originalAeroport;
    }

    public Aeroport getDestinationAeroport() {
        return destinationAeroport;
    }

    public void setDestinationAeroport(Aeroport destinationAeroport) {
        this.destinationAeroport = destinationAeroport;
    }

    public int getStatus() {
        return status;
    }

    public void setStatus(int status) {
        this.status = status;
    }


    public Cruise(Brigade brigade, Plane plane, Aeroport originalAeroport, Aeroport
destinationAeroport) {
        this.brigade = brigade;
        this.plane = plane;
        this.originalAeroport = originalAeroport;
        this.destinationAeroport = destinationAeroport;
        this.status=0;
    }
}
```

IWorker:

```
package com.company;

public interface IWorker {
```

```
    void work();
    void show();
}
```

Main:

```
package com.company;

import java.util.ArrayList;

public class Main {

    public static void main(String[] args) {
        AeroSystem system = new AeroSystem();

        Administrator administrator = new Administrator(123, "Серый А.А.");

        Pilot pilot1 = new Pilot(12334, "Белый В.С.", administrator);
        Pilot pilot2 = new Pilot(13421, "Цветочкин А.М.", administrator);
        ArrayList<Pilot> pilots = new ArrayList<Pilot>();
        pilots.add(pilot1);
        pilots.add(pilot2);

        Navigator navigator = new Navigator(67, "Курочкин М.М.", 5);
        Radioman radioman = new Radioman(412, "Радионов Я.И.", "63.2");
        Stewardess stewardess = new Stewardess(326, "Ромашкина А.Ю", 1);

        Brigade brigade = administrator.createBrigade(pilots, navigator, radioman, stewardess);

        Chassis chassis = new Chassis("ОАО АвиаСтрой", 100);
        Plane plane = new Plane(chassis, 150, 3000);

        Aeroport originalAeroport = new Aeroport("Беларусь", "Минск-2");
        Aeroport destinationAeroport = new Aeroport("Россия", "Внуково");

        Cruise cruise = administrator.createCruise(brigade, plane, originalAeroport,
destinationAeroport);

        pilot1.setCruise(cruise);
        pilot2.setCruise(cruise);

        system.addCruise(cruise);

        administrator.cancelCruise(cruise);

        system.showCruises();

        cruise.setStatus(0);//только для демонстрации функций

        Aeroport aeroport3 = new Aeroport("Украина", "Киев-1");
        pilot2.callAdministratorChangeAeroportDestination(aeroport3);

        for(int i=0; i<10; i++)
            System.out.println();
```

```
        system.showCruises();


    }
}
```

Navigator:

```
package com.company;

public class Navigator extends Worker {
    private int experience;

    public int getExperience() {
        return experience;
    }

    public void setExperience(int experience) {
        this.experience = experience;
    }

    public void work(){
        System.out.println("Штурман работает!");
    }

    public void show(){
        System.out.println("Штурман:");
        super.show();
        System.out.println("experience: "+experience);
    }

    public Navigator(int id, String fio, int experience){
        super(id, fio);
        this.experience = experience;
    }
}
```

Pilot:

```
package com.company;

public class Pilot extends Worker {

    private Administrator administrator = null;
    private Cruise cruise = null;

    public Administrator getAdministrator() {
        return administrator;
    }

    public void setAdministrator(Administrator administrator) {
        this.administrator = administrator;
    }

    public Cruise getCruise() {
```

```
        return cruise;
    }

    public void setCruise(Cruise cruise) {
        this.cruise = cruise;
    }

    public void callAdministratorChangeAeroportDestination(Aeroport destinationAeroport){
        if(administrator!=null&&cruise!=null){
            administrator.askChangeDestinationAeroport(cruise, destinationAeroport);
        }
    }

    public void work(){
        System.out.println("Пилот работает!");
    }

    public void show(){
        System.out.println("Пилот:");
        super.show();
    }

    public Pilot(int id, String fio, Administrator administrator){
        super(id, fio);
        this.administrator = administrator;
    }
}
```

Plane:

```
package com.company;

public class Plane {
    private Chassis chassis;
    private int capacity;
    private int flightRange;

    public Chassis getChassis() {
        return chassis;
    }

    public void setChassis(Chassis chassis) {
        this.chassis = chassis;
    }

    public int getCapacity() {
        return capacity;
    }

    public void setCapacity(int capacity) {
        this.capacity = capacity;
    }

    public int getFlightRange() {
```

```
        return flightRange;
    }

    public void setFlightRange(int flightRange) {
        this.flightRange = flightRange;
    }

    public Plane(Chassis chassis, int capacity, int flightRange) {
        this.chassis = chassis;
        this.capacity = capacity;
        this.flightRange = flightRange;
    }

    public void show(){
        chassis.show();
        System.out.println("capacity: "+capacity);
        System.out.println("flightRange: "+flightRange);
    }
}
```

Radioman:

```
package com.company;

public class Radioman extends Worker{

    private String radioRange;

    public String getRadioRange() {
        return radioRange;
    }

    public void setRadioRange(String radioRange) {
        this.radioRange = radioRange;
    }

    public void work(){
        System.out.println("Радист работает!");
    }

    public void show(){
        System.out.println("Радист:");
        super.show();
        System.out.println("radioRange: "+radioRange);
    }

    public Radioman(int id, String fio, String radioRange){
        super(id, fio);
        this.radioRange = radioRange;
    }
}
```

Stewardess:

```
package com.company;
```

```
public class Stewardess extends Worker {
    private int category;

    public int getCategory() {
        return category;
    }

    public void setCategory(int category) {
        this.category = category;
    }

    public void work(){
        System.out.println("Стюардесса работает!");
    }

    public void show(){
        System.out.println("Стюардесса:");
        super.show();
        System.out.println("category: "+category);
    }

    public Stewardess(int id, String fio, int category){
        super(id, fio);
        this.category = category;
    }
}
```

Worker:

```
package com.company;

public class Worker implements IWorker {
    private int id;
    private String fio;

    public void work(){
        System.out.println("Работник работает!");
    }

    public void show(){
        System.out.println("id: "+id);
        System.out.println("fio: "+fio);
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getFio() {
        return fio;
```

```
    }

    public void setFio(String fio) {
        this.fio = fio;
    }

    public Worker(int id, String fio){
        this.id = id;
        this.fio = fio;
    }
}
```

Результат

```
Рейс:
Бригада:
Пилот:
id: 12334
fio: Белый В.С.
Пилот:
id: 13421
fio: Цветочкин А.М.
Штурман:
id: 67
fio: Курочкин М.М.
experience: 5
Радист:
id: 412
fio: Радионов Я.И.
radioRange: 63.2
Стюардесса:
id: 326
fio: Ромашкина А.Ю
category: 1
Самолет:
Шасси:
producer: ОАО АвиаСтрой
size: 100
capacity: 150
flightRange: 3000
Аэропорт отлета:
country: Беларусь
name: Минск-2
Аэропорт назначения:
country: Украина
name: Киев-1
Статус: готовится к вылету
```

```
Рейс:
Бригада:
Пилот:
id: 12334
fio: Белый В.С.
Пилот:
id: 13421
fio: Цветочкин А.М.
Штурман:
id: 67
fio: Курочкин М.М.
experience: 5
Радист:
id: 412
fio: Радионов Я.И.
radioRange: 63.2
Стюардесса:
id: 326
fio: Ромашкина А.Ю
category: 1
Самолет:
Шасси:
producer: ОАО АвиаСтрой
size: 100
capacity: 150
flightRange: 3000
Аэропорт отлета:
country: Беларусь
name: Минск-2
Аэропорт назначения:
country: Россия
name: Внуково
Статус: отменен
```