МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Российский государственный университет им. А.Н. Косыгина   
(Технологии. Дизайн. Искусство)»

Кафедра автоматизированных систем обработки информации и управления

Отчет по лабораторной работе № 1

по дисциплине «Программирование»

Тема: «Создание собственных классов»

Выполнил: Меньков С. А., МВА-18

Проверил: Самойлова Т. А.

Москва 2019

# **Цель работы**

Изучить основные понятия объектно-ориентированного программирования в языке Java, приобрести навыки разработки классов, создания в классах методов, а также их переопределения в наследуемых классах, реализации объектов

**Задание**

Грузоперевозчик. Подклассы Самолет, Корабль. Определить стоимость перевозки заданного объема груза каждым видом транспортного средства. Вывести информацию о транспортных средствах, способных перевезти заданный объем груза.

# **Наследование**

Текст программы:

public class Test01 {

public static void main(String[] args) {

Carrier[] carriers = new Carrier[7];

carriers[0] = new Ship(new Point2D(33.4, 55.3), 3, "Ship", 99.4, "Ship cargo", new Date());

carriers[1] = new Airplane("RU4277773433", 3, "Airplane", 9.4, "Air rout 01", new Date());

carriers[2] = new Airplane("ZA4343223433", 3, "Airplane", 11.4, "Air rout 02", new Date());

carriers[3] = new Airplane("LL4343223433", 3, "Airplane", 3.4, "Air rout 03", new Date());

carriers[4] = new Ship(new Point2D(63.4, 23.3), 3, "Ship", 909.4, "", new Date());

carriers[5] = new Airplane("RU4345223433", 3, "Airplane", 349.4, "Air rout 04", new Date());

carriers[6] = new Ship(new Point2D(3.4, 52.3), 3, "Ship", 22.4, "", new Date());

Scanner scanner = new Scanner(System.in);

System.out.print("Enter volume (T):");

double volume = scanner.nextDouble();

System.out.println("Extending...");

double minPlane = 0;

double minShip = 0;

for (Carrier c : carriers) {

if (c.getMaxVolume() >= volume) {

if (c instanceof Airplane && (c.getPrice(volume) <= minPlane || minPlane == 0))

minPlane = c.getPrice(volume);

else if (c instanceof Ship && (c.getPrice(volume) <= minShip || minShip == 0))

minShip = c.getPrice(volume);

}

}

System.out.println(minPlane != 0 ? "\nAirplane minimal price = " + minPlane + "\n" : "");

System.out.println(minShip != 0 ? "\nShip minimal price = " + minShip + "\n" : "");

for (Carrier c : carriers) {

if (c.getMaxVolume() >= volume) {

System.out.println(c);

System.out.println("\t Price : " + c.getPrice(volume));

System.out.println("\t Description : " + c.getDescription());

System.out.println("\t Max volume : " + c.getMaxVolume() + ‘\n’);

}

}

}

}

public class Carrier {

private int id;

private String title;

private Double MaxVolume;

private String description;

private Date transitDate;

public Carrier(int id, String title, Double MaxVolume, String description, Date transitDate) {

this.id = id;

this.title = title;

this.MaxVolume = MaxVolume;

this.description = description;

this.transitDate = transitDate;

}

public double getPrice(double volume) {

return volume;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public Double getMaxVolume() {

return MaxVolume;

}

public void setMaxVolume(Double maxVolume) {

this.MaxVolume = maxVolume;

}

public String getDescription() {

return description;

}

public void setDescription(String description) {

this.description = description;

}

public Date getTransitDate() {

return transitDate;

}

public void setTransitDate(Date transitDate) {

this.transitDate = transitDate;

}

@Override

public String toString() {

return "{" + "title:" + title + ", Max volume:" + MaxVolume + ", description:" + description + ", date:" + transitDate + "}";

}

}

public class Ship extends Carrier {

private Point2D googleMapsXY;

private String description;

private final static Double shipMultiply = 10.3;

public Ship(Point2D gmXY, int id, String title, Double volume, String description, Date transitDate) {

super(id, title, volume, description, transitDate);

googleMapsXY = gmXY;

this.description = description;

}

@Override

public String getDescription() {

return description + " { Destination is :" + googleMapsXY + " }";

}

@Override

public double getPrice(double volume) {

return volume \* shipMultiply;

}

public Point2D getGoogleMapsXY() {

return googleMapsXY;

}

public void setGoogleMapsXY(Point2D googleMapsXY) {

this.googleMapsXY = googleMapsXY;

}

}

public class Airplane extends Carrier {

private String trackID;

private String description;

private final static Double airMultiply = 15.4;

public Airplane(String trackID, int id, String title, Double volume, String description, Date transitDate) {

super(id, title, volume, description, transitDate);

this.trackID = trackID;

this.description = description;

}

public String getTrackID() {

return trackID;

}

public void setTrackID(String trackID) {

this.trackID = trackID;

}

@Override

public double getPrice(double volume) {

return volume \* airMultiply;

}

@Override

public String getDescription() {

return description + " { Track on -> " + trackID + " } ";

}

}

Результат выполнения (рис. 1):

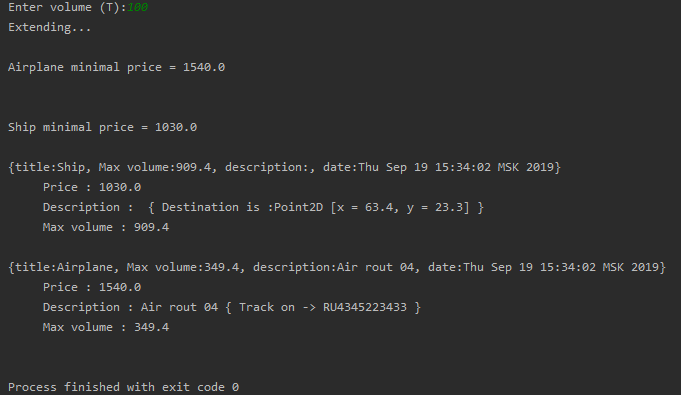


Рис. 1

**Абстракция**

Текст программы:

public class Test02 {

public static void main(String[] args) {

CarrierInfo[] carriers = new CarrierInfo[7];

carriers[0] = new Ship(new Point2D(33.4, 55.3), 3, "Airplane", 99.4, "Ship cargo", new Date());

carriers[1] = new Airplane("RU4277773433", 3, "Airplane", 9.4, "Air rout 01", new Date());

carriers[2] = new Airplane("ZA4343223433", 3, "Airplane", 11.4, "Air rout 02", new Date());

carriers[3] = new Airplane("LL4343223433", 3, "Airplane", 3.4, "Air rout 03", new Date());

carriers[4] = new Ship(new Point2D(63.4, 23.3), 3, "Ship", 909.4, "", new Date());

carriers[5] = new Airplane("RU4345223433", 3, "Airplane", 349.4, "Air rout 04", new Date());

carriers[6] = new Ship(new Point2D(3.4, 52.3), 3, "Ship", 22.4, "", new Date());

Scanner scanner = new Scanner(System.in);

System.out.print("Enter volume (T):");

double volume = scanner.nextDouble();

double minPlane = 0;

double minShip = 0;

System.out.println("Abstracting...");

for (CarrierInfo c : carriers) {

if (c.getMaxVolume() >= volume) {

if (c instanceof Airplane && (c.getPrice(volume) <= minPlane || minPlane == 0))

minPlane = c.getPrice(volume);

else if (c instanceof Ship && (c.getPrice(volume) <= minShip || minShip == 0))

minShip = c.getPrice(volume);

}

}

System.out.println(minPlane != 0 ? "\nAirplane minimal price = " + minPlane + "\n" : "");

System.out.println(minShip != 0 ? "\nShip minimal price = " + minShip + "\n" : "");

for (CarrierInfo c : carriers) {

if (c.getMaxVolume() >= volume) {

System.out.println(c);

System.out.println("\t Price : " + c.getPrice(volume));

System.out.println("\t Description : " + c.getDescription());

System.out.println("\t Max volume : " + c.getMaxVolume()+'\n');

}

}

}

}

public abstract class CarrierInfo {

protected int id;

public String getDescription() {

return "{" + id + "}";

}

abstract Double getPrice(double volume);

abstract Double getMaxVolume();

}

public class Carrier extends CarrierInfo {

private String title;

protected Double maxVolume;

protected String description;

private Date transitDate;

public Carrier(int id, String title, Double maxVolume, String description, Date transitDate) {

super.id = id;

this.title = title;

this.maxVolume = maxVolume;

this.description = description;

this.transitDate = transitDate;

}

public Double getPrice(double volume) {

return volume;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public Double getMaxVolume() {

return maxVolume;

}

public void setMaxVolume(Double maxVolume) {

this.maxVolume = maxVolume;

}

public String getDescription() {

return description;

}

public void setDescription(String description) {

this.description = description;

}

public Date getTransitDate() {

return transitDate;

}

public void setTransitDate(Date transitDate) {

this.transitDate = transitDate;

}

@Override

public String toString() {

return "{" + "title:" + title + ", max volume:" + maxVolume + ", description:" + description + ", date:" + transitDate + "}";

}

}

public class Ship extends Carrier {

private Point2D googleMapsXY;

private final static Double shipMultiply = 10.3;

public Ship(Point2D gmXY, int id, String title, Double maxVolume, String description, Date transitDate) {

super(id, title, maxVolume, description, transitDate);

googleMapsXY = gmXY;

}

@Override

public String getDescription() {

return description + " { Destination is :" + googleMapsXY + " }";

}

@Override

public Double getPrice(double volume) {

return volume \* shipMultiply;

}

public Point2D getGoogleMapsXY() {

return googleMapsXY;

}

public void setGoogleMapsXY(Point2D googleMapsXY) {

this.googleMapsXY = googleMapsXY;

}

}

public class Airplane extends Carrier {

private String trackID;

private final static Double airMultiply = 15.4;

public Airplane(String trackID, int id, String title, Double maxVolume, String description, Date transitDate) {

super(id, title, maxVolume, description, transitDate);

this.trackID = trackID;

}

public String getTrackID() {

return trackID;

}

public void setTrackID(String trackID) {

this.trackID = trackID;

}

@Override

public Double getPrice(double volume) {

return volume \* airMultiply;

}

@Override

public String getDescription() {

return description + " { Track on -> " + trackID + " } ";

}

}

Результат выполнения (рис. 2):

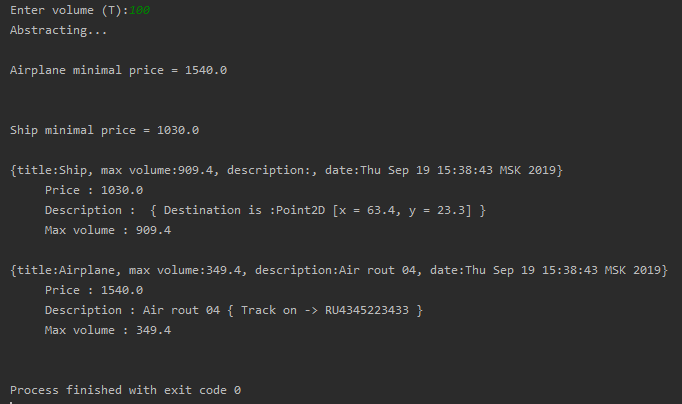


Рис. 2

**Интерфейсы**

Текст программы:

public class Test03 {

public static void main(String[] args) {

Comparable[] carriers = new Comparable[7];

carriers[0] = new Ship(new Point2D(33.4, 55.3), "Ship", 99.4, "Ship cargo", new Date());

carriers[1] = new Airplane("RU4277773433", "Airplane", 9.4, "Air rout 01", new Date());

carriers[2] = new Airplane("ZA4343223433", "Airplane", 11.4, "Air rout 02", new Date());

carriers[3] = new Airplane("LL4343223433", "Airplane", 3.4, "Air rout 03", new Date());

carriers[4] = new Ship(new Point2D(63.4, 23.3), "Ship", 909.4, "", new Date());

carriers[5] = new Airplane("RU4345223433", "Airplane", 349.4, "Air rout 04", new Date());

carriers[6] = new Ship(new Point2D(3.4, 52.3), "Ship", 22.4, "", new Date());

Scanner scanner = new Scanner(System.in);

System.out.print("Enter volume (T):");

double volume = scanner.nextDouble();

double minPlane = 0;

double minShip = 0;

System.out.println("Implementing...");

for (Comparable c : carriers) {

if (c.getMaxVolume() >= volume) {

if (c instanceof Airplane && (c.getPrice(volume) <= minPlane || minPlane == 0))

minPlane = c.getPrice(volume);

else if (c instanceof Ship && (c.getPrice(volume) <= minShip || minShip == 0))

minShip = c.getPrice(volume);

}

}

System.out.println(minPlane != 0 ? "\nAirplane minimal price = " + minPlane + "\n" : "");

System.out.println(minShip != 0 ? "\nShip minimal price = " + minShip + "\n" : "");

for (Comparable c : carriers) {

if (c.getMaxVolume() >= volume) {

System.out.println(c);

System.out.println("\t Price : " + c.getPrice(volume));

System.out.println("\t Description : " + c.getDescription());

System.out.println("\t Max volume : " + c.getMaxVolume()+'\n');

}

}

}

}

public interface Comparable {

String getDescription();

Double getMaxVolume();

Double getPrice(double volume);

}

public class Carrier implements Comparable {

private String title;

protected Double maxVolume;

protected String description;

private Date transitDate;

private boolean isRemoved = false;

public Carrier(String title, Double maxVolume, String description, Date transitDate) {

this.title = title;

this.maxVolume = maxVolume;

this.description = description;

this.transitDate = transitDate;

}

public Double getPrice(double volume) {

return volume;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public String getDescription() {

return description;

}

@Override

public Double getMaxVolume() {

return maxVolume;

}

public void setDescription(String description) {

this.description = description;

}

public Date getTransitDate() {

return transitDate;

}

public void setTransitDate(Date transitDate) {

this.transitDate = transitDate;

}

@Override

public String toString() {

return "{" + "title:" + title + ", Max volume:" + maxVolume + ", description:" + description + ", date:" + transitDate + "}";

}

}

public class Airplane extends Carrier {

private String trackID;

private final static Double airMultiply = 15.4;

public Airplane(String trackID, String title, Double volume, String description, Date transitDate) {

super(title, volume, description, transitDate);

this.trackID = trackID;

}

public String getTrackID() {

return trackID;

}

public void setTrackID(String trackID) {

this.trackID = trackID;

}

@Override

public Double getPrice(double volume) {

return volume \* airMultiply;

}

@Override

public String getDescription() {

return description + " { Track on -> " + trackID + " } ";

}

}

public class Ship extends Carrier {

private Point2D googleMapsXY;

private final static Double shipMultiply = 10.3;

public Ship(Point2D gmXY, String title, Double volume, String description, Date transitDate) {

super(title, volume, description, transitDate);

googleMapsXY = gmXY;

}

@Override

public String getDescription() {

return description + " { Destination is :" + googleMapsXY + " }";

}

@Override

public Double getPrice(double volume) {

return volume \* shipMultiply;

}

public Point2D getGoogleMapsXY() {

return googleMapsXY;

}

public void setGoogleMapsXY(Point2D googleMapsXY) {

this.googleMapsXY = googleMapsXY;

}

}

Результат выполнения (рис. 3):

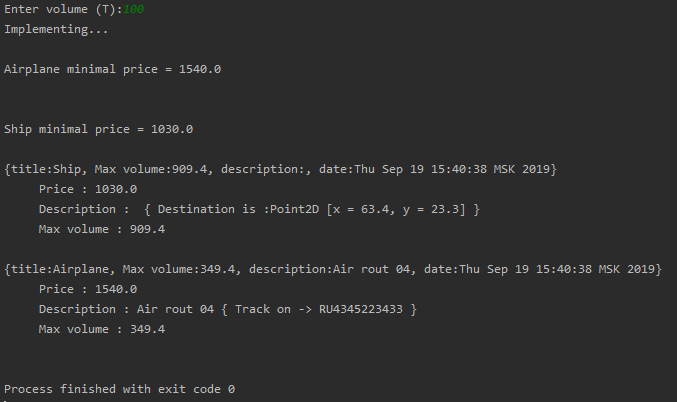


Рис. 3