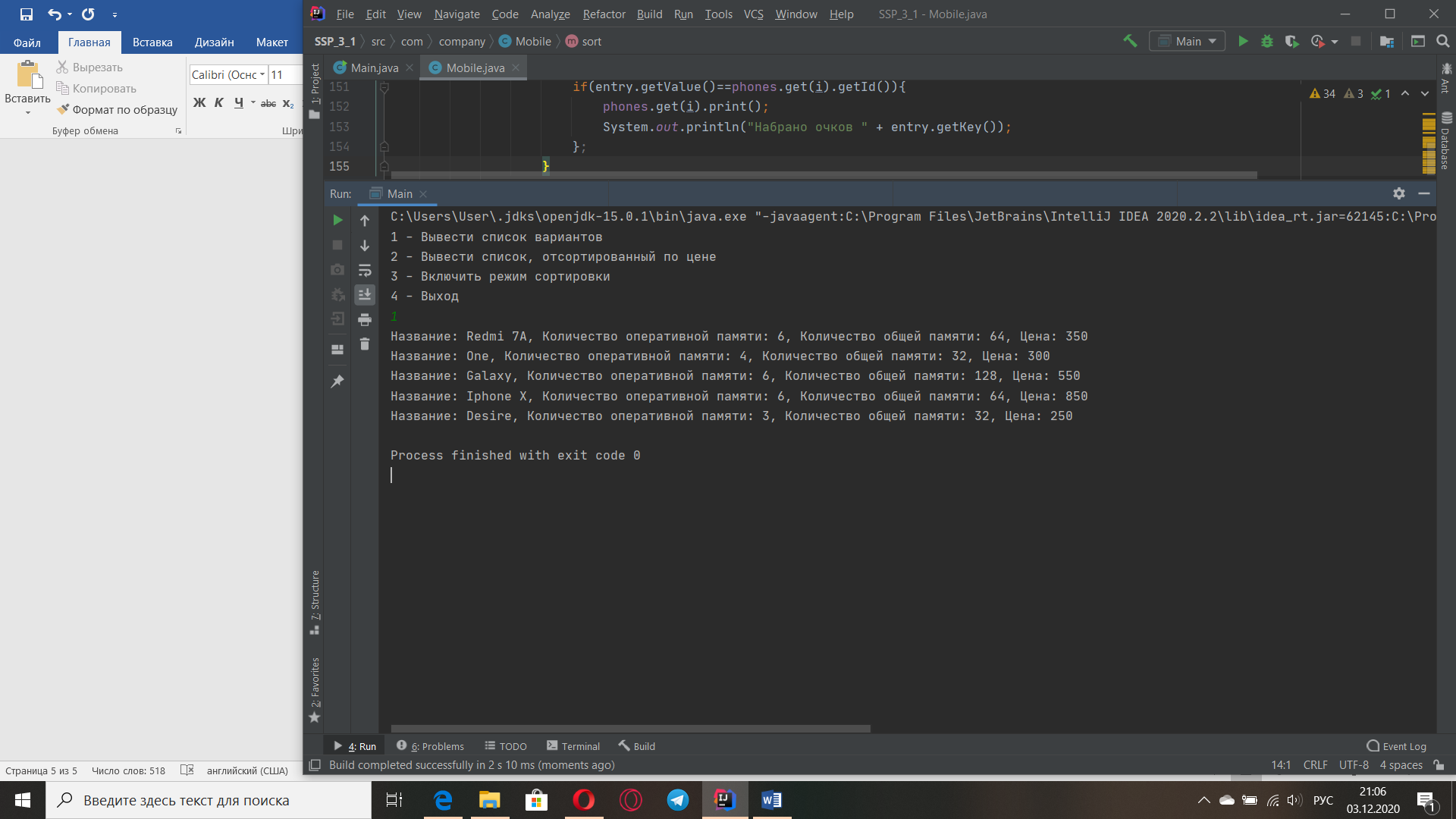
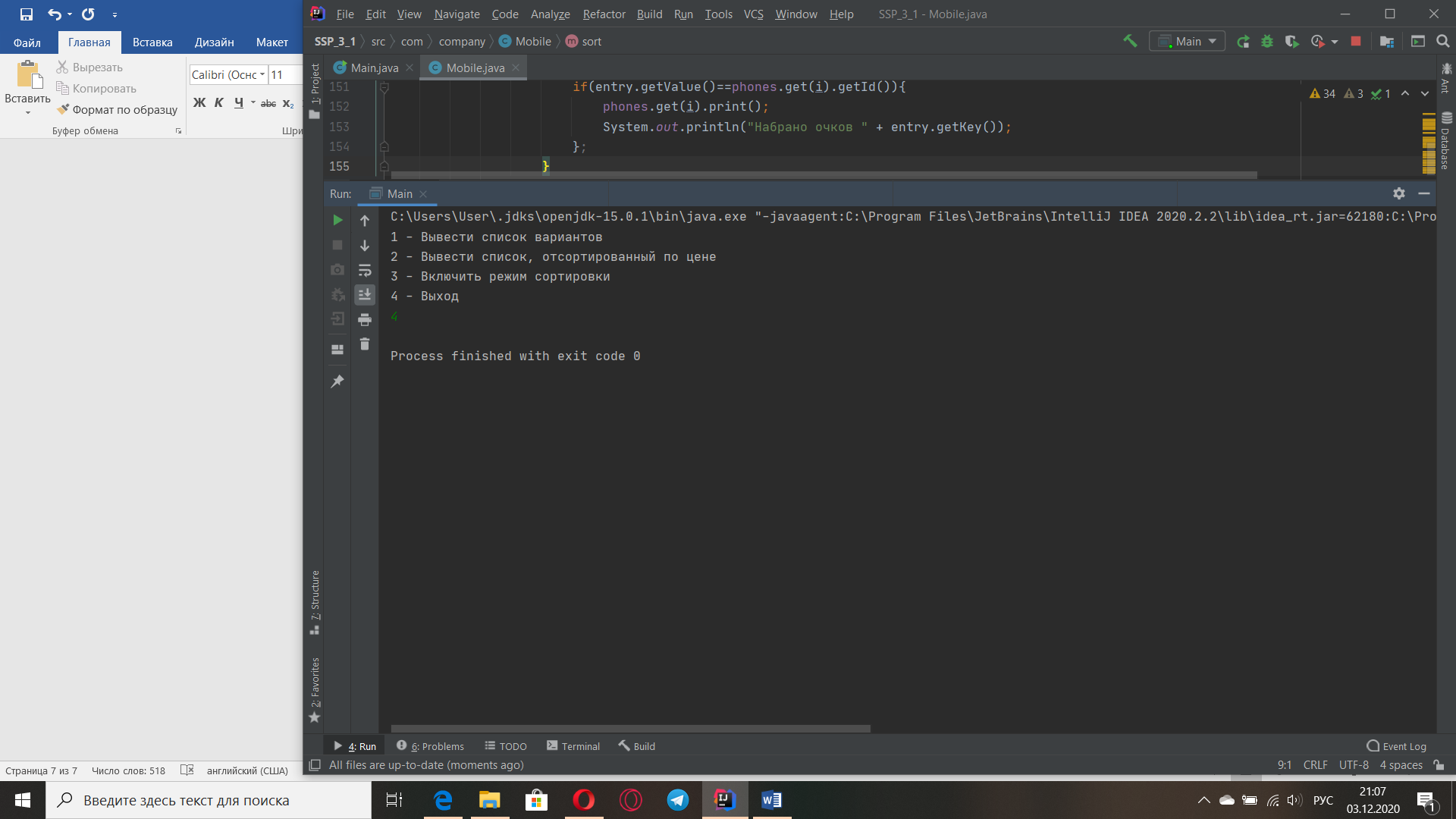
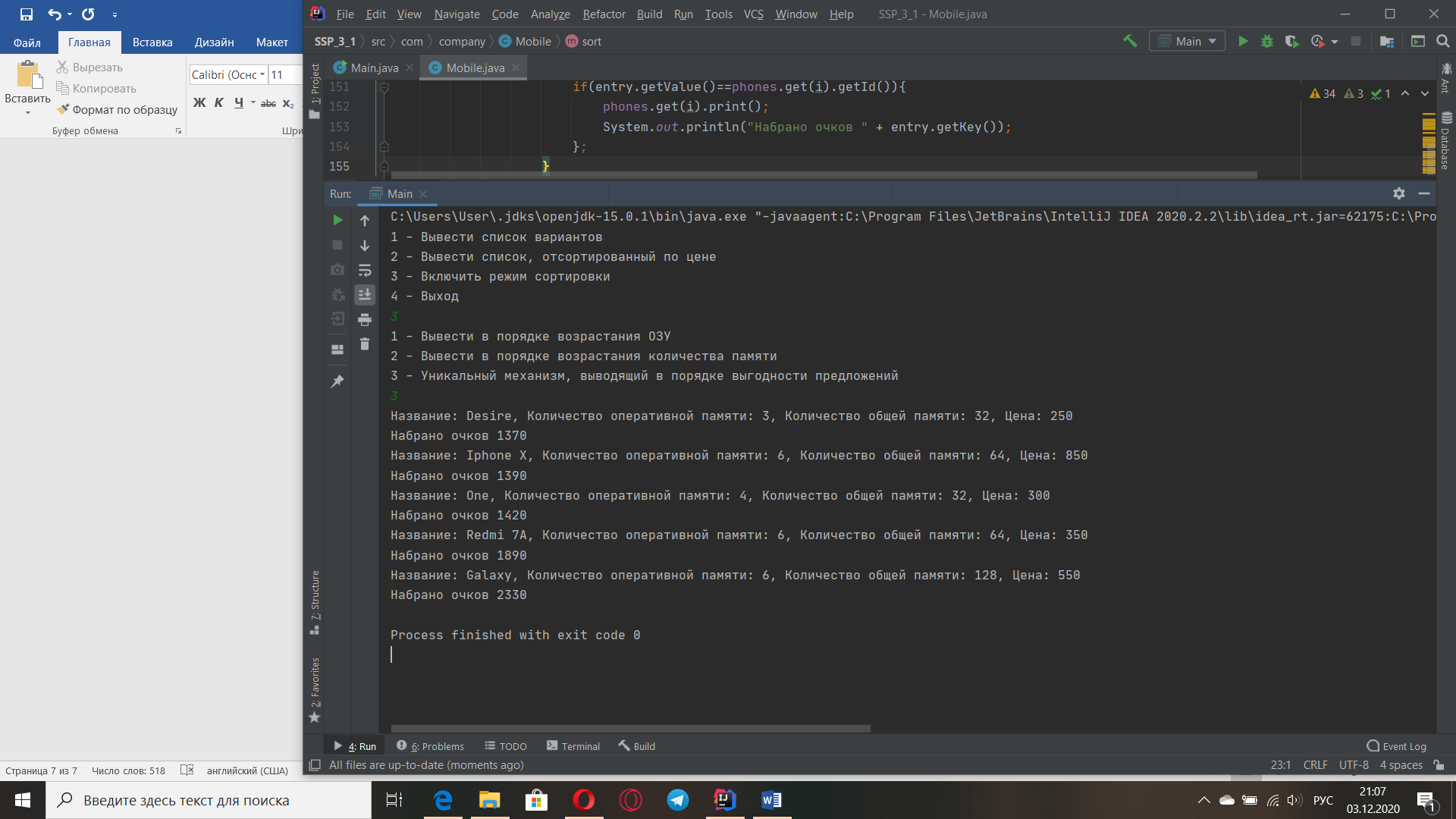
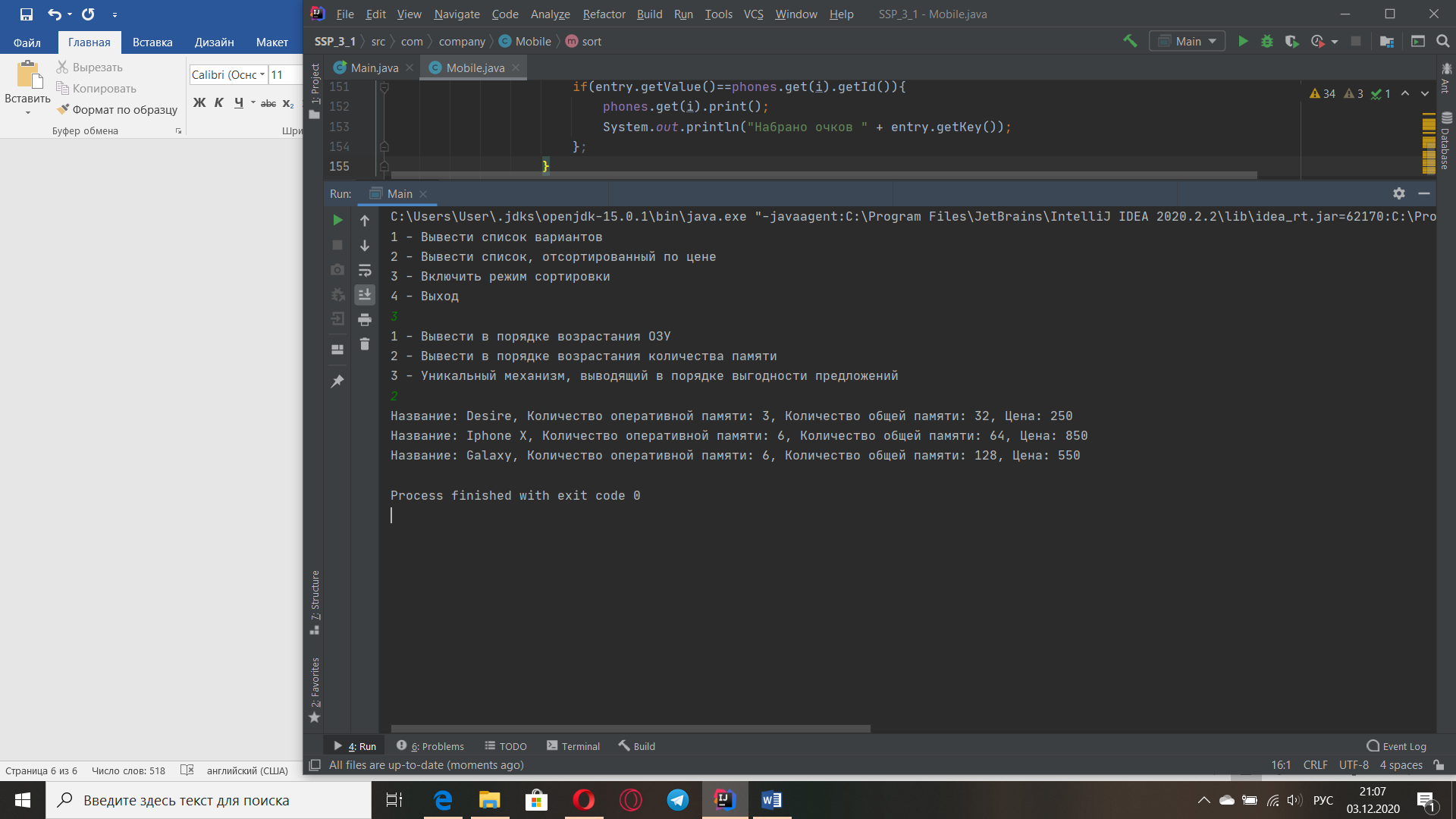
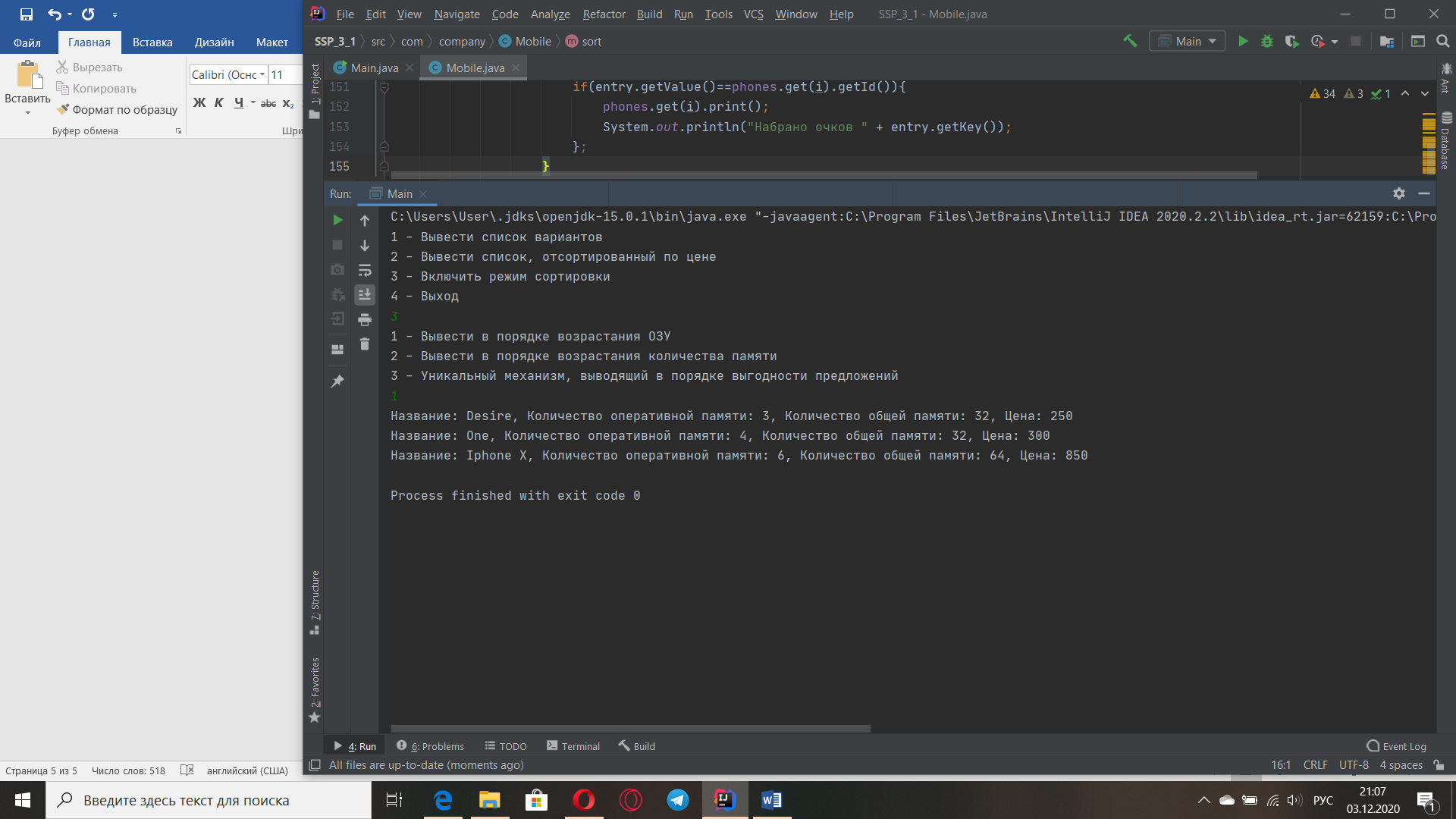
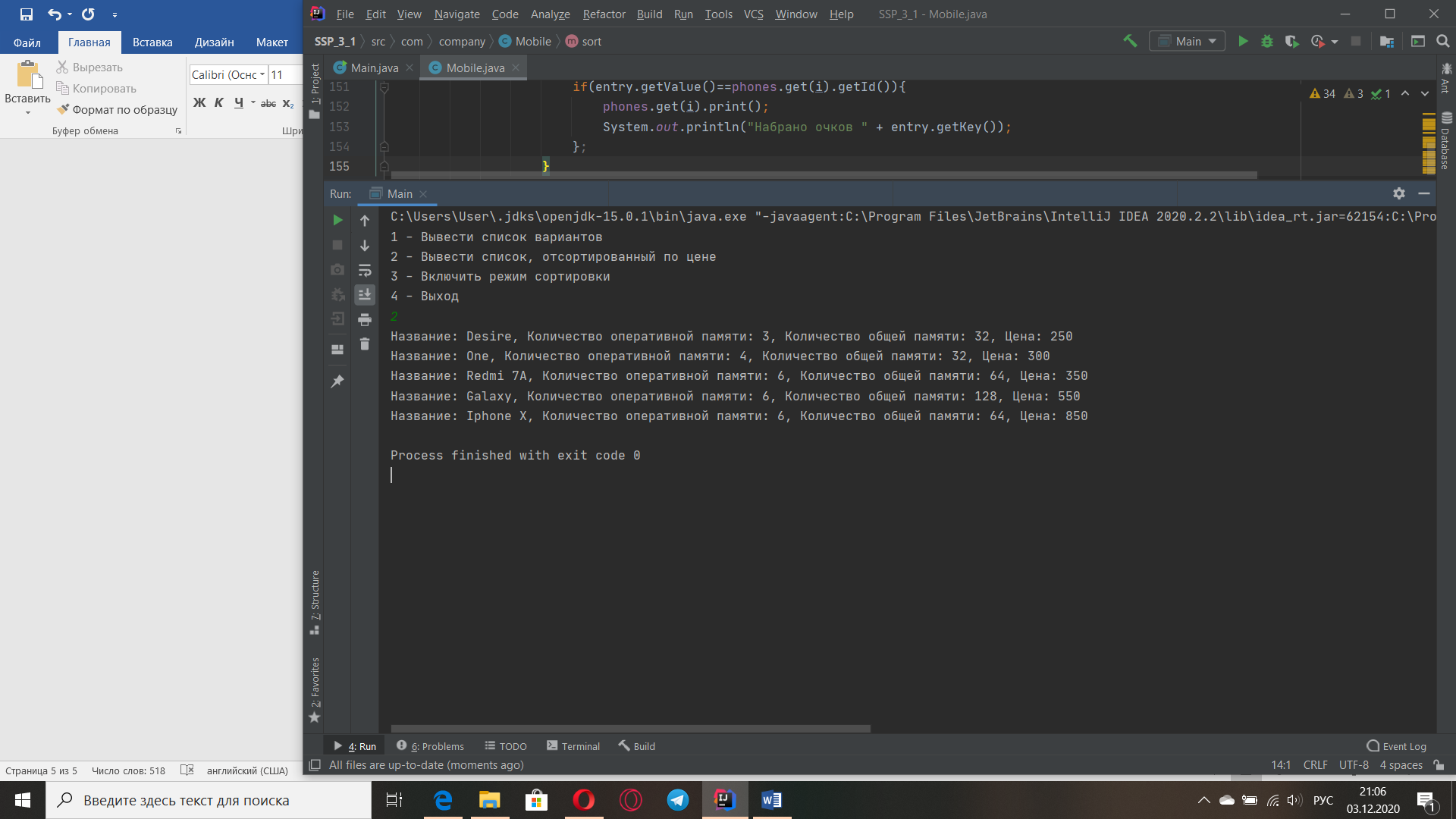
**Задание 1**

9) Создать класс Mobile с внутренним классом, с помощью объектов которого можно хранить информацию о моделях телефонов и их свойствах.

package com.company;  
  
import java.util.\*;  
  
class Mobile{  
  
 static class NewPhone {  
 private int id;  
 private String manufacturer;  
 private String name;  
 private int OZU;  
 private int memory;  
 private int price;  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public NewPhone(int id, String manufacturer, String name, int OZU, int memory, int price) {  
 this.id = id;  
 this.manufacturer = manufacturer;  
 this.name = name;  
 this.OZU = OZU;  
 this.memory = memory;  
 this.price = price;  
 }  
  
 public String getManufacturer() {  
 return manufacturer;  
 }  
  
 public void setManufacturer(String manufacturer) {  
 this.manufacturer = manufacturer;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public int getOZU() {  
 return OZU;  
 }  
  
 public void setOZU(int OZU) {  
 this.OZU = OZU;  
 }  
  
 public int getMemory() {  
 return memory;  
 }  
  
 public void setMemory(int memory) {  
 this.memory = memory;  
 }  
  
 public int getPrice() {  
 return price;  
 }  
  
 public void setPrice(int price) {  
 this.price = price;  
 }  
  
 public void print(){  
 System.*out*.println("Название: " + name + ", Количество оперативной памяти: " + OZU + ", Количество общей памяти: " + memory + ", Цена: " + price);  
 }  
 }  
 private ArrayList<NewPhone> phones = new ArrayList();  
 public Mobile(){  
 phones.add(new NewPhone(1,"Xiaomi","Redmi 7A", 6, 64, 350));  
 phones.add(new NewPhone(2,"Motorolla","One", 4, 32, 300));  
 phones.add(new NewPhone(3,"Samsung","Galaxy", 6, 128, 550));  
 phones.add(new NewPhone(4,"Apple","Iphone X", 6, 64, 850));  
 phones.add(new NewPhone(5,"HTC","Desire", 3, 32, 250));  
 }  
  
 public void printMobile(){  
 for (int i = 0 ; i < phones.size(); i++){  
 phones.get(i).print();  
 }  
 }  
 private Map<Integer,Integer> prices = new HashMap<Integer, Integer>();  
 private HashMap<Integer, Integer> ozu = new HashMap<>();  
 private Map<Integer, Integer> memory = new HashMap<>();  
 private Map<Integer,Integer> count = new HashMap<>();  
  
 public void sortFromPrice(){  
 for (int i = 0 ; i < phones.size(); i++){  
 prices.put(phones.get(i).getPrice(),phones.get(i).getId());  
 }  
 Map<Integer, Integer> sortPrice = new TreeMap<Integer,Integer>(prices);  
 for (Map.Entry<Integer, Integer> entry : sortPrice.entrySet()){  
 for (int j = 0 ; j < phones.size(); j++){  
 if(entry.getValue()==phones.get(j).getId()){  
 phones.get(j).print();  
 };  
 }  
 }  
 }  
 public void sort(){  
 System.*out*.println("1 - Вывести в порядке возрастания ОЗУ");  
 System.*out*.println("2 - Вывести в порядке возрастания количества памяти");  
 System.*out*.println("3 - Уникальный механизм, выводящий в порядке выгодности предложений");  
 Scanner input = new Scanner(System.*in*);  
 int menu = input.nextInt();  
 switch (menu){  
 case 1:{  
 for (int i = 0 ; i < phones.size(); i++){  
 ozu.put(phones.get(i).getOZU(), phones.get(i).getId());  
 }  
 Map<Integer, Integer> sortOZU = new TreeMap<Integer,Integer>(ozu);  
 for (Map.Entry<Integer, Integer> entry : sortOZU.entrySet()){  
 for (int i = 0 ; i < phones.size(); i++){  
 if(entry.getValue()==phones.get(i).getId()){  
 phones.get(i).print();  
 };  
 }  
 }  
 break;  
 }  
 case 2:{  
 for (int i = 0 ; i < phones.size(); i++){  
 memory.put(phones.get(i).getMemory(),phones.get(i).getId());  
 }  
 Map<Integer, Integer> sortMemory = new TreeMap<Integer,Integer>(memory);  
 for (Map.Entry<Integer, Integer> entry : sortMemory.entrySet()){  
 for (int i = 0 ; i < phones.size(); i++){  
 if(entry.getValue()==phones.get(i).getId()){  
 phones.get(i).print();  
 };  
 }  
 }  
 break;  
 }  
 case 3:{  
 for (int i = 0 ; i < phones.size(); i++){  
 int counts = 1000 - phones.get(i).getPrice() + phones.get(i).getOZU()\*100 + phones.get(i).getMemory()\*10;  
 count.put(counts,phones.get(i).getId());  
 }  
 Map<Integer, Integer> sortCount = new TreeMap<Integer,Integer>(count);  
 for (Map.Entry<Integer, Integer> entry : sortCount.entrySet()){  
 for (int i = 0 ; i < phones.size(); i++){  
 if(entry.getValue()==phones.get(i).getId()){  
 phones.get(i).print();  
 System.*out*.println("Набрано очков " + entry.getKey());  
 };  
 }  
 }  
 break;  
 }  
 }  
 }  
}



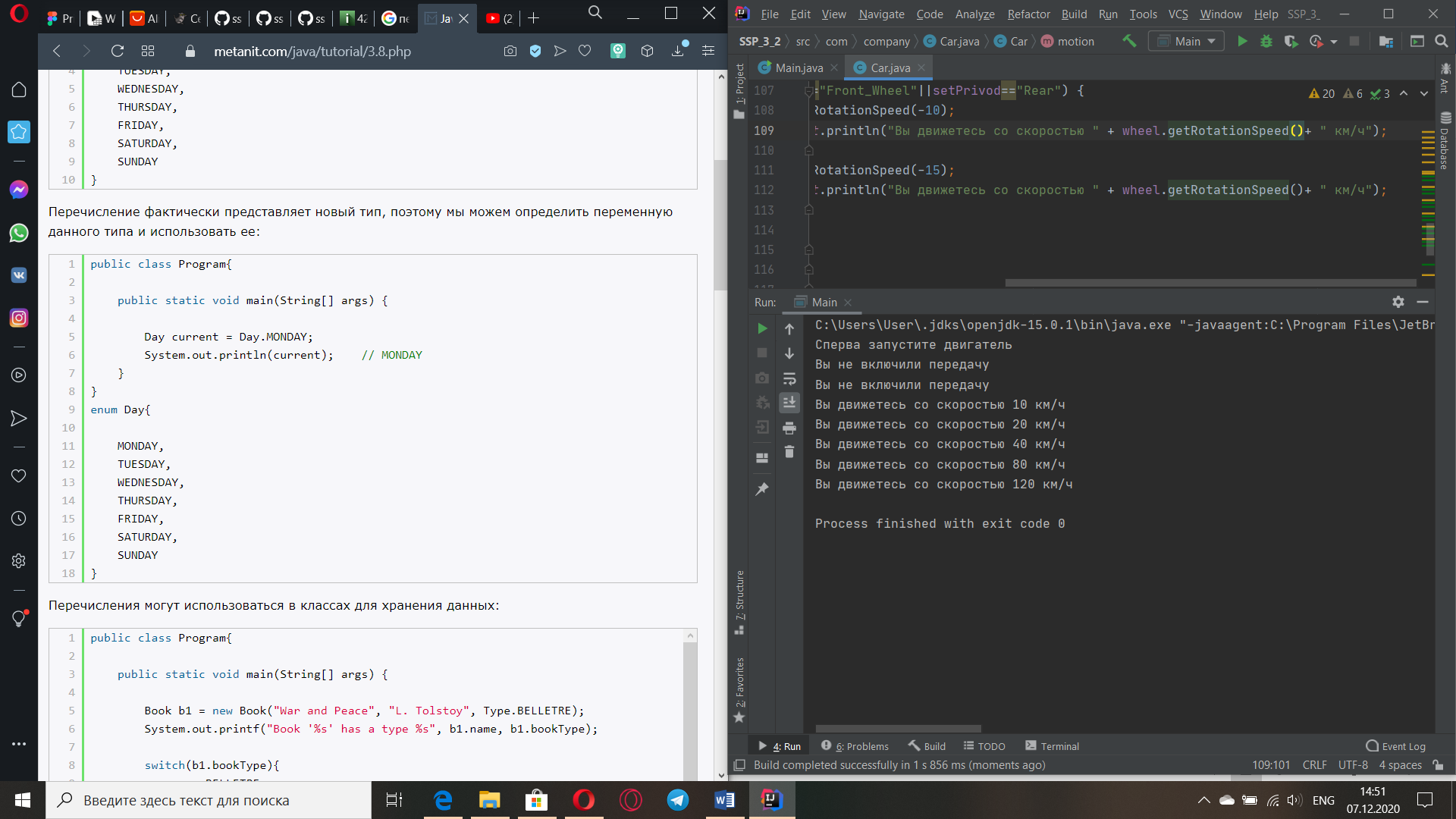


**Задание 2**

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.

9) Создать класс Автомобиль, используя класс Колесо.

package com.company;  
  
  
enum privod {  
 *Front\_Wheel*,  
 *Rear*,  
 *Four\_Wheel*}  
  
public class Car {  
 Wheel wheel = new Wheel() ;  
 String setPrivod;  
 boolean isEngineStarted = false;  
 int turnGear;  
 int [] gear = {0,1,2,3,4,5,-1};  
  
 public Car(String setPrivod) {  
 this.setPrivod = setPrivod;  
 }  
  
 public boolean isEngineStarted() {  
 return isEngineStarted;  
 }  
  
 public void setEngineStarted(boolean engineStarted) {  
 isEngineStarted = engineStarted;  
 }  
  
 public int[] getGear() {  
 return gear;  
 }  
  
 public void setGear(int[] gear) {  
 this.gear = gear;  
 }  
  
 public boolean startEngine(){  
 setEngineStarted(true);  
 return isEngineStarted;  
 }  
  
 public int turnOnGear(int gear){  
 turnGear=gear;  
 return turnGear;  
 }  
  
 public void motion(){  
 if(isEngineStarted==false){  
 System.*out*.println("Сперва запустите двигатель");  
 }else {  
 if (turnGear == gear[0]) {  
 System.*out*.println("Вы не включили передачу");  
 }  
 else {  
 switch (turnGear){  
 case 1:{  
 if(setPrivod=="Front\_Wheel"||setPrivod=="Rear") {  
 wheel.setRotationSpeed(10);  
 System.*out*.println("Вы движетесь со скоростью " + wheel.getRotationSpeed()+" км/ч");  
 }else{  
 wheel.setRotationSpeed(15);  
 System.*out*.println("Вы движетесь со скоростью " + wheel.getRotationSpeed() + " км/ч");  
 }  
 break;  
 }  
 case 2:{  
 if(setPrivod=="Front\_Wheel"||setPrivod=="Rear") {  
 wheel.setRotationSpeed(20);  
 System.*out*.println("Вы движетесь со скоростью " + wheel.getRotationSpeed()+ " км/ч");  
 }else{  
 wheel.setRotationSpeed(30);  
 System.*out*.println("Вы движетесь со скоростью " + wheel.getRotationSpeed()+ " км/ч");  
 }  
 break;  
 }  
 case 3:{  
 if(setPrivod=="Front\_Wheel"||setPrivod=="Rear") {  
 wheel.setRotationSpeed(40);  
 System.*out*.println("Вы движетесь со скоростью " + wheel.getRotationSpeed()+ " км/ч");  
 }else{  
 wheel.setRotationSpeed(60);  
 System.*out*.println("Вы движетесь со скоростью " + wheel.getRotationSpeed()+ " км/ч");  
 }  
 break;  
 }  
 case 4:{  
 if(setPrivod=="Front\_Wheel"||setPrivod=="Rear") {  
 wheel.setRotationSpeed(80);  
 System.*out*.println("Вы движетесь со скоростью " + wheel.getRotationSpeed()+ " км/ч");  
 }else{  
 wheel.setRotationSpeed(120);  
 System.*out*.println("Вы движетесь со скоростью " + wheel.getRotationSpeed()+ " км/ч");  
 }  
 break;  
 }  
 case 5:{  
 if(setPrivod=="Front\_Wheel"||setPrivod=="Rear") {  
 wheel.setRotationSpeed(120);  
 System.*out*.println("Вы движетесь со скоростью " + wheel.getRotationSpeed()+ " км/ч");  
 }else{  
 wheel.setRotationSpeed(160);  
 System.*out*.println("Вы движетесь со скоростью " + wheel.getRotationSpeed()+ " км/ч");  
 }  
 break;  
 }  
 case -1:{  
 if(setPrivod=="Front\_Wheel"||setPrivod=="Rear") {  
 wheel.setRotationSpeed(-10);  
 System.*out*.println("Вы движетесь со скоростью " + wheel.getRotationSpeed()+ " км/ч");  
 }else{  
 wheel.setRotationSpeed(-15);  
 System.*out*.println("Вы движетесь со скоростью " + wheel.getRotationSpeed()+ " км/ч");  
 }  
 break;  
 }  
 }  
 }  
 }  
  
 }  
}  
  
class Wheel {  
 int rotationSpeed = 0;  
  
 public int getRotationSpeed() {  
 return rotationSpeed;  
 }  
  
 public void setRotationSpeed(int rotationSpeed) {  
 this.rotationSpeed = rotationSpeed;  
 }  
  
 public int returnRotationSpeed(){  
 return rotationSpeed;  
 }  
}



**Задание 3**

9) Система Железнодорожная касса. Пассажир делает Заявку на станцию назначения, время и дату поездки. Система регистрирует Заявку и осуществляет поиск подходящего Поезда. Пассажир делает выбор Поезда и получает Счет на оплату. Администратор вводит номера Поездов, промежуточные и конечные станции, цены.

**Main.java**

package com.company;  
  
import java.util.ArrayList;  
import java.util.Scanner;  
  
public class Main {  
  
 public static void main(String[] args) {  
 String date;  
 String point;  
 Passenger passenger = new Passenger();  
 passenger.setDestination("Пинск");  
 passenger.setData("07.12.2020");  
 passenger.setPoint("Брест");  
 passenger.setTime(12.15);  
 passenger.create\_request();  
  
 passenger.setDestination("Заболоть");  
 passenger.setData("14.12.2020");  
 passenger.setPoint("Брест");  
 passenger.setTime(18.25);  
 passenger.create\_request();  
  
 passenger.setDestination("Ольховск");  
 passenger.setData("09.12.2020");  
 passenger.setPoint("Брест");  
 passenger.setTime(01.00);  
 passenger.create\_request();  
  
 passenger.setDestination("Подгорье");  
 passenger.setData("07.12.2020");  
 passenger.setPoint("Брест");  
 passenger.setTime(23.00);  
 passenger.create\_request();  
  
 passenger.setDestination("Воронеж");  
 passenger.setData("17.12.2020");  
 passenger.setPoint("Брест");  
 passenger.setTime(21.48);  
 passenger.create\_request();  
  
  
 passenger.printRequest();  
  
 System.*out*.println("Введите номер вашей заявки для отображения списка поездов");  
 Scanner requestNumber = new Scanner(System.*in*);  
 int number = requestNumber.nextInt();  
  
 date = passenger.requestsList.get(number).departure\_date;  
 point = passenger.requestsList.get(number).final\_destination;  
  
 Administrator administrator = new Administrator();  
 administrator.setData("07.12.2020");  
 administrator.setDestination("Пинск");  
 ArrayList<String> stations = new ArrayList<>();  
 stations.add("Барановичи");  
 stations.add("Подгорье");  
 stations.add("Витебск");  
 stations.add("Могилёв");  
 administrator.setStations(stations);  
 ArrayList<Double> stationsTimes = new ArrayList<>();  
 stationsTimes.add(12.00);  
 stationsTimes.add(14.00);  
 stationsTimes.add(18.25);  
 stationsTimes.add(23.12);  
 administrator.setStationsTime(stationsTimes);  
  
 administrator.create\_railway();  
  
 administrator.printAvailableRailway(date, point);  
 System.*out*.println("Введите номер выбраного поезда");  
 Scanner railwayNumber = new Scanner(System.*in*);  
 int r\_number =railwayNumber.nextInt();  
 System.*out*.println("Ваш счёт на оплату");  
 System.*out*.println("Оплатить поезд № " + number);  
 System.*out*.println("Стоимость оплаты: " + 50\*stations.size());  
 }  
}

**Railway.java**

package com.company;  
  
import java.util.ArrayList;  
  
public class Railway {  
 String fin\_destination;  
 String departure\_date;  
 ArrayList<String> stations;  
 ArrayList<Double> stationsTime;  
  
 public Railway(String fin\_destination, String departure\_date, ArrayList<String> stations, ArrayList<Double> stationsTime) {  
 this.fin\_destination = fin\_destination;  
 this.departure\_date = departure\_date;  
 this.stations = stations;  
 this.stationsTime = stationsTime;  
 }  
}

**Request.java**

package com.company;  
  
public class Request {  
 String departure\_date;  
 String departure\_point;  
 double departure\_time;  
 String final\_destination;  
  
 public Request(String departure\_date, String departure\_point, double departure\_time, String final\_destination) {  
 this.departure\_date = departure\_date;  
 this.departure\_point = departure\_point;  
 this.departure\_time = departure\_time;  
 this.final\_destination = final\_destination;  
 }  
}

**Passenger.java**

package com.company;  
  
import java.util.ArrayList;  
  
public class Passenger {  
 ArrayList<Request> requestsList = new ArrayList<>();  
 String data;  
 String point;  
 double time;  
 String destination;  
  
 public String getData() {  
 return data;  
 }  
  
 public void setData(String data) {  
 this.data = data;  
 }  
  
 public String getPoint() {  
 return point;  
 }  
  
 public void setPoint(String point) {  
 this.point = point;  
 }  
  
 public double getTime() {  
 return time;  
 }  
  
 public void setTime(double time) {  
 this.time = time;  
 }  
  
 public String getDestination() {  
 return destination;  
 }  
  
 public void setDestination(String destination) {  
 this.destination = destination;  
 }  
  
 public void create\_request(){  
 requestsList.add(new Request(data,point,time,destination));  
 }  
   
 void printRequest() {  
 int i = 0;  
 for (Request requeste: requestsList) {  
 System.*out*.println("Заявка номер: " + i + ", Дата: " + requeste.departure\_date + ", Время: " + requeste.departure\_time + ", Начальный пункт: " + requeste.departure\_point + ", Пункт назначения: " + requeste.final\_destination);  
 i++;  
 }  
 }  
}

**Administrator.java**

package com.company;  
  
import java.util.ArrayList;  
  
public class Administrator {  
 ArrayList<Railway> railwaysList = new ArrayList<>();  
 String data;  
 String destination;  
 ArrayList<String> stations;  
 ArrayList<Double> stationsTime;  
  
 public String getData() {  
 return data;  
 }  
  
 public void setData(String data) {  
 this.data = data;  
 }  
  
 public String getDestination() {  
 return destination;  
 }  
  
 public void setDestination(String destination) {  
 this.destination = destination;  
 }  
  
 public ArrayList<String> getStations() {  
 return stations;  
 }  
  
 public void setStations(ArrayList<String> stations) {  
 this.stations = stations;  
 }  
  
 public ArrayList<Double> getStationsTime() {  
 return stationsTime;  
 }  
  
 public void setStationsTime(ArrayList<Double> stationsTime) {  
 this.stationsTime = stationsTime;  
 }  
  
  
 public void create\_railway(){  
  
 railwaysList.add(new Railway(destination,data,stations,stationsTime));  
 }  
  
 void printAvailableRailway(String date, String point) {  
 int i = 0;  
 for (Railway railway: railwaysList) {  
 System.*out*.println(railway.departure\_date);  
 if(railway.departure\_date==date) {  
 if (railway.fin\_destination == point) {  
 System.*out*.println("Поезд номер: " + i + ", Дата: " + railway.departure\_date + ", Конечный пункт: " + railway.fin\_destination + ", Список станций: " + railway.stations + ", Время прибытия на станции: " + railway.stationsTime);  
 }  
 else{  
 for (String stations: railway.stations) {  
 if(stations==point){  
 System.*out*.println("Поезд номер: " + i + ", Дата: " + railway.departure\_date + ", Конечный пункт: " + railway.fin\_destination + ", Список станций: " + railway.stations + ", Время прибытия на станции: " + railway.stationsTime);  
 }  
 }  
 }  
 }  
 i++;  
 }  
 }  
}

