# 1.

package task1.quadraticEquations;  
/\*  
Разработайте класс для решения квадратных уравнений. Вычисление  
дискриминанта должен осуществлять вложенный класс. После компиляции  
объясните структуру class файлов. Проанализируйте использование вложенного  
класса.  
 \*/  
public class QuadraticEquations {  
 private double x1;  
 private double x2;  
 static class Discriminant{  
 double findDiscriminant(double a, double b, double c){  
 return Math.*pow*(b, 2) -4\*a\*c;  
 }  
 }  
 void solve(double a, double b, double c){  
 Discriminant discriminant = new QuadraticEquations.Discriminant();  
 double d = discriminant.findDiscriminant(a, b, c);  
 x1 = (-b + Math.*sqrt*(d))/(2\*a);  
 x2 = (-b - Math.*sqrt*(d))/(2\*a);  
 }  
  
 public static void main(String[] args) {  
 QuadraticEquations quadraticEquations = new QuadraticEquations();  
 quadraticEquations.solve(1, 4, 4);  
 System.*out*.println("a = 1 b = 4 c = 4");  
 System.*out*.println("x1 = " + quadraticEquations.x1 + "\nx2 = " + quadraticEquations.x2);  
 }  
}

a = 1 b = 4 c = 4

x1 = -2.0

x2 = -2.0

После компиляции получаем следующую структуру class файлов: 

Статический вложенный класс имеет метод findDiscriminant, с входными параметрами, равными коэфициентам квадратного уравнения, и возвращающий вычисленное значения дискриминанта.

# 2.

package task1.dice;  
  
import java.security.InvalidParameterException;  
import java.util.\*;  
  
/\*  
Реализуйте игру в кости. Играют N игроков (компьютер в списке последний).  
Подкидываются одновременно К кубиков. Выигрывает тот, у кого большая  
сумма очков. Кто выиграл, тот и кидает первым в следующем кону. Игра идет  
до 7 выигрышей. Начинаете игру Вы.  
 \*/  
  
public class Dice {  
 private int N; //N players and 1 computer  
 private int K; //K cubes  
 int[] points; //sum of points  
 int[] wins; //amount of wins  
 Dice(int N, int K){  
 if (N <= 0 || K <= 0){  
 throw new InvalidParameterException();  
 }  
 this.N = N;  
 this.K = K;  
 points = new int[N+1];  
 wins = new int[N+1];  
 }  
 public void play(){  
 Random random = new Random();  
 int winner = 0; //winner's index  
 int firstTurn = 0;  
 int max = 0;  
 while (true){  
 for(int i=firstTurn;i<N+1;i++){  
 System.*out*.print("player #" + i + " drop cubes: ");  
 for(int j=0;j<K;j++){  
 int point = random.nextInt(5) + 1;  
 points[i] += point;  
 System.*out*.print(point + " ");  
 }  
 System.*out*.println("\n" + points[i] + " points!");  
 if (points[i] > max){  
 max = points[i];  
 winner = i;  
 }  
 points[i] = 0;  
 }  
 for(int i=0;i<firstTurn;i++){  
 System.*out*.print("player #" + i + " drop cubes: ");  
 for(int j=0;j<K;j++){  
 int point = random.nextInt(5) + 1;  
 points[i] += point;  
 System.*out*.print(point + " ");  
 }  
 System.*out*.println("\n" + points[i] + " points!");  
 if (points[i] > max){  
 max = points[i];  
 winner = i;  
 }  
 points[i] = 0;  
 }  
 System.*out*.println("Winner of the round: player #" + winner + " with " + max + " points!\n");  
 max = 0;  
 wins[winner]++;  
 for (int player: wins) {  
 System.*out*.print(player + " ");  
 }  
 System.*out*.println("\n");  
 firstTurn = winner; //first try in next round for winner  
 if (wins[winner] == 7){  
 System.*out*.println("Winner of the GAME: player #" + winner + " won!");  
 break;  
 }  
 }  
 }  
 public static void main(String[] args) {  
 Dice dice = new Dice(3, 3);  
 dice.play();  
 }  
}

player #0 drop cubes: 3 2 4

9 points!

player #1 drop cubes: 1 4 5

10 points!

player #2 drop cubes: 5 4 3

12 points!

player #3 drop cubes: 5 5 5

15 points!

Winner of the round: player #3 with 15 points!

0 0 0 1

player #3 drop cubes: 5 4 1

10 points!

player #0 drop cubes: 1 2 4

7 points!

player #1 drop cubes: 3 1 4

8 points!

player #2 drop cubes: 4 4 3

11 points!

Winner of the round: player #2 with 11 points!

0 0 1 1

… и т.д.

player #3 drop cubes: 5 3 5

13 points!

player #0 drop cubes: 4 1 4

9 points!

player #1 drop cubes: 3 3 1

7 points!

player #2 drop cubes: 3 4 5

12 points!

Winner of the round: player #3 with 13 points!

5 5 5 7

Winner of the GAME: player #3 won!

Process finished with exit code 0

# 3.

package task1.address;  
  
import java.util.Calendar;  
import java.util.Objects;  
  
class Address {  
 //Один из возможных наборо полей  
 private String country;  
 private String city;  
 private String street;  
  
 public Address() {  
 }  
  
 Address(String country, String city, String street){  
 this.country = country;  
 this.city = city;  
 this.street = street;  
 }  
  
 public String getCountry() {  
 return country;  
 }  
  
 public void setCountry(String country) {  
 this.country = country;  
 }  
  
 public String getCity() {  
 return city;  
 }  
  
 public void setCity(String city) {  
 this.city = city;  
 }  
  
 public String getStreet() {  
 return street;  
 }  
  
 public void setStreet(String street) {  
 this.street = street;  
 }  
  
 @Override  
 public String toString() {  
 return "country='" + country + '\'' +  
 ", city='" + city + '\'' +  
 ", street='" + street + '\'';  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Address address = (Address) o;  
 return Objects.*equals*(country, address.country) && Objects.*equals*(city, address.city) && Objects.*equals*(street, address.street);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(country, city, street);  
 }  
}

package task1.address;  
  
import java.util.Calendar;  
import java.util.Objects;  
  
class Person {  
 private String name;  
 private String lastName;  
 private Calendar birthday;  
 private Address address;  
  
 public Person() {  
 }  
  
 Person(String name, String lastName, Calendar birthday, Address address){  
 this.name = name;  
 this.lastName = lastName;  
 this.birthday = birthday;  
 this.address = address;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getLastName() {  
 return lastName;  
 }  
  
 public void setLastName(String lastName) {  
 this.lastName = lastName;  
 }  
  
 public Calendar getBirthday() {  
 return birthday;  
 }  
  
 public void setBirthday(Calendar birthday) {  
 this.birthday = birthday;  
 }  
  
 public Address getAddress() {  
 return address;  
 }  
  
 public void setAddress(Address address) {  
 this.address = address;  
 }  
  
 @Override  
 public String toString() {  
 return "name='" + name + '\'' +  
 ", lastName='" + lastName + '\'' +  
 ", birthday=" + birthday.getTime() +  
 ", address : " + address.toString();  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Person person = (Person) o;  
 return Objects.*equals*(name, person.name) && Objects.*equals*(lastName, person.lastName) && Objects.*equals*(birthday, person.birthday) && Objects.*equals*(address, person.address);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(name, lastName, birthday, address);  
 }  
}

package task1.address;  
  
import java.util.\*;  
  
class Data {  
 List<Person> persons = new ArrayList<Person>();  
  
 public Data() {  
 }  
  
 Data(List<Person> persons){  
 this.persons = persons;  
 }  
 void add(Person... people){  
 for (Person p : people){  
 persons.add(p);  
 }  
 }  
 void print(){  
 for(Person p : persons){  
 System.*out*.println(p);  
 }  
 }  
 void searchByLastName(String lastName){  
 int count = 0;  
 for(Person p : persons){  
 if (p.getLastName().equals(lastName)){  
 System.*out*.println(p);  
 count++;  
 }  
 }  
 if(count == 0){  
 System.*out*.println("no such people");  
 }  
 }  
 void searchByCountry(String country){  
 int count = 0;  
 for(Person p : persons){  
 if (p.getAddress().getCountry().equals(country)){  
 System.*out*.println(p);  
 count++;  
 }  
 }  
 if(count == 0){  
 System.*out*.println("no such people");  
 }  
 }  
 void searchByCity(String city){  
 int count=0;  
 for(Person p : persons){  
 if (p.getAddress().getCity().equals(city)){  
 System.*out*.println(p);  
 count++;  
 }  
 }  
 if(count == 0){  
 System.*out*.println("no such people");  
 }  
 }  
 void searchByStreet(String street){  
 int count=0;  
 for(Person p : persons){  
 if (p.getAddress().getStreet().equals(street)){  
 System.*out*.println(p);  
 count++;  
 }  
 }  
 if(count == 0){  
 System.*out*.println("no such people");  
 }  
 }  
 void searchByAddress(Address address){  
 int count=0;  
 for(Person p : persons){  
 if (p.getAddress().equals(address)){  
 System.*out*.println(p);  
 count++;  
 }  
 }  
 if(count == 0){  
 System.*out*.println("no such people");  
 }  
 }  
 void printBetweenDates(Calendar c1, Calendar c2){  
 int count = 0;  
 for(Person p : persons){  
 if(p.getBirthday().compareTo(c1) >= 0 && p.getBirthday().compareTo(c2) <= 0){  
 System.*out*.println(p);  
 count++;  
 }  
 }  
 if(count == 0){  
 System.*out*.println("no such people");  
 }  
 }  
 void youngest(){  
 Person youngest = persons.get(0);  
 for(Person p : persons){  
 if(!p.getBirthday().before(youngest.getBirthday())){  
 youngest = p;  
 }  
 }  
 System.*out*.println(youngest);  
 }  
 void oldest(){  
 Person old = persons.get(0);  
 for(Person p : persons){  
 if(p.getBirthday().before(old.getBirthday())){  
 old = p;  
 }  
 }  
 System.*out*.println(old);  
 }  
 void oneStreet(){  
 /\*  
 В моей реализации жить на одной улицу = одинаковые адреса  
 \*/  
 Set<Address> streets2 = new HashSet<>();  
 List<Person> people2 = new ArrayList<>();  
 for(int i=0;i<persons.size();i++){  
 if(streets2.contains(persons.get(i).getAddress())) continue;  
 people2.add(persons.get(i));  
 for(int j=i+1;j<persons.size();j++){  
 if (persons.get(i).getAddress().equals(persons.get(j).getAddress())){  
 people2.add(persons.get(j));  
 streets2.add(persons.get(i).getAddress());  
 }  
 }  
 if(people2.size() > 1)  
 for(Person p : people2){  
 System.*out*.println(p);  
 }  
 people2.clear();  
 }  
 }  
}

package task1.address;  
  
import java.util.Calendar;  
import java.util.Date;  
import java.util.GregorianCalendar;  
  
public class Main {  
 public static void main(String[] args) {  
 Calendar calendar1 = new GregorianCalendar(2001, 0,14);  
 Address address1 = new Address("Russia", "Moscow", "Lenina");  
 Person p1 = new Person("Alexey", "Volkov", calendar1, address1);  
  
 Calendar calendar2 = new GregorianCalendar(1999, 3,5);  
 Address address2 = new Address("Russia", "Moscow", "Lenina");  
 Person p2 = new Person("Sergey", "Petrov", calendar2, address2);  
  
 Calendar calendar3 = new GregorianCalendar(1975, 5,6);  
 Address address3 = new Address("Italy", "Rome", "Main");  
 Person p3 = new Person("ItalicName", "ItalicSurname", calendar3, address3);  
  
 Calendar calendar4 = new GregorianCalendar(2006, 9,25);  
 Address address4 = new Address("USA", "New-York", "Wall");  
 Person p4 = new Person("Jhon", "Walker", calendar4, address4);  
  
 Data data = new Data();  
 data.add(p1,p2,p3,p4);  
  
 data.print();  
 System.*out*.println("\n");  
  
 System.*out*.println("SEARCHING last name: ItalicSurname");  
 data.searchByLastName("ItalicSurname");  
  
 System.*out*.println("SEARCHING city: NY");  
 data.searchByCity("New-York");  
  
 System.*out*.println("SEARCHING country: Russia");  
 data.searchByCountry("Russia");  
  
 System.*out*.println("SEARCHING street: Wall");  
 data.searchByStreet("Wall");  
  
 System.*out*.println("SEARCHING city: Zelenograd");  
 data.searchByCity("Zelenograd");  
  
 System.*out*.println("BETWEEN DATES: 13.07.1990 AND 25.09.2007");  
 data.printBetweenDates(new GregorianCalendar(1990,6,13),  
 new GregorianCalendar(2007, 8, 26));  
  
 System.*out*.println("YOUNGEST:");  
 data.youngest();  
  
 System.*out*.println("OLDEST: ");  
 data.oldest();  
  
 System.*out*.println("ON ONE STREET: ");  
 data.oneStreet();  
 }  
}

name='Alexey', lastName='Volkov', birthday=Sun Jan 14 00:00:00 MSK 2001, address : country='Russia', city='Moscow', street='Lenina'

name='Sergey', lastName='Petrov', birthday=Mon Apr 05 00:00:00 MSD 1999, address : country='Russia', city='Moscow', street='Lenina'

name='ItalicName', lastName='ItalicSurname', birthday=Fri Jun 06 00:00:00 MSK 1975, address : country='Italy', city='Rome', street='Main'

name='Jhon', lastName='Walker', birthday=Wed Oct 25 00:00:00 MSD 2006, address : country='USA', city='New-York', street='Wall'

SEARCHING last name: ItalicSurname

name='ItalicName', lastName='ItalicSurname', birthday=Fri Jun 06 00:00:00 MSK 1975, address : country='Italy', city='Rome', street='Main'

SEARCHING city: NY

name='Jhon', lastName='Walker', birthday=Wed Oct 25 00:00:00 MSD 2006, address : country='USA', city='New-York', street='Wall'

SEARCHING country: Russia

name='Alexey', lastName='Volkov', birthday=Sun Jan 14 00:00:00 MSK 2001, address : country='Russia', city='Moscow', street='Lenina'

name='Sergey', lastName='Petrov', birthday=Mon Apr 05 00:00:00 MSD 1999, address : country='Russia', city='Moscow', street='Lenina'

SEARCHING street: Wall

name='Jhon', lastName='Walker', birthday=Wed Oct 25 00:00:00 MSD 2006, address : country='USA', city='New-York', street='Wall'

SEARCHING city: Zelenograd

no such people

BETWEEN DATES: 13.07.1990 AND 25.09.2007

name='Alexey', lastName='Volkov', birthday=Sun Jan 14 00:00:00 MSK 2001, address : country='Russia', city='Moscow', street='Lenina'

name='Sergey', lastName='Petrov', birthday=Mon Apr 05 00:00:00 MSD 1999, address : country='Russia', city='Moscow', street='Lenina'

name='Jhon', lastName='Walker', birthday=Wed Oct 25 00:00:00 MSD 2006, address : country='USA', city='New-York', street='Wall'

YOUNGEST:

name='Jhon', lastName='Walker', birthday=Wed Oct 25 00:00:00 MSD 2006, address : country='USA', city='New-York', street='Wall'

OLDEST:

name='ItalicName', lastName='ItalicSurname', birthday=Fri Jun 06 00:00:00 MSK 1975, address : country='Italy', city='Rome', street='Main'

ON ONE STREET:

name='Alexey', lastName='Volkov', birthday=Sun Jan 14 00:00:00 MSK 2001, address : country='Russia', city='Moscow', street='Lenina'

name='Sergey', lastName='Petrov', birthday=Mon Apr 05 00:00:00 MSD 1999, address : country='Russia', city='Moscow', street='Lenina'

Process finished with exit code 0