Оглавление

[Задание 1. 2](#_Toc77533628)

[Задание 2. 3](#_Toc77533629)

[Задание 3. 9](#_Toc77533630)

[Задание 4. 10](#_Toc77533631)

[Ссылка на GitHub 11](#_Toc77533632)

# Задание 1.

package trainingTask2.p1;  
  
public interface Voice {  
 void voice();  
}

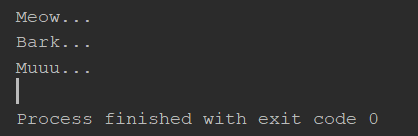
package trainingTask2.p1;  
  
public class Cat implements Voice {  
 @Override  
 public void voice() {  
 System.*out*.println("Meow...");  
 }  
}

package trainingTask2.p1;  
  
public class Dog implements Voice {  
 @Override  
 public void voice() {  
 System.*out*.println("Bark...");  
 }  
}

package trainingTask2.p1;  
  
public class Cow implements Voice {  
 @Override  
 public void voice() {  
 System.*out*.println("Muuu...");  
 }  
}

package trainingTask2.p1;  
  
public class Main {  
 public static void main(String[] args){  
 Cat cat = new Cat();  
 Dog dog = new Dog();  
 Cow cow = new Cow();  
  
 cat.voice();  
 dog.voice();  
 cow.voice();  
 }  
}

Результат выполнения:

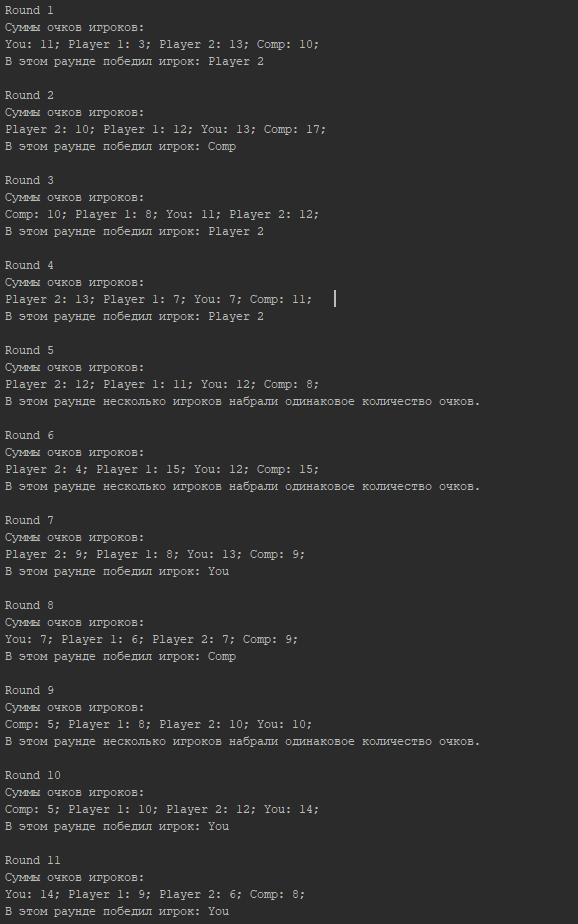


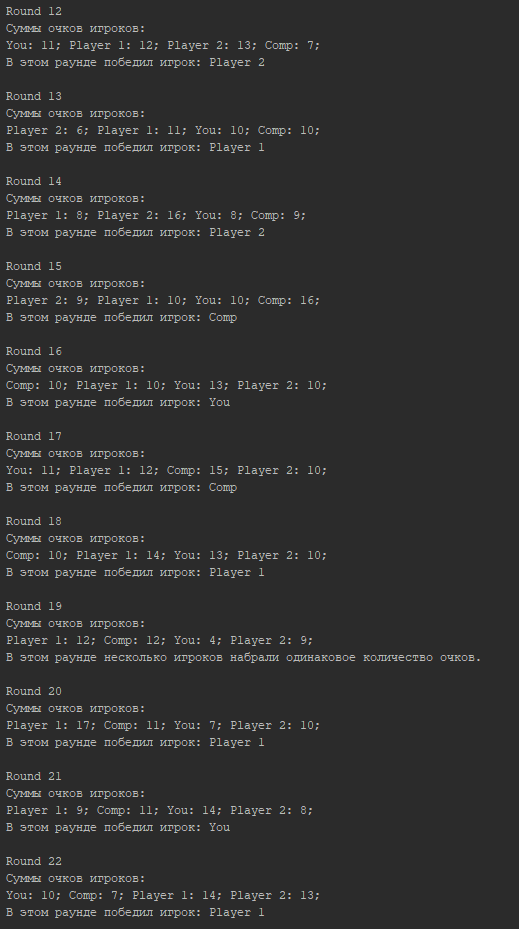
# Задание 2.

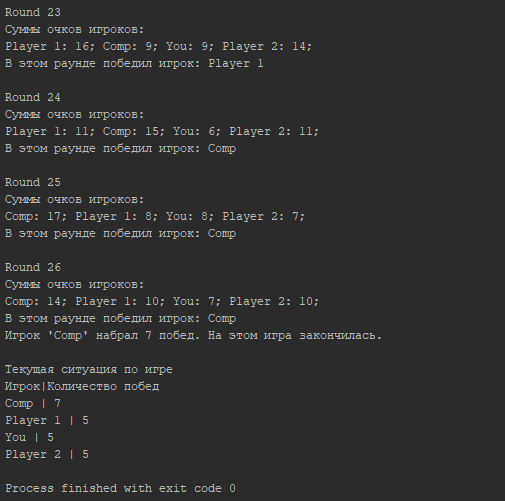
package trainingTask2.p2;  
  
public interface Game {  
 //метод инициализации начала игры  
 void initNewGame(int N,int K);  
 //метод проверки активности игры  
 boolean isActive();  
 //метод проведения следующего раунда игры  
 void nextRound();  
 //метод вывода таблицы игроков  
 void checkTable();  
}

package trainingTask2.p2;  
  
public class DiceGameImpl {  
 public static void main(String[] args){  
 int rnd = 1;  
 int N = 4;  
 int K = 3;  
 DiceGame game = new DiceGame();  
 game.initNewGame(N,K);  
 while (game.isActive()){  
 System.*out*.println("Round "+rnd);  
 game.nextRound();  
 rnd++;  
 System.*out*.println();  
 }  
 game.checkTable();  
 }  
}  
  
class DiceGame implements Game{  
 private int dicesNumber;  
 private Player[] players;  
 private boolean activeness = false;  
  
 //метод для инициализации начала игры  
 @Override  
 public void initNewGame(int N,int K){  
 this.dicesNumber = K;  
 players = new Player[N];  
 players[0] = new Player("You");  
 for (int i=1; i < N-1; i++){  
 players[i] = new Player("Player "+i);  
 }  
 players[N-1] = new Player("Comp");  
 activeness = true;  
 }  
  
 @Override  
 public boolean isActive(){  
 return this.activeness;  
 }  
  
 //метод поиска победителей  
 private static int findWinner(int[] cnt){  
 int index = 0;  
 int max = 0;  
 for (int i=0; i<cnt.length; i++){  
 if (max < cnt[i]){  
 max = cnt[i];  
 index = i;  
 }  
 }  
 return index;  
 }  
  
 //метод проверка единственности победителя  
 private static boolean isOneWinner(int[] cnt){  
 int max = 0;  
 int counts = 0;  
 for (int i=0; i<cnt.length; i++){  
 if (max < cnt[i]){max = cnt[i];}  
 }  
 for (int i=0; i<cnt.length; i++){  
 if (max == cnt[i]){counts++;}  
 }  
 if (counts==1) {return true;}  
 else {return false;}  
 }  
  
 //метод - проведение раунда  
 @Override  
 public void nextRound(){  
 if (!this.isActive()){  
 System.*out*.println("Игра не активна. Начните новую.");  
 return;  
 }  
 int[] cnt= new int[players.length];  
 int winner;  
 System.*out*.println("Суммы очков игроков:");  
 for (int j=0; j < players.length;j++){  
 for (int i=0; i < dicesNumber; i++){  
 cnt[j] += players[j].rollDices();  
 }  
 System.*out*.print(players[j].getName()+": "+cnt[j]+"; ");  
 }  
 System.*out*.println();  
 winner = DiceGame.*findWinner*(cnt);  
 if (!DiceGame.*isOneWinner*(cnt)) {  
 System.*out*.println("В этом раунде несколько игроков набрали одинаковое количество очков.");  
 }else {  
 System.*out*.println("В этом раунде победил игрок: " + players[winner].getName());  
 players[winner].addWin();  
 Player tmp = players[winner];  
 players[winner] = players[0];  
 players[0] = tmp;  
  
 if (players[0].getWins() == 7){  
 System.*out*.println("Игрок '" + players[0].getName()+"' набрал 7 побед. На этом игра закончилась.");  
 activeness = false;  
 }  
 }  
 }  
 @Override  
 public void checkTable(){  
 System.*out*.println("Текущая ситуация по игре");  
 System.*out*.println("Игрок|Количество побед");  
 for (int i = 0; i<players.length; i++){  
 System.*out*.println(players[i].getName()+" | "+players[i].getWins());  
 }  
 }  
  
}  
  
class Player{  
 private int wins;  
 private String name;  
  
 public Player(){}  
  
 public Player(String name){  
 this.name = name;  
 }  
  
 public int getWins(){  
 return wins;  
 }  
  
 public void addWin(){  
 this.wins++;  
 }  
  
 public String getName(){  
 return this.name;  
 }  
  
 //имитация бросания кости  
 public int rollDices(){  
 return (int) (Math.*random*() \* 6) + 1;  
 }  
}

Результат выполнения:







# Задание 3.

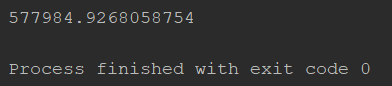
package trainingTask2.p3;  
  
public class ExtendedClass {  
 byte b;  
 int i;  
 double d;  
 String s;  
  
 @Override  
 public boolean equals(Object anObject) {  
 if (this == anObject) return true;  
 if (anObject == null || getClass() != anObject.getClass()) return false;  
 ExtendedClass that = (ExtendedClass) anObject;  
 return this.b == that.b &&  
 this.i == that.i &&  
 this.d == that.d &&  
 s.equals(that.s);  
 }  
  
 @Override  
 public int hashCode() {  
 final int prime = 31;  
 int result = 1;  
 result = result\*prime + b;  
 result = result\*prime + i;  
 result = result\*prime + (d == 0 ? 0 : ((Double)d).hashCode());  
 result = result\*prime + (s == null ? 0 : s.hashCode());  
 return result;  
 }  
  
 @Override  
 public String toString() {  
 return "ExtendedClass{" +  
 "b=" + b +  
 ", i=" + i +  
 ", d=" + d +  
 ", s='" + s + '\'' +  
 '}';  
 }  
}

# Задание 4.

package trainingTask2.p4;  
  
public interface TwoNumbers {  
 void setX();  
  
 void setY();  
  
 void getDifference();  
}

package trainingTask2.p4;  
  
public class MainClass{  
 public static void main(String[] args){  
 TheNumbersImpl tni = new TheNumbersImpl();  
 tni.setX();  
 tni.setY();  
 tni.getDifference();  
 }  
}  
  
class TheNumbersImpl implements TwoNumbers{  
 private double x;  
 private double y;  
 final double limit = 1000000;  
  
 //должен устанавливать значение поля x  
 @Override  
 public void setX() {  
 x = limit\*(Math.*random*()\*2 - 1);  
 }  
 //должен устанавливать значение поля y  
 @Override  
 public void setY() {  
 y = limit\*(Math.*random*()\*2 - 1);  
 }  
 //печатает на экран абсолютное значение разности x и y  
 @Override  
 public void getDifference() {  
 System.*out*.println(Math.*abs*(x-y));  
 }  
}

Результаты выполнения:



# Ссылка на GitHub

[yagorkas/Training-Tasks (github.com)](https://github.com/yagorkas/Training-Tasks)