**Лабораторная работа №4**

**Задание 1.**

Задание состоит из двух частей.

1. Описать структуру данных класса Character, взяв за основу описание характеристик персонажей в Genshin Impact (или любой другой RPG, но не менее 5 базовых характеристик):

* Максимальное кол-во HP;
* Сила атаки; (ATK)
* Защита; (DEF)
* Мастерство стихий; (ElementalMaster)
* Выносливость. (Stamina)
* Тип Архонта (для геншина) или специализация (для других RPG);

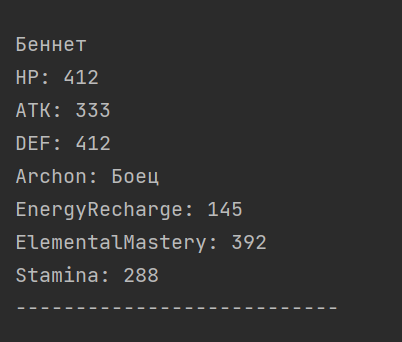
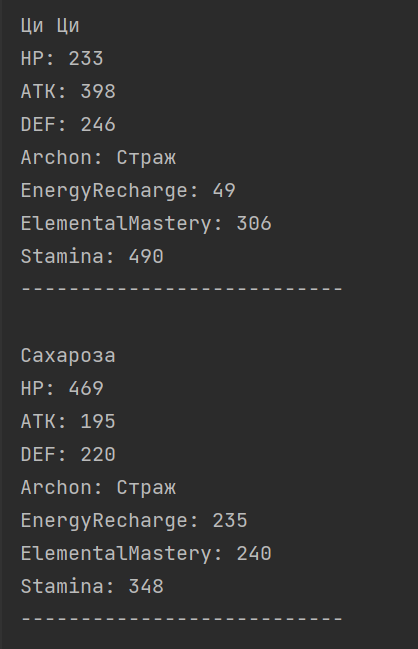
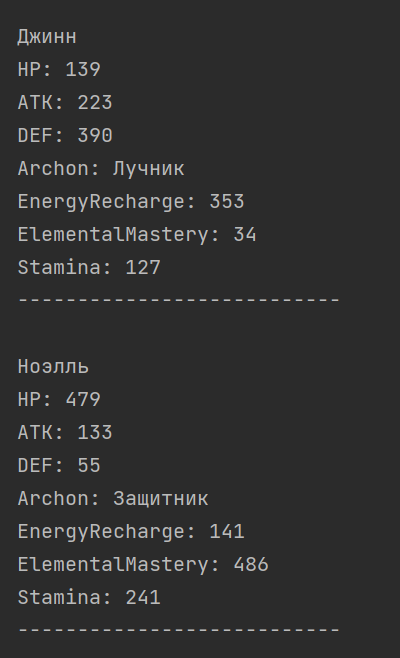
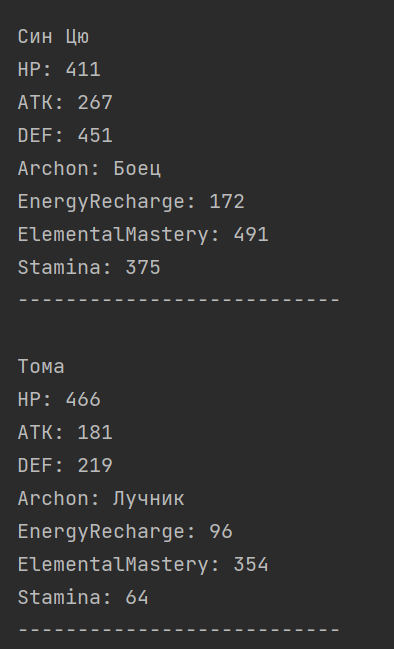
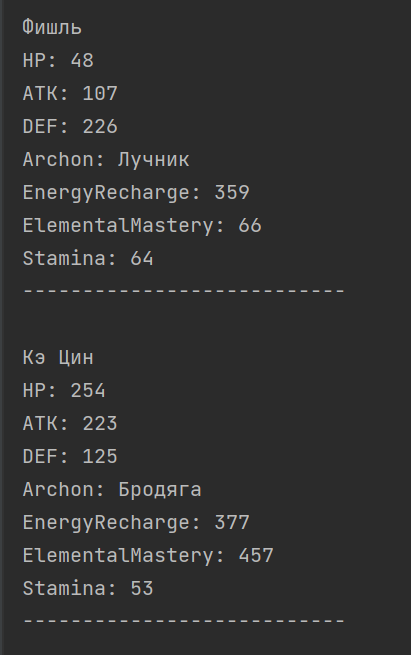
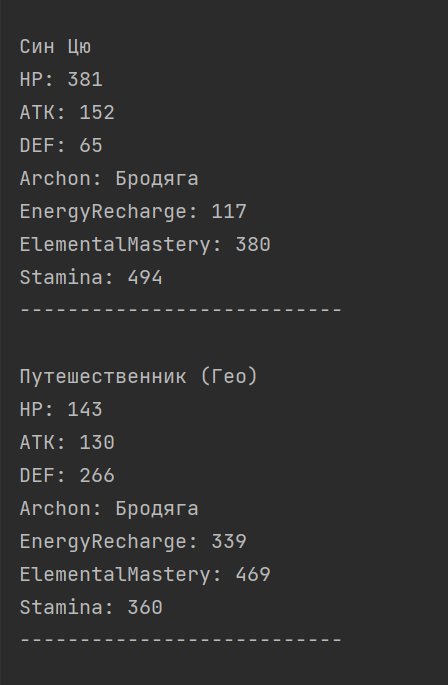
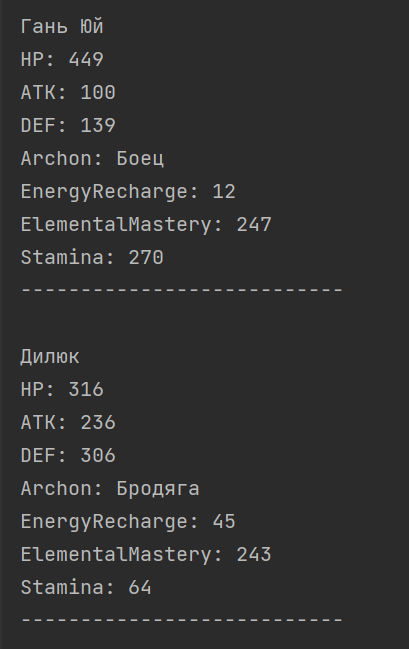
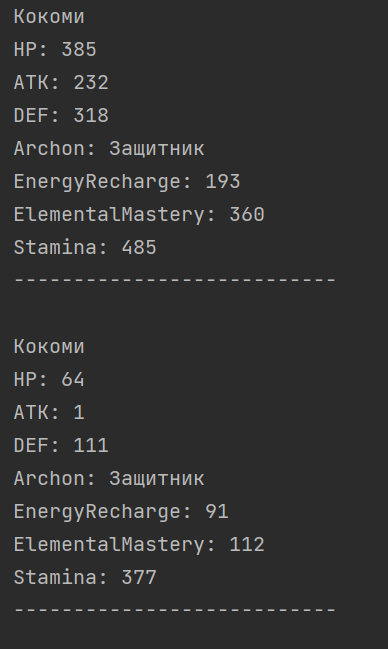
Для специализации я взяла характеристику – Тип персонажа - Archon (лучник, охотник, воин, бродяга и т.д.).

Далее взять одну произвольную характеристику из списка второстепенных:

* Шанс крит.попадания;
* Крит.урон;
* Бонус лечения;
* Бонус получаемого лечения;
* Восст. энергии;
* Снижение времени отката;
* Прочность щита

Я взяла характеристику – восстановление энергии (EnergyRecharge).

**Набор исходных данных:**

****

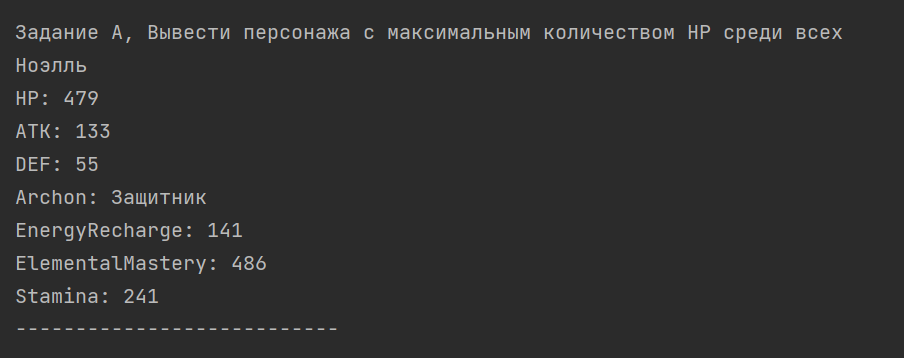
2. С имеющейся коллекцией объектов, при помощи методов Stream API произвести следующие действия:

**a. Вывести персонажа с максимальным количеством HP среди всех;**

**Код:**

*//A*Optional<Character> character = *heroes*.stream().max(Comparator.*comparing*(Character::getHp));  
System.*out*.println("Задание A, Вывести персонажа с максимальным количеством HP среди всех\n" + character.get());

**Вывод:**

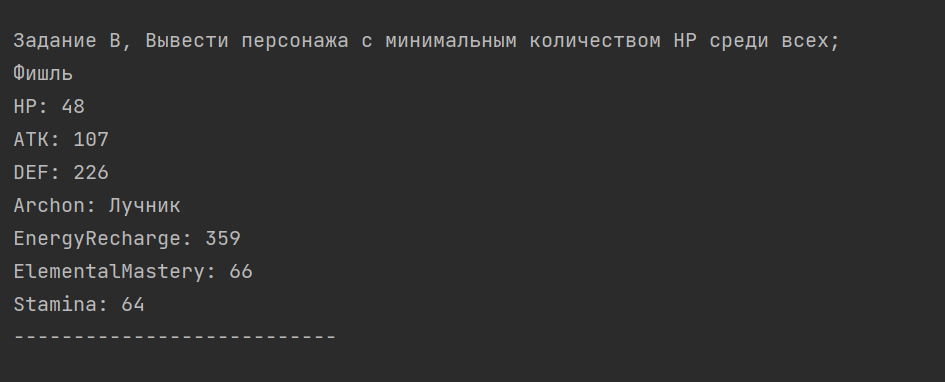


**b. Вывести персонажа с минимальным количеством HP среди всех;**

**Код:**

*//B*character = *heroes*.stream().min(Comparator.*comparing*(Character::getHp));  
System.*out*.println("Задание B, Вывести персонажа с минимальным количеством HP среди всех;\n" + character.get());

**Вывод:**

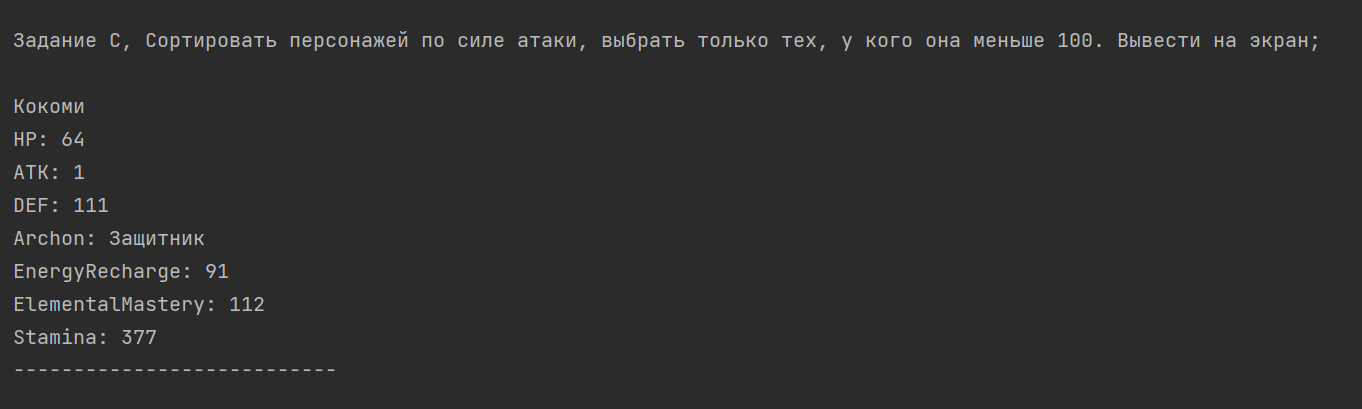
****

**c. Сортировать персонажей по силе атаки, выбрать только тех, у кого она меньше 100. Вывести на экран;**

**Код:**

*//C*System.*out*.println("Задание C, Сортировать персонажей по силе атаки, выбрать только тех, у кого " +  
 "она меньше 100. Вывести на экран;\n");  
*heroes*.stream().sorted(Comparator.*comparing*(Character::getAtk)).filter(x -> x.getAtk() < 100)  
 .forEach(System.*out*::println);

**Вывод:**

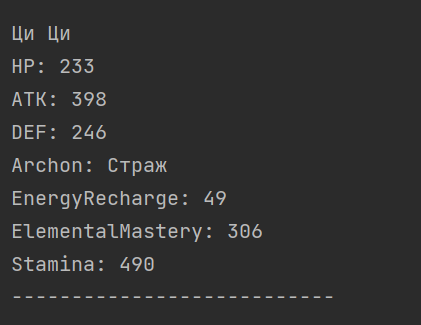
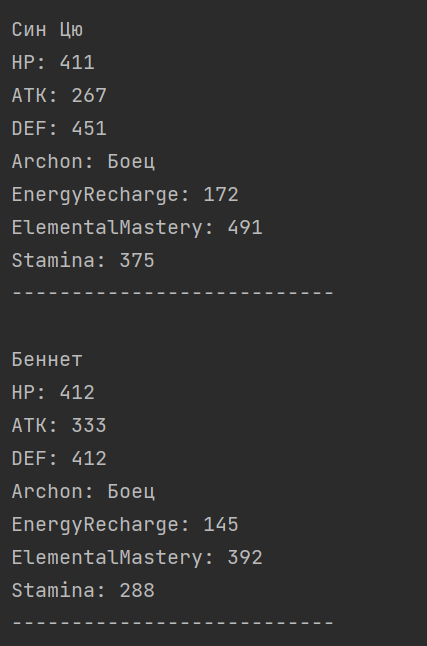
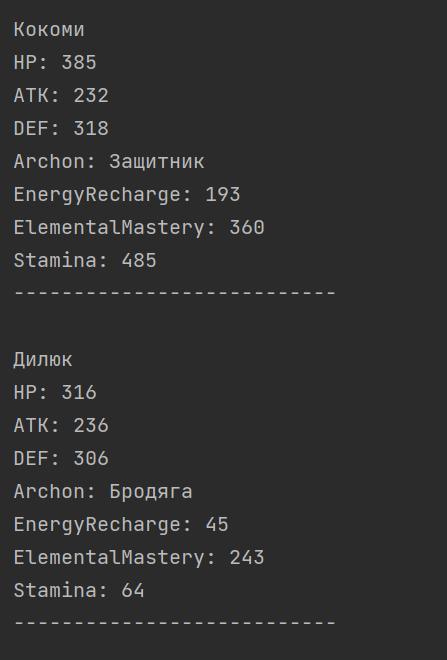
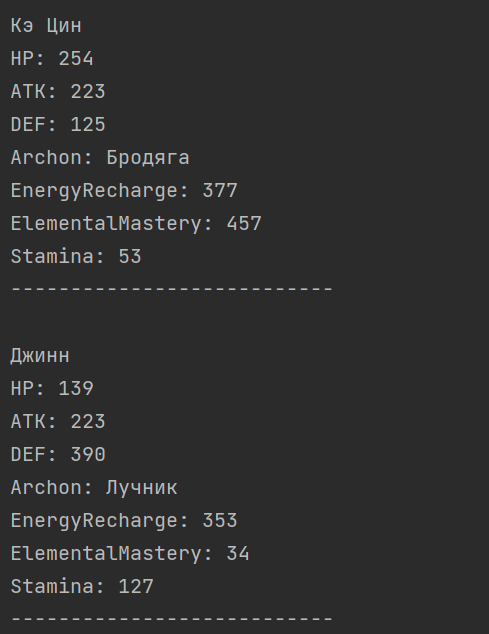
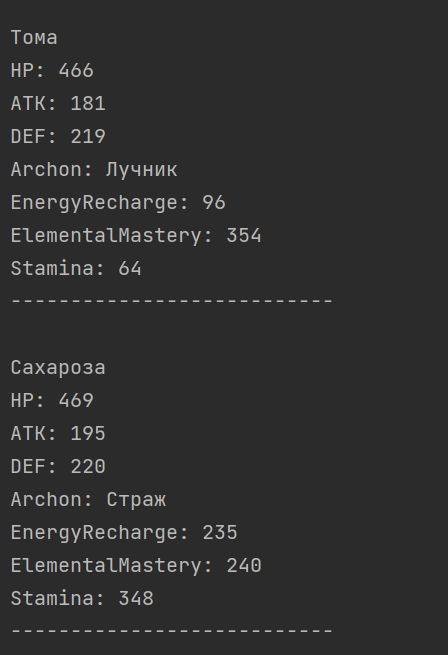
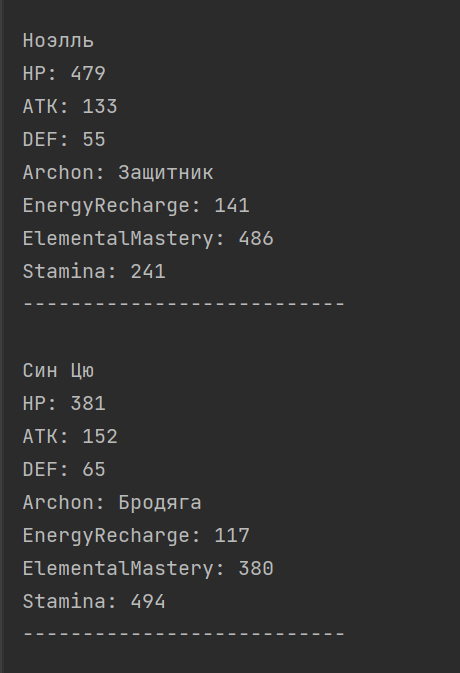
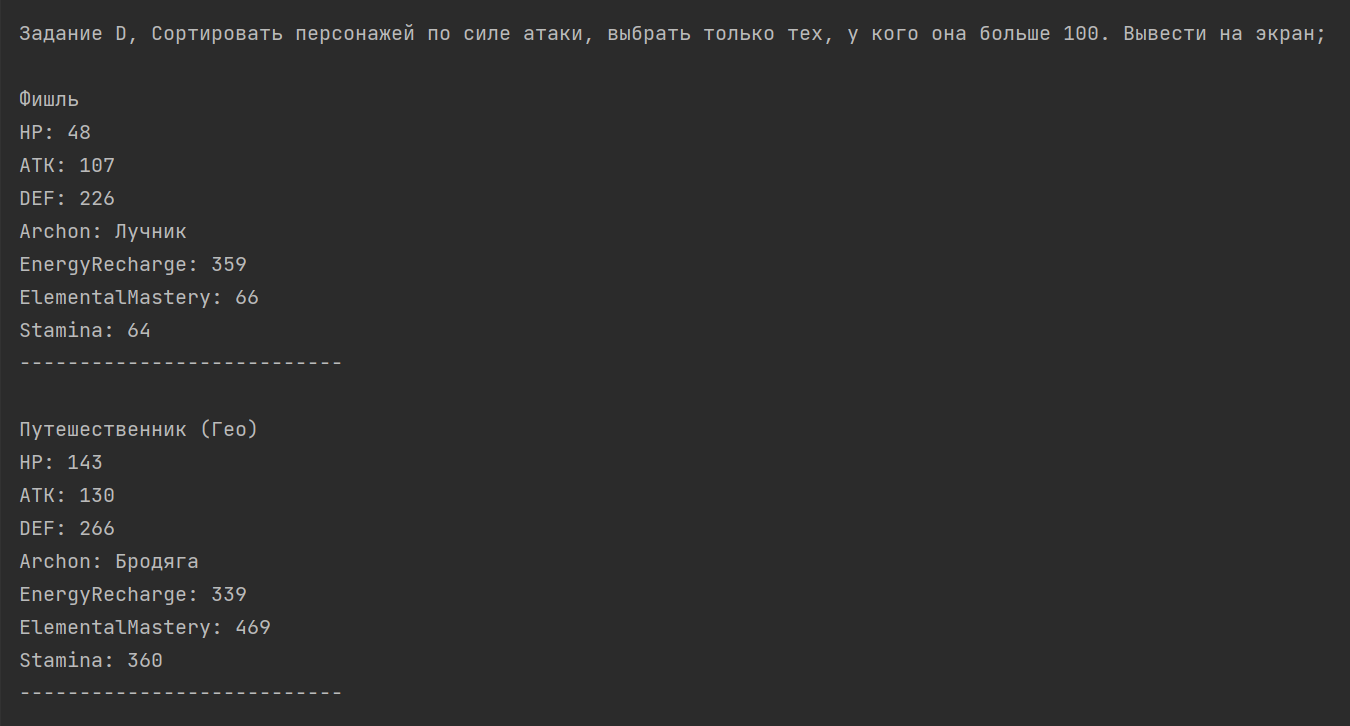
****

**d. Тоже самое, что в (c), но убрать из выборки всех, у кого меньше 100;**

**Код:**

*//D*System.*out*.println("Задание D, Сортировать персонажей по силе атаки, выбрать только тех, у кого " +  
 "она больше 100. Вывести на экран;\n");  
*heroes*.stream().sorted(Comparator.*comparing*(Character::getAtk)).filter(x -> x.getAtk() > 100)  
 .forEach(System.*out*::println);

**Вывод:**

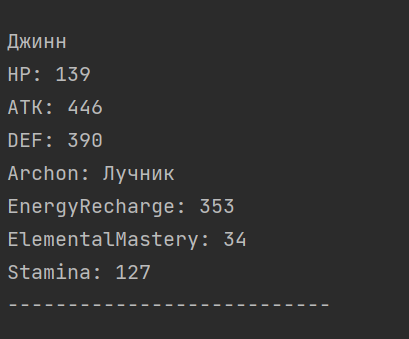
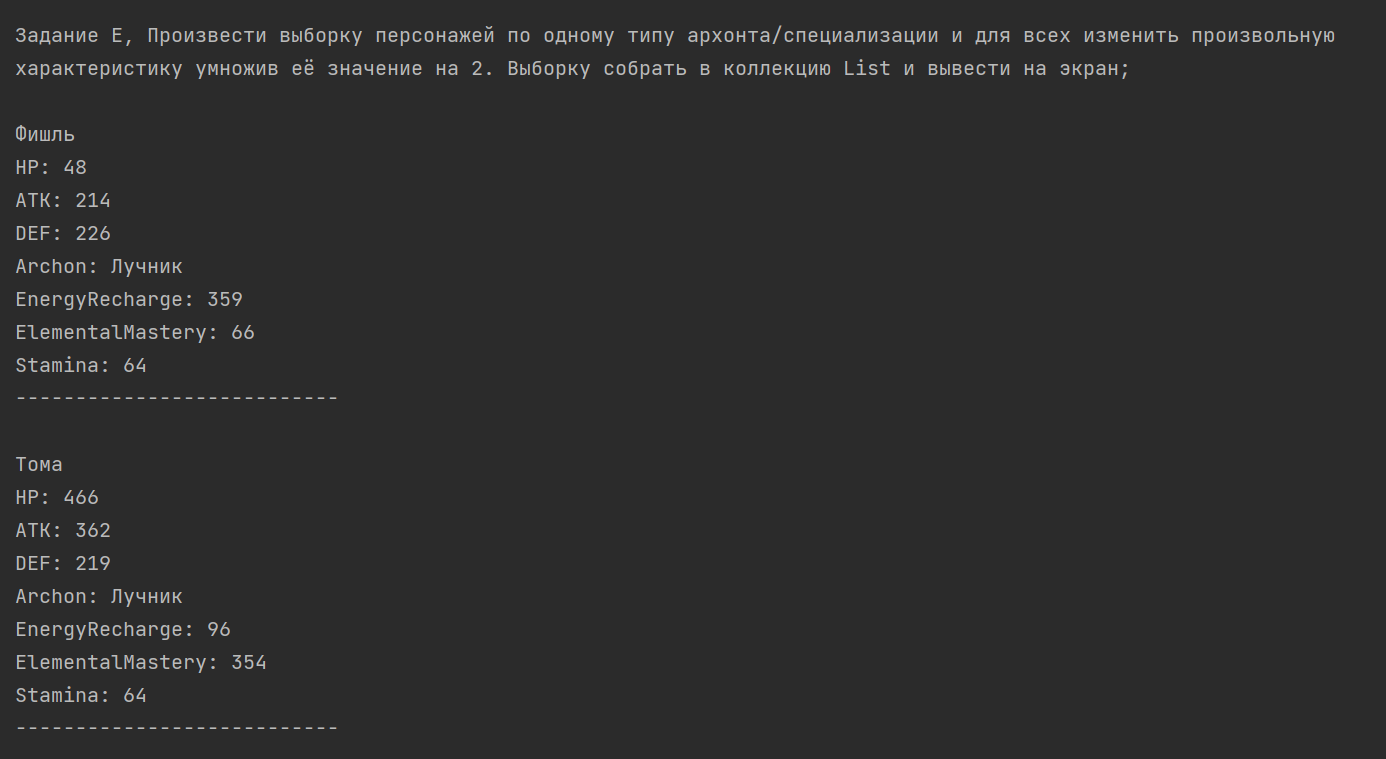
****

**e. Произвести выборку персонажей по одному типу архонта/специализации и для всех изменить произвольную характеристику умножив её значение на 2. Выборку собрать в коллекцию List и вывести на экран;**

**Код:**

*//E*List<Character> list = *heroes*.stream().filter(x -> Objects.*equals*(x.getArchon(), "Лучник"))  
 .peek(x -> x.setAtk(x.getAtk() \* 2))  
 .collect(Collectors.*toList*());  
System.*out*.println("Задание E, Произвести выборку персонажей по одному типу " +  
 "архонта/специализации и для всех изменить произвольную\n" +  
 "характеристику умножив её значение на 2. Выборку собрать в " +  
 "коллекцию List и вывести на экран;\n");  
list.stream().forEach(System.*out*::println);

**Вывод:**

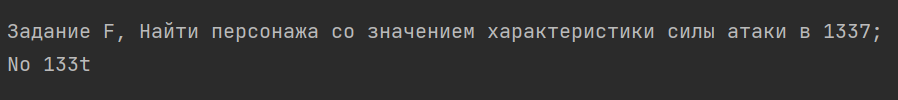
****

**f. Найти персонажа со значением характеристики силы атаки в 1337, вывод функции .get() произвести в объект класса Optional. Если такой объект существует, то вывести на экран его и добавить перед его именем слово l33t. Если такого объекта нет, вывести на экран “No l33t”);**

**Код:**

*//F  
// heroes.get(5).setAtk(1337);* character = *heroes*.stream().filter(x -> x.getAtk() == 1337).findFirst();  
 System.*out*.println("Задание F, Найти персонажа со значением характеристики силы атаки в 1337; \n" +  
 (character.isEmpty() ? "No 133t\n" : "133t " + character.get()));

**Вывод:**

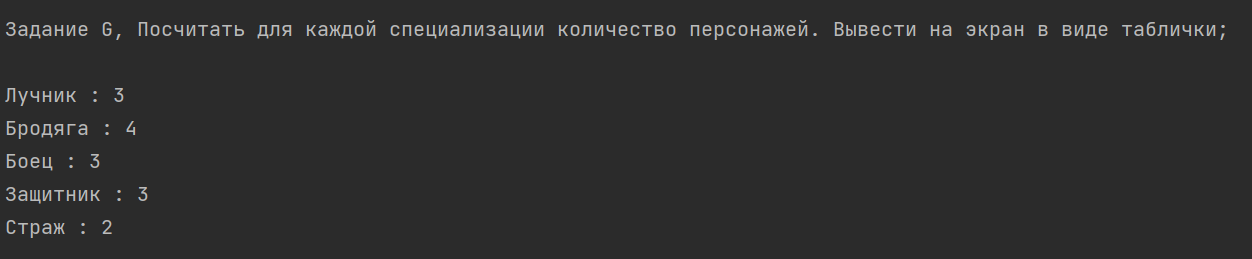
****

**g. Посчитать для каждой специализации количество персонажей. Вывести на экран в виде таблички.**

**Код:**

*//G*Map<String, Long> map = *heroes*.stream().collect(*groupingBy*(x -> x.getArchon(), Collectors.*counting*()));  
System.*out*.println("Задание G, Посчитать для каждой специализации количество персонажей. " +  
 "Вывести на экран в виде таблички;\n");  
map.entrySet().forEach(entry -> {  
 System.*out*.println(entry.getKey() + " : " + entry.getValue());  
});

**Вывод:**

****

**Примечание.**

В задание F был пункт про характеристику атаки равной в 1337. В моей игре данная характеристика может принимать значения от 1 до 500, следовательно, этот пункт задачи всегда будет выводить “No 133t”. Но это никак не отменяет того, что решение данного пункта абсолютно рабочее.

**Приложение.**

**Class Character**

public class Character {  
  
 private String name;  
 private int hp;  
 private int atk;  
 private int def;  
 private int elementalMastery;  
 private int stamina;  
 private String archon;  
 private int energyRecharge;  
  
 public int getEnergyRecharge() {  
 return energyRecharge;  
 }  
  
 public void setEnergyRecharge(int energyRecharge) {  
 this.energyRecharge = energyRecharge;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public int getHp() {  
 return hp;  
 }  
  
 public void setHp(int hp) {  
 this.hp = hp;  
 }  
  
 public int getAtk() {  
 return atk;  
 }  
  
 public void setAtk(int atk) {  
 this.atk = atk;  
 }  
  
 public int getDef() {  
 return def;  
 }  
  
 public void setDef(int def) {  
 this.def = def;  
 }  
  
 public int getElementalMastery() {  
 return elementalMastery;  
 }  
  
 public void setElementalMastery(int elementalMastery) {  
 this.elementalMastery = elementalMastery;  
 }  
  
 public int getStamina() {  
 return stamina;  
 }  
  
 public void setStamina(int stamina) {  
 this.stamina = stamina;  
 }  
  
 public String getArchon() {  
 return archon;  
 }  
  
 public void setArchon(String archon) {  
 this.archon = archon;  
 }  
  
 public Character(String name, int hp, int atk, int def, int elementalMastery, int stamina, String archon, int energyRecharge) {  
 this.name = name;  
 this.hp = hp;  
 this.atk = atk;  
 this.def = def;  
 this.elementalMastery = elementalMastery;  
 this.stamina = stamina;  
 this.archon = archon;  
 this.energyRecharge = energyRecharge;  
 }  
  
 public Character() {  
 name = "Name";  
 archon = "Бродяга";  
 hp = atk = def = elementalMastery = stamina = energyRecharge = 10;  
 }  
  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Character character = (Character) o;  
 return def == character.def  
 && hp == character.hp  
 && atk == character.atk  
 && Objects.*equals*(archon, character.archon)  
 && energyRecharge == character.energyRecharge  
 && elementalMastery == character.elementalMastery  
 && stamina == character.stamina  
 && Objects.*equals*(name, character.name);  
 }  
  
 @Override  
 public String toString() {  
 return new String()  
 .concat(*/\*"Name: " +\*/* getName())  
 .concat("\n")  
 .concat("HP: " + getHp())  
 .concat("\n")  
 .concat("ATK: " + getAtk())  
 .concat("\n")  
 .concat("DEF: " + getDef())  
 .concat("\n")  
 .concat("Archon: " + getArchon())  
 .concat("\n")  
 .concat("EnergyRecharge: " + getEnergyRecharge())  
 .concat("\n")  
 .concat("ElementalMastery: " + getElementalMastery())  
 .concat("\n")  
 .concat("Stamina: " + getStamina())  
 .concat("\n---------------------------\n");  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(name, hp, atk, def, archon, energyRecharge, elementalMastery, stamina);  
 }  
}

**class Main**

public class Main {  
  
 private static ArrayList<Character> *heroes* = new ArrayList<>();  
  
 public static void main(String[] args) {  
  
 *generateCharacters*();  
 *heroes*.stream().forEach(System.*out*::println);  
  
 *//A* Optional<Character> character = *heroes*.stream().max(Comparator.*comparing*(Character::getHp));  
 System.*out*.println("Задание A, Вывести персонажа с максимальным количеством HP среди всех\n" + character.get());  
  
 *//B* character = *heroes*.stream().min(Comparator.*comparing*(Character::getHp));  
 System.*out*.println("Задание B, Вывести персонажа с минимальным количеством HP среди всех;\n" + character.get());  
  
 *//C* System.*out*.println("Задание C, Сортировать персонажей по силе атаки, выбрать только тех, у кого " +  
 "она меньше 100. Вывести на экран;\n");  
 *heroes*.stream().sorted(Comparator.*comparing*(Character::getAtk)).filter(x -> x.getAtk() < 100)  
 .forEach(System.*out*::println);  
  
 *//D* System.*out*.println("Задание D, Сортировать персонажей по силе атаки, выбрать только тех, у кого " +  
 "она больше 100. Вывести на экран;\n");  
 *heroes*.stream().sorted(Comparator.*comparing*(Character::getAtk)).filter(x -> x.getAtk() > 100)  
 .forEach(System.*out*::println);  
  
 *//E* List<Character> list = *heroes*.stream().filter(x -> Objects.*equals*(x.getArchon(), "Лучник"))  
 .peek(x -> x.setAtk(x.getAtk() \* 2))  
 .collect(Collectors.*toList*());  
 System.*out*.println("Задание E, Произвести выборку персонажей по одному типу " +  
 "архонта/специализации и для всех изменить произвольную\n" +  
 "характеристику умножив её значение на 2. Выборку собрать в " +  
 "коллекцию List и вывести на экран;\n");  
 list.stream().forEach(System.*out*::println);  
  
 *//F  
// heroes.get(5).setAtk(1337);* character = *heroes*.stream().filter(x -> x.getAtk() == 1337).findFirst();  
 System.*out*.println("Задание F, Найти персонажа со значением характеристики силы атаки в 1337; \n" +  
 (character.isEmpty() ? "No 133t\n" : "133t " + character.get()));  
  
 *//G* Map<String, Long> map = *heroes*.stream().collect(*groupingBy*(x -> x.getArchon(), Collectors.*counting*()));  
 System.*out*.println("Задание G, Посчитать для каждой специализации количество персонажей. " +  
 "Вывести на экран в виде таблички;\n");  
 map.entrySet().forEach(entry -> {  
 System.*out*.println(entry.getKey() + " : " + entry.getValue());  
 });  
 }  
  
  
 private static void generateCharacters() {  
 Random rnd = new Random();  
 int count = rnd.nextInt(30) + 15;  
 String[] names = {"Элой", "Кокоми", "Баал", "Сара", "Саю", "Тома", "Розария", "Альбедо",  
 "Барбара", "Бэй Доу", "Беннет", "Чун Юнь", "Дилюк", "Диона", "Фишль", "Гань Юй",  
 "Джинн", "Ху Тао", "Кэйа", "Кэ Цин", "Кли", "Лиза", "Мона", "Нин Гуан", "Ноэлль",  
 "Ци Ци", "Рэйзор", "Сахароза", "Путешественник (Гео)", "Путешественник (Анемо)", "Тарталья",  
 "Венти", "Сян Лин", "Сяо", "Син Цю"};  
 String[] archons = {"Лучник", "Боец", "Страж", "Бродяга", "Защитник"};  
 while(count-- > 0) {  
 String name = names[rnd.nextInt(names.length)];  
 int hp = rnd.nextInt(500) + 1;  
 int atk = rnd.nextInt(500) + 1;  
 int def = rnd.nextInt(500) + 1;  
 int elementalMastery = rnd.nextInt(500) + 1;  
 int stamina = rnd.nextInt(500) + 1;  
 String archon = archons[rnd.nextInt(archons.length)];  
 int energyRecharge = rnd.nextInt(500) + 1;  
 *heroes*.add(new Character(name, hp, atk, def, elementalMastery, stamina, archon, energyRecharge));  
 }  
 }  
}