\*Ссылка на резюме: <a href="https://hh.ru/resume/d74e6b15ff0798e3ba0039ed1f647072793333">https://hh.ru/resume/d74e6b15ff0798e3ba0039ed1f647072793333</a>

\*гугл-диск: <a href="https://drive.google.com/drive/folders/1EOgSQSXfw-3nGcxLsgnHXk5cGLWXVa6e">https://drive.google.com/drive/folders/1EOgSQSXfw-3nGcxLsgnHXk5cGLWXVa6e</a>?usp=sharing

#### Содержание проекта:

- Файлы скрипты:
  - 1) BoosterPack.ts
  - 2) Collection.ts
  - 3) Items.ts
  - 4) ItemSet32.ts
  - 5) StartProgramm.ts
  - 6) UICard.ts
  - 7) UICollection.ts
- Тестовое задание.pdf codepжит описание полученного тестового задания.
- Папка compile *coдержит рабочий прототип программы* (web-desktop.rar).

#### Решение:

#### Подготовка

Для успешного выполнения индивидуального задания были описаны и унифицированы основные типы данных:

- Collection коллекция предметов;
- ItemSet32 базовая коллекция предметов;
- BoosterPack интерфейс, реализующий выборку пяти карт из базовой коллекции предметов;
- UICard (опционально) сущность реализующая графическое отображение предмета;
- UICollection (опционально) сущность реализующая графическое отображение коллекции предметов;

#### Реализация сущности Item (предмет)

Параметры 'тип предмета' и 'редкость предмета' заданы соответствующими перечислениями: Туре и Rarity

```
//Тип предмета: ОРУЖИЕ, ШЛЕМ, ДОСПЕХ, ЩИТ export enum Type { BOOSTERPACK = 0, WEAPON = 1, HELMET , ARMOR , SHIELD };
```

<sup>\*</sup>Прототип задачи выполнен с помощью CocosCreator: https://www.cocos.com/en/products

```
//Редкость предмета: ОБЫЧНЫЙ, НЕОБЫЧНЫЙ, РЕДКИЙ, ЛЕГЕНДАРНЫЙ export enum Rarity {COMMON = 2, UNCOMMON, RARE, LEGENDARY}; // 2 ... 5 звёзд. (\phi a \check{u} \pi): Items.ts cmpo \kappa u: 6-10)
```

### Описан базовый абстрактный класс 'Iltem'

```
//Базовый класс для предметов
    export abstract class IItem {
        constructor(name: string, type: Type, rarity: Rarity = Rarity.COMMON){
            this._name = name;
            this. type = type;
            this._rarity = rarity;
      protected _name: string = 'emptyItem';
      protected readonly _rarity?: Rarity;
      protected readonly _type: Type;
      public set Name(name: string) { this._name = name; };
      public get Name(): string { return this. name; };
      public get Rarity(): Rarity { return this._rarity};
      public get Type(): Type { return this._type};
       //Получить string с описанием значений всех существующих полей класса.
      public DebugInfo(): string {
           return "[TypeItem: " + Type[this._type] + "] " + " (name)"+this._name + " -
RarityItem(" + Rarity[this._rarity] + ")";
```

(файл: Items.ts *строки*: 12-35)

с тремя свойствами: 'имя', 'тип', 'редкость'.

Созданы производные классы, устанавливающие строгий тип объекта, в момент его создания.

```
//Оружие
export class Weapon extends IItem {
    constructor(name: string, rarity?: Rarity){
        super(name, Type. WEAPON, rarity);
    }
}

//Шлем
export class Helmet extends IItem {
    constructor(name: string, rarity?: Rarity){
        super(name, Type. HELMET, rarity);
    }
}

//Доспех
```

```
export class Armor extends IItem {
    constructor(name: string, rarity?: Rarity){
        super(name, Type. ARMOR, rarity);
    }
}

//Щит

export class Shield extends IItem {
    constructor(name: string, rarity?: Rarity){
        super(name, Type. SHIELD, rarity);
    }
}
```

(файл: Items.ts *строки*: 38-64)

# Реализация сущности Collection (коллекция)

Каждый объект коллекции содержит в себе массив предметов и реализацию методов в зависимости от дочернего типа.

```
//Базовый класс для коллекции предметов
export abstract class ICollection{

protected _collection: Array<Item> = [];

public get List(): Array<Item> {return this._collection};

//Удалить предмет из коллекции
public RemoveItem(item: Item){/*<-----!прописать_логику!----->*/}

//Показать коллекцию в консоле
protected abstract DebugInfo(): void;

//Добавить предмет в коллекцию
protected abstract AddItem(item: Item): void;

//Метод для графического отображения елементов (содержимое будет зависить от используемого движка или фреймворка)
protected abstract Show(): void;

}

(файл: Collection.ts строки: 6-24)
```

На основе базового класса созданы производные:

Collection – используется для временного хранения получаемого лута и содержит метод Shuffle, позволяющий перемешать её элементы.

\*коллекция содержит ссылки на объект.

```
//Коллекция, тип предметов: Item.IItem
export class Collection extends ICollection{

//Добавить единицу предмета в инвентарь.
public AddItem(item: Item){
    this._collection.push(item);
```

```
//Метод для графического отображения елементов (содержимое будет зависить от испол
ьзуемого движка или фреймворка)
        public Show(): void{
                                };
        //Показать коллекцию в консоле
        public DebugInfo(): void {
           console.log('Collection DebugInfo:');
           this._collection.forEach(item => {
               console.log(item.DebugInfo());
           });
        //Перемешать элементы массива
        public Shuffle(){
            console.log('Shuffle element collection')
            let tmp;
            for(let i = this._collection.length - 1; i > 0;){
                let rnd = Math.floor((Math.random() * 10) % this._collection.length)
                if(rnd < this._collection.length && rnd >= 0){
                    tmp = Object.assign(this._collection[i])
                    this. collection[i] = Object.assign(this. collection[rnd])
                    this._collection[rnd] = Object.assign(tmp)
                    --i;
                }
```

 $(\phi a \ddot{u}\pi$ : Collection.ts *строки*: 28-64)

Inventory –инструмент для хранения полученных предметов, присутствуют методы: AddItem – для добавления единицы предмета и addCollection, для добавления списка предметов.

\*полученные объекты копируются по значению в инвентарь.

 $(\phi a \ddot{u} \pi : Collection.ts$ *строки*: 66-95)

## Реализация сущности BoosterPack (бустерпак)

IBoosterPack является базовым классом, содержит в себе ссылку на коллекцию предметов, переопределяемый метод открытия объекта Open и свойство определяющее редкость бустерпака, соответствующее ранее принятому аргументу. \*объект класса приводится к предмету

```
//Базовый класс для бустерпаков
export abstract class IBoosterPack extends rpgItem.IItem {
    constructor(rarity: rpgItem.Rarity, collection: rpgCollection.Collection){

    if(rarity == rpgItem.Rarity.COMMON)
        rarity = rpgItem.Rarity.UNCOMMON;

    let name: string = "[" + rpgItem.Rarity[rarity] + "] BOOSTERPACK";
        super(name,rpgItem.Type.BOOSTERPACK,rarity);

    this._rarity = rarity;
    this._collection = collection;//<---содержит входящую коллекцию предметов.
}

protected readonly _rarity: rpgItem.Rarity;
    protected readonly _collection: rpgCollection.Collection;

public abstract Open(): rpgCollection.Collection;
}</pre>
```

 $(\phi a \ddot{u} \pi: BoosterPack.ts$ *строки: 8-25*)

# Задание №1:

✓ Скрипт должен содержать базу из 32 предметов (по 2 каждой редкости, каждого типа)

Для реализация данного пункта задания создан класс ItemSet32.

Единственный метод класса создаёт и возвращает коллекцию 32-ух предметов.

```
createItemSet(): rpgCollection.Collection{
    let collection: rpgCollection.Collection = new rpgCollection.Collection();

let names: string[] = ["type1", "type2"];
    let countCreation: number = 0; //as rarity item

for(countCreation: countCreation<4; countCreation++){
    for(let i:number = 0; i < 2; ++i){
        collection.AddItem(new rpgItem.Armor(names[i],countCreation+2));
    }
    for(let i:number = 0; i < 2; ++i){
        collection.AddItem(new rpgItem.Helmet(names[i],countCreation+2));
    }
    for(let i:number = 0; i < 2; ++i){
        collection.AddItem(new rpgItem.Shield(names[i],countCreation+2));
    }
    for(let i:number = 0; i < 2; ++i){
        collection.AddItem(new rpgItem.Weapon(names[i],countCreation+2));
    }
}
return collection;
}</pre>
```

 $(\phi a \ddot{u} \pi: \text{ItemSet32.ts} \ cmpo \kappa u: 5-29)$ 

- ✓ Скрипт должен содержать функцию, реализующую открытие бустерпака
- ✓ Параметром функции является редкость бустерпака

✓ Результатом должны быть 5 предметов, выдаваемых этим бустерпаком

Метод Open, класса BoosterPack, позволяет получить 5 предметов из входящей коллекции: 2 предмета соответствующие редкости бустерпака и 3 предмета с редкостью ниже на единицу.

\*данный вид бустерпака не контролирует количество повторяющихся предметов

```
//Стандартный бустерпак: 5 любых предметов.
export class BoosterPack extends IBoosterPack{

Open(): rpgCollection.Collection{
    let list: rpgCollection.Collection = new rpgCollection.Collection();

    let tmp: number = 0;
    while(tmp < 2)
    {
```

<sup>\*</sup>параметр перенесён в конструктор класса IBoosterPack

```
let rnd:number = Math.floor(Math.random() * 100) % this._collection.Li
st.length;
                         if(this. collection.List[rnd].Rarity == this. rarity){
                             list.AddItem(this._collection.List[rnd]);
                             tmp++;
                         }
                tmp = 0;
                while(tmp < 3)</pre>
                    let rnd:number = Math.floor(Math.random() * 100) % this._collection.Li
st.length;
                        if(this._collection.List[rnd].Rarity == this._rarity - 1){
                             list.AddItem(this. collection.List[rnd]);
                             tmp++;
                        }
                console.log('Бустерпак открыт' + list.List);
                list.Shuffle();
            return list;
        }
```

 $(\phi a \ddot{u} \pi: BoosterPack.ts$ *строки: 29-61)* 

# Задание №2:

✓ Игрокам не нравится, когда из бустерпака падает много однотипных предметов (например четыре щита). Необходимо ввести новый вид бустерпака (consistent pack), в котором не выдается более чем два предмета одинакового типа

```
//Последовательный бустерпак: в наборе из 5-
ти вещей, выпадет неболее двух одинаковых пар идентичного типа.
  export class ConsistentBoosterPack extends IBoosterPack{

    Open(): rpgCollection.Collection{
        let list: rpgCollection.Collection = new rpgCollection.Collection();
        let tmp = 0;
        let n = 1;

        while(tmp < 1)
        {
            let rnd:number = Math.floor(Math.random() * 100) % this._collection.List.length;

        if(this._collection.List[rnd].Rarity == this._rarity-1){
            list.AddItem(this._collection.List[rnd]);
            tmp++;
```

```
tmp = 0;
            while(tmp < 2)
                let rnd:number = Math.floor(Math.random() * 100) % this._collection.List.l
ength;
                    if(this._collection.List[rnd].Rarity == this._rarity){
                        if(this._collection.List[rnd].Type != list.List[n-1].Type){
                        list.AddItem(this._collection.List[rnd]);
                        tmp++;
                        n++;
            }
            tmp = 0;
            while(tmp < 2)
                let rnd:number = Math.floor(Math.random() * 100) % this._collection.List.1
ength;
                    if(this._collection.List[rnd].Rarity == this._rarity-1){
                        if(this._collection.List[rnd].Type != list.List[n-
2].Type && this._collection.List[rnd].Type != list.List[n-1].Type){
                        list.AddItem(this._collection.List[rnd]);
                        tmp++;
                        n++;
                    }
            }
            console.log('Бустерпак открыт' + list.List);
            list.Shuffle();
            return list;
```

 $(\phi a \ddot{u} \pi: BoosterPack.ts$ *строки: 63-115*)

# Задание №3:

✓ Для гарантии сбора коллекции нужно завести новый вид бустерпака (fair\_pack) такой, чтобы игрок открыв максимум 24 таких бустерпака редкости X получил все предметы редкости X.

```
//Справидливый бустерпак: гарантирует получение всех предметов редкости X, при открытии н
е более 24 бустерпаков.
export class FairBoosterPack extends IBoosterPack{
static uncommon_q: number = 0;
```

```
static rare q: number = 0;
static legendary_q: number = 0;
constructor(rarity: rpgItem.Rarity, collection: rpgCollection.Collection){
    super(rarity, collection);
    switch (rarity) {
        case 3:
                FairBoosterPack.uncommon_q += 1;
            break;
        case 4:
                FairBoosterPack.rare_q += 1;
        case 5:
                FairBoosterPack.legendary_q += 1;
        default:
                FairBoosterPack.uncommon_q += 1;
            break;
    console.log('uncommon: ' + FairBoosterPack.uncommon_q);
    console.log('rare: ' + FairBoosterPack.rare_q);
    console.log('legendary: ' + FairBoosterPack.legendary_q);
Open(): rpgCollection.Collection{
    let list: rpgCollection.Collection = new rpgCollection.Collection();
    let tmp = 0;
   let n = 1;
    let x = 0; //<- для числа открытых бустер-паков
    let max = 24;
    switch (this._rarity) {
        case 3:
            x = FairBoosterPack.uncommon_q;
            break;
        case 4:
            x = FairBoosterPack.rare q;
            break;
        case 5:
            x = FairBoosterPack.legendary_q
            break:
        default:
            x = FairBoosterPack.uncommon_q;
            break;
    }
    if(x == max){
        switch (this._rarity) {
            case 3:
                    FairBoosterPack.uncommon_q = 1;
                    x = FairBoosterPack.uncommon_q;
```

```
break;
                    case 4:
                             FairBoosterPack.rare q = 1;
                            x = FairBoosterPack.rare q;
                        break:
                    case 5:
                             FairBoosterPack.legendary_q = 1;
                            x = FairBoosterPack.legendary_q
                        break;
                    default:
                             FairBoosterPack.uncommon_q = 1;
                            x = FairBoosterPack.uncommon_q;
                        break;
                }
                let k = Math.round((1/(max-x)) * 1000);
                console.log(k);
                if(k \le 55)
                    list.AddItem(new rpgItem.Shield('uni_shield',this._rarity));
                if(k >= 56 \&\& k <= 89)
                    list.AddItem(new rpgItem.Weapon('uni weapon',this. rarity));
                if(k >= 90 \&\& k <= 199)
                    list.AddItem(new rpgItem.Armor('uni_armor',this._rarity));
                if(k \ge 200 \&\& k < 1000 || k = 1000)
                    list.AddItem(new rpgItem.Helmet('uni_helmet',this._rarity));
            while(tmp < 1)</pre>
                let rnd:number = Math.floor(Math.random() * 100) % this._collection.List.l
ength;
                    if(this._collection.List[rnd].Rarity == this._rarity){
                         if(this._collection.List[rnd].Type != list.List[n-1].Type){
                        list.AddItem(this._collection.List[rnd]);
                        tmp++;
                        n++;
                         }
                    }
            }
            tmp = 0;
            while(tmp < 3)
                let rnd:number = Math.floor(Math.random() * 100) % this._collection.List.l
ength;
                    if(this._collection.List[rnd].Rarity == this._rarity-1){
                        if(this._collection.List[rnd].Type != list.List[n-
2].Type && this._collection.List[rnd].Type != list.List[n-1].Type){
                        list.AddItem(this._collection.List[rnd]);
                        tmp++;
```

```
n++;
}
}
console.log('Бустерпак открыт' + list.List);
list.Shuffle();
return list;
}
```

 $(\phi a \ddot{u} \pi: BoosterPack.ts \it cmpoкu: 117-234)$ 

 ✓ Реализовать инвентарь игрока, где будут содержаться все полученные им предметы

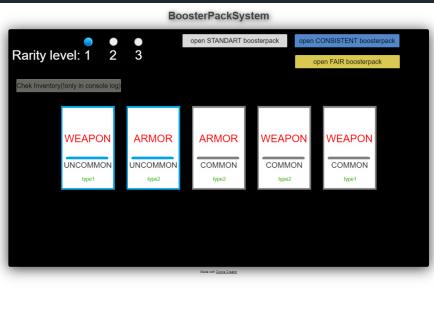
\*Нажмите для ознакомления с реализацией инвентаря (см.Inventory)

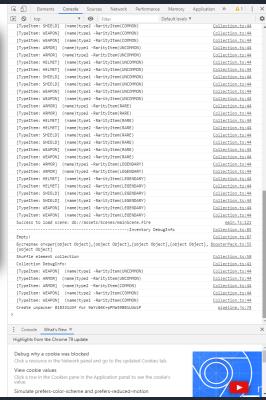
✓ Реализовать функцию выдачи N бустерпаков вида X (N, X - параметры)

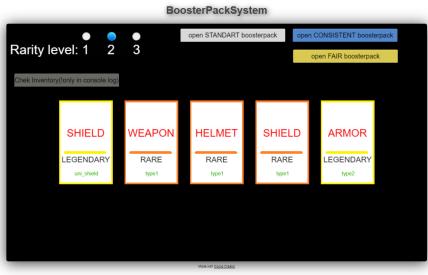
Количество бустерпаков равное значению переменной quantity будет добавлено в инвентарь.

```
getBoosterPack(quantity: number, type: BoosterPack.Type){
      let boosterPack: BoosterPack.IBoosterPack;
      switch (type) {
        case BoosterPack.Type.Standart:
          boosterPack = new BoosterPack.BoosterPack(rpgItem.Rarity.COMMON,this.itemList);
        case BoosterPack.Type.Consistent:
          boosterPack = new BoosterPack.ConsistentBoosterPack(rpgItem.Rarity.COMMON,this.i
temList);
          break;
        case BoosterPack.Type.Fair:
          boosterPack = new BoosterPack.FairBoosterPack(rpgItem.Rarity.COMMON,this.itemLis
t);
            break:
        default:
          boosterPack = new BoosterPack.BoosterPack(rpgItem.Rarity.COMMON,this.itemList);
          break;
      }
      for(let i = 0; i < quantity; ++i){</pre>
          this._inventory.AddItem(Object.assign(boosterPack));
      }
```

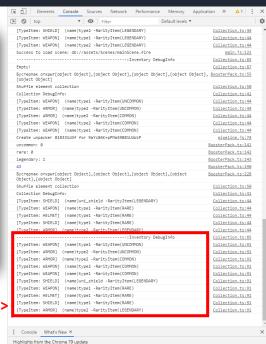
### Работа прототипа:







СОДЕРЖИМОЕ ИНВЕНТАРЯ --->



a nel and go to the updated Cook

View cookie values
Click a row in the Cookies pane in the Application panel to see the cookie's