CATEDRA DE INFORMATICĂ I

ИНДИВИДУАЛЬНАЯ РАБОТА №3

Дисциплина :

JavaScript

Студент : BZOVII VALENTIN

Группа: R-2222L

Профессор: Nichita Nartea

Кишинев , 2025

Изложенная задача :

Цель работы

Создайте консольное приложение, моделирующее систему инвентаря, где можно добавлять предметы, изменять их свойства и управлять ими.

Шаг 1. Создание класса Item

Создайте класс Item, который будет представлять предмет в инвентаре.

- Поля класса:
  - name – название предмета.
  - weight – вес предмета.
  - rarity – редкость предмета (common, uncommon, rare, legendary).
- Методы:
  - getInfo() – возвращает строку с информацией о предмете.
  - setWeight(newWeight) – изменяет вес предмета.

Пример использования:

```
const sword = new Item("Steel Sword", 3.5, "rare");
console.log(sword.getInfo());
sword.setWeight(4.0);
```

Шаг 2. Создание класса Weapon

Создайте класс Weapon, который расширяет Item.

- Дополнительные поля:
  - damage – урон оружия.
  - durability – прочность (от 0 до 100).
- Методы:
  - use() – уменьшает durability на 10 (если durability > 0).
  - repair() – восстанавливает durability до 100.

Пример использования:

```
const bow = new Weapon("Longbow", 2.0, "uncommon", 15, 100);
console.log(bow.getInfo());
bow.use();
console.log(bow.durability); // должно уменьшиться
bow.repair();
```

Шаг 3. Тестирование

1. Создайте несколько объектов классов Item и Weapon.
2. Вызовите их методы, чтобы убедиться в правильности работы.

Практическая часть

Объектно-ориентированное программирование:

- Есть базовый класс Item, представляющий обычный предмет с полями: название, вес и редкость. Класс Weapon наследует Item и добавляет свойства урона (damage) и прочности (durability), а также методы использования (use) и ремонта (repair) оружия.
- Интерактивное консольное меню: Реализовано через модуль readline для взаимодействия с пользователем через командную строку.
- Меню с пунктами добавления предметов и оружия, просмотра инвентаря, использования и ремонта оружия, а также выхода из программы.
- Управление инвентарём: Все предметы и оружие хранятся в массиве inventory.
- Отображение инвентаря выводит список всех объектов с их описанием. Обработка логики использования и ремонта оружия: Метод use снижает прочность оружия на 10, пока она больше 0, иначе сообщает, что оружие сломано.
- Метод repair восстанавливает прочность до 100%. При выборе действия пользователь вводит номер предмета из списка, проверяется тип объекта и выполняется действие только если это Weapon.
- Асинхронность и вложенные вопросы: Для последовательного ввода данных используется цепочка колбеков rl.question, что обеспечивает пошаговое взаимодействие. Простота и расширяемость: Код легко дополняется новыми типами предметов за счёт наследования Модульность функций позволяет легко изменять и поддерживать код. В итоге, реализация демонстрирует базовые принципы ООП и работу с пользовательским вводом в консоли для управления игровым инвентарём.

Вывод:

Данная программа представляет собой простую и наглядную консольную систему управления инвентарём, основанную на принципах объектно-ориентированного программирования. В её основе лежат два класса: базовый класс для обычных предметов и расширенный класс для оружия, что обеспечивает логичную и удобную структуру данных. Через интерактивное меню пользователь может добавлять предметы и оружие, просматривать содержимое инвентаря, а также использовать и ремонтировать оружие, при этом программа корректно обрабатывает ввод и обеспечивает проверку типов объектов перед выполнением действий.

👉 **GitHub: Valentin Bzovii (JavaScript)**

Ответы на контрольные вопросы

1. Методы массивов для обработки объектов в JavaScript

1. Какое значение имеет this в методах класса?

В методах класса this ссылается на конкретный экземпляр объекта, вызвавший этот метод. То есть внутри метода this указывает на тот объект, для которого был вызван метод, позволяя обращаться к его свойствам и другим методам. Это поведение аналогично методам в обычных объектах JavaScript. В классах this определяется в момент вызова метода и обычно указывает на экземпляр класса.

2. Как работает модификатор доступа # в JavaScript?

Модификатор # в JavaScript используется для объявления приватных полей и методов класса. Такие поля и методы доступны только внутри самого класса и недоступны из внешнего кода или у экземпляров класса.

3. В чем разница между классами и функциями-конструкторами?

Классы в JavaScript позволяет создавать объекты и наследовать их более понятным и структурированным способом. Функции-конструкторы — более старый способ создания объектов с помощью обычных функций, которые вызываются с new и манипулируют свойством this для инициализации. Главные отличия: Синтаксис: классы имеют более читаемый и декларативный синтаксис, а функции-конструкторы — более низкоуровневый и менее наглядный.