Лабораториска вежба 3 - Работа со објекти во JS

1 Работа со објекти

1.1 Задача

Да се развие објект наречен `NPC`. За секој `NPC` се чува име и неговото здравје (`hitpoints`). Името на секој NPC објект претставува број, кој го претставува редниот број на инстанцираниот `NPC`.

Да се развие објект наречен `Hero`, кој го проширува објектот `NPC`. За секој `Hero` објект потребно е да се чува име на карактерот (како `String`), неговото здравје (`hitpoints`, предефинирано од 0-100) и колку штета прави неговиот напад (`damage`). При креирање на нов `Hero`, името задолжително се задава како параметар, додека додека не е дополнително наведено, секој карактер има `100` `hitpoints` и прави штета од `10` поени

За објектите `Hero` потребно е да се креира метода `attack`, која напаѓа друг објект од типот `NPC` или `Hero`. Дополнително за секој карактер да се креира метода `status` која го печати името на карактерот и неговите моментални hitpoints.

Во поле треба да крерирате (произволно) `N` објекти од типот `Hero` и `NPC` (каде `N` е произволен број помеѓу `10` и `50`. Во еден круг, еден `Hero` објект може да нанесе 1 удар на произволен противник. Еден карактер (`NPC` или `Hero`) умира ако неговиот `hitpoint` е 0. Играта завршува кога останува само 1 жив `Hero`, кој се печати на конзола со порака `Winner: name, hitpoints!`. Првенствено, **еден** од хероите има `critical` својство кое му овозможува да нападне со `150%` од штетата која може да ја направи (пр. ако прави штета од 10, со ова својство ќе прави штета од 15). Доколку тоа својство го искористи врз `NPC` објект, го задржува истото во следниот круг, додека ако нападне друг `Hero` објект, и нападнатиот `Hero` објект преживее, му го предава на нападнатиот херој, за тој да може да го искористи во следниот круг. Доколку со `critical` нападот го убие `Hero` објектот, тогаш својството се губи и не се користи од ниту еден `Hero` објект понатаму.

1.2 Задача

`Singleton` претставува објект од кој може да има максимум една инстанца (или да нема ниту една инстанца). За да се постигне тоа, потребно е да се креира посебна метода/функција (пр. `getInstance`), која **ќе го креира објектот (доколку претходно не постоел) или го враќа моментално креираниот објект доколку претходно бил креиран**. Притоа, програмата не смее да има пристап до деловите од Singleton-от кои се одговорни за неговото инстанцирање (односно треба да има пристап само до методот `getInstance` (тука се подразбира дека треба да има пристап до останатите стандардни методи кои доаѓаат предефинирано).

Ваша задача е да напишете JavaScript код кој ќе овозможи да се креира Singleton објект кој во себе ќе го содржи времето кога објектот е креиран (односно вредноста `Date.now()`). Притоа доколку некој проба да креира нов објект, треба да се врати веќе креираниот.

Во `zad2/zad2.js` е даден код кој треба да се проработи според претходно наведеното. Пример излез Ви е даден во `example_output.txt`.

1.3 Задача

Во JavaScript постои можност да се додефинираат методи на веќе постоечки (па дури и вградени) објекти. Таа техника се нарекува polyfill.

Познато е дека елементите во дадено поле во JavaScript не мора да се со последователни индекси. Тоа значи дека може помеѓу првиот (нултиот) и последниот елемент да има недефинирани елементи. Да се напише (прошири) дефиницијата на Array во JavaScript, со што ќе се воведе метод fillDefault кој ќе ги пополни сите недефинирани елементи од полето (со индекси поголеми од нула а помали од крајниот индексот) на дадена вредност.

Пример:

```
var x = [1, 2, 4];
x[10] = 100;
x[8] = 8;
console.log(x.toString()); // Prints 1,2,4,,,,,,8,,100
x.fillDefault(0);
console.log(x.toString()); // Prinst 1,2,4,0,0,0,0,0,8,0,100
```