# NETHACK PROJECT JS

## ABSTRACT CLASSES

<https://stackoverflow.com/questions/29480569/does-ecmascript-6-have-a-convention-for-abstract-classes>

## DOCUMENT.WRITE()

- Write JS to the html document

<https://www.w3schools.com/jsref/met_doc_write.asp>

## CLASS

- Function associated with class is a method

- Object created from class is an instance

### PRIVATE VARIABLES

- Use ‘\_’ on the variable

- Then place method in the constructor to access

- Access with obj.getSymbol()

**class** Dog **extends** Tile{

**constructor**(symbol){

super();

**let** \_symbol = symbol;

this.getSymbol=()**=>**{

return \_symbol;

}

}

}

### INHERITENCE

- Avoids duplication of code

- Data and methods are passed from Tile to Dog

**class** Tile{

**constructor**(symbol,isPassable){

this.symbol = symbol; *// This will represent the tile*

this.isPassable = isPassable; *// This will determin if can pass this tile*

this.getSymbol = () **=>** {

*// Fetch symbol to be displayed*

return this.symbol;

}

if (new.target === Tile) {

*// Will act like abstract class, where cant make class from this*

throw new TypeError("Cannot construct Abstract instances directly");

}

}

}

**class** Dog **extends** Tile{

**constructor**(symbol,isPassable){

super(symbol, isPassable); *// Inherits from tile class (superclass)*

}

}

#### IS-A AND LIKE-A

<https://en.wikipedia.org/wiki/Liskov_substitution_principle>

- Inheritance should always model an IS-A and WORKS-LIKE-A relationship. That is, a manager “is a” and “works like a” specific kind of employee,

#### HAS-A

- If inheritance models the IS-A relationship, then composition models the HAS-A relationship

- Where a class can contain a type of another class, but isnt a is-a or a like-a relationship

## IMPORT ONE CLASS TO ANOTHER USING NODE.JS

- Need to export the class

module.exports = **class** Tile{}

- Then to require the class where you want to use it

**var** Tile = require('./Tile.js');