2019 | PHP DEVELOPER EVALUATION

Thank you for your interest in joining TrackTik!
Our developers have concocted a test to see your PHP coding skills in action.
There is no time limit to complete the test but please make sure you demonstrate the best practices you usually use. Once you're done, submit the test via the provided link and let us know how long it took you to complete it.
Fingers crossed!

Question 1: Using the code given, create each type of electronic as classes. Every **ElectronicItem** must have a function called **maxExtras** that limits the number of extras an electronic item can have. The extras are a list of electronic items that are attached to another electronic item to complement it.

- The console can have a maximum of 4 extras
- The television has no maximum extras
- The microwave can't have any extras
- The controller can't have any extras

Create a scenario where a person would buy:

1 console, 2 televisions with different prices and 1 microwave

The console and televisions have extras; those extras are controllers. The console has 2 remote controllers and 2 wired controllers. The TV #1 has 2 remote controllers and the TV #2 has 1 remote controller.

Sort the items by price and output the total pricing.

Question 2: That person's friend saw her with her new purchase and asked her how much the console and its controllers had cost her. Give the answer.

Please return the test in a compressed PHP file or through an online Git repository (GitHub or similar).

You will be evaluated by several TrackTik developers on the following aspects:
(scale of 1 to 10)

- Correct Output of Code and Bug Free
- Code Clarity and Simplicity
- Code Structure
- Application of Object-Oriented Concepts
- Technical Level of Solution vs your level of expertise

Code:

```php
<?php

class ElectronicItems
    {

    private $items = array();

    public function __construct(array $items)
        {

        $this->items = $items;
        }

    /**
     * Returns the items depending on the sorting type requested
     *
     * @return array
     */
    public function getSortedItems($type)
        {

        $sorted = array();
        foreach ($this->items as $item)
            {

            $sorted[($item->price * 100)] = $item;
            }

        return ksort($sorted, SORT_NUMERIC);
        }

    /**
     *
     * @param string $type
     * @return array
     */
    public function getItemsByType($type)
        {

        if (in_array($type, ElectronicItem::$types))
            {

            $callback = function($item) use ($type)
                {

                return $item->type == $type;
                };

            $items = array_filter($this->items, $callback);
            }

        return false;
        }
    }
```

```
class ElectronicItem
    {

    /**
     * @var float
     */
    public $price;

    /**
     * @var string
     */
    private $type;
    public $wired;

    const ELECTRONIC_ITEM_TELEVISION = 'television';
    const ELECTRONIC_ITEM_CONSOLE = 'console';
    const ELECTRONIC_ITEM_MICROWAVE = 'microwave';

    private static $types = array(self::ELECTRONIC_ITEM_CONSOLE,
              self::ELECTRONIC_ITEM_MICROWAVE, self::ELECTRONIC_ITEM_TELEVISION);

    function getPrice()
        {
        return $this->price;
        }

    function getType()
        {
        return $this->type;
        }

    function getWired()
        {
        return $this->wired;
        }

    function setPrice($price)
        {
        $this->price = $price;
        }

    function setType($type)
        {
        $this->type = $type;
        }

    function setWired($wired)
        {
        $this->wired = $wired;
        }
    }
```