```
<?php
/**
 * This class can be used to convert a BBAN to IBAN
 * For the conversion the bban2iban function should be used
 * That function will take the BBAN as it's only paramater and will return the IBAN if possible
 */
class bban {

    const ACCOUNT_LENGTH = 14;

    /**
     * Array of valid bankcodes
     * @var array
     */
    private $bankCodes = array(
        16 => 16,
        10 => 10,
        33 => 33,
        42 => 42,
        22 => 22,
        11 => 22,
        55 => 55,
        93 => '00',
        17 => 17,
        12 => 12,
        96 => 96,
```

```
    83 => 83,
    77 => 77,
    51 => 51,
    75 => 75);

/**
 * List of valid bank specific BBAN lengths
 * @var array
 */
private $lengths = array(
    16 => array(9, 10, 12),
    10 => array(9, 14),
    33 => array(12),
    42 => array(13),
    22 => array(6, 12),
    11 => array(10),
    55 => array(9),
    93 => array(10),
    17 => array(8, 11),
    12 => array(10),
    96 => array(13),
    83 => array(10),
    77 => array(12),
    51 => array(10),
    75 => array(14));
private $digits = array('0', '1', '2', '3', '4', '5', '6', '7', '8', '9',
    'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J',
```

```
    'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T',
    'U', 'V', 'W', 'X', 'Y', 'Z');

/**
 * Main function that converts BBAN to IBAN
 * @param string $bban
 * @return mixed BBAN if the the conversion was successful, false otherwise
 */
public function bban2iban($bban) {
        // Check for valid symbols
        if (!$this->isBbanValid($bban)) {
                return FALSE;
        }

        // Extract bank code
        $bankCode = $this->getBankCodeFromBban($bban);
        if (FALSE === $bankCode) {
                return FALSE;
        }

        // Pad to required length
        $this->padBbanLength($bban);
        return $this->getIbanFromBban($bankCode, $bban);
}

/**
 * Checks if the BBAN is valid
```

```
     * @param string $bban
     * @return boolean True if BBAN is valid, false otherwise
     */
    private function isBbanValid($bban) {

        $bban = str_replace(' ', '', $bban);
        // check if its a number
        if (!is_numeric($bban)) {
            return FALSE;
        }
        // check for length
        if (4 > strlen($bban) || 14 < strlen($bban)) {
            return FALSE;
        }
        // identify which banks code it is and check for that banks code length
        $bankCode = substr($bban, 0, 2);
        if (is_array($this->lengths[$bankCode]) && !in_array(strlen($bban), $this->lengths[$bankCode]))
            return FALSE;

        // validate checksum
        return $this->is731ChecksumValid($bban);
    }

    /**
     * Validate code checksum
     * @param string $bban
```

```
     * @return boolean True if checksum is valid, false otherwise
     */
    private function is731ChecksumValid($bban) {
        $sum = 0;
        $parts = array(7, 3, 1, 7, 3, 1, 7, 3, 1, 7, 3, 1, 7, 3, 1);
        $accountNr = substr($bban, 0, strlen($bban) - 1);
        $z = 0;
        for ($i = (strlen($accountNr) - 1); $i > -1; $i--) {
            $sum += $accountNr[$i] * $parts[$z];
            $z++;
        }
        if (ceil($sum / 10) * 10 - $sum != substr($bban, -1)) {
            return FALSE;
        }
        return TRUE;
    }

    /**
     * Returns IBAN bank code from BBAN account number
     * @param string $bban
     * @return mixed 2 digit bank code if it exists, false if it doesn't
     */
    private function getBankCodeFromBban($bban) {
        $bankCode = substr($bban, 0, 2);
        if (!array_key_exists($bankCode, $this->bankCodes)) {
            return FALSE;
        }
```

```
        return $this->bankCodes[$bankCode];
    }

    /**
     * Pads BBAN number to required length with zeros
     * @param type $bban
     */
    private function padBbanLength(&$bban) {
        if (self::ACCOUNT_LENGTH > strlen($bban)) {
            $bban = str_pad($bban, self::ACCOUNT_LENGTH, '0', STR_PAD_LEFT);
        }
    }

    /**
     * Returns a valid IBAN from BBAN
     * @param type $bankCode 2 digit bank code
     * @param type $bban
     * @return string valid BBAN code
     */
    private function getIbanFromBban($bankCode, $bban) {
        $hash = 'EE00' . $bankCode . $bban;

        // Move first 4 symbols to the end
        $part = substr($hash, 0, 4);
        $hash = substr($hash, -(strlen($hash) - 4)) . $part;

        $this->convertDigits($hash);
```

```
        $checkDigits = $this->applyMod9710($hash);
        return 'EE' . $checkDigits . $bankCode . $bban;
    }

    private function applyMod9710($s) {
        $result = 98 - (int) bcmod($s, 97);
        if (2 > strlen($result)) {
            $result = str_pad($result, 2, '0', STR_PAD_LEFT);
        }
        return $result;
    }

    /**
     * Convert the alphabetical characters into numbers.
     * @param string
     */
    private function convertDigits(&$hash) {
        $newHash = '';
        for ($i = 0; $i < strlen($hash); $i++) {
            $symbol = $hash[$i];
            if (!is_numeric($symbol)) {
                $symbol = array_search($symbol, $this->digits);
            }
            $newHash .= $symbol;
        }
        $hash = $newHash;
    }
```

```
}


?>
```