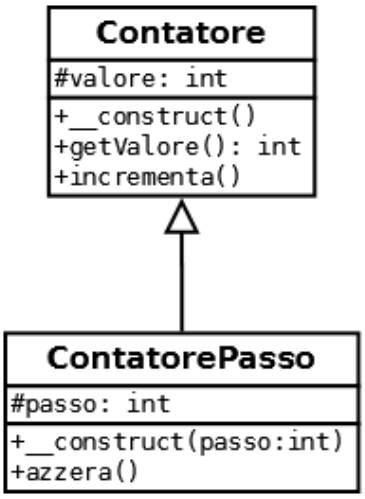
Q1 (3 punti)

Si chiede di sviluppare i metodi costruttore, getValore() e incrementa() di cui è fornita la firma:

```
class Contatore {
  /** @var Int il valore del conteggio
  protected $valore;
  /** set a 0 (zero) il valore del conteggio */
  function __construct(){...}
  /** @return Int il valore di conteggio */
  function getValore() {...}
  /** aggiunge 1 (uno) al valore di conteggio */
  function incrementa() {...}
}
```



Q2 (3 punti)

Si chiede di sviluppare la classe ContatorePasso indicata nel diagramma in cui:

- il costruttore ha un parametro con il passo di incremento da assegnare all'attributo;
- il metodo azzera() porta a zero l'attributo valore;
- il metodo incrementa() è ridefinito per aggiungere all'attributo valore tante unità quanto è il valore di passo.

Q3 (3 punti)

Illustra le principali proprietà degli array in php.

Q4 (3 punti)

Si esamini la descrizione della funzione array_key_exists tratta dal manuale PHP e si consideri poi la classe:

```
class Spese {
 /** @var Float le diverse voci di spesa, es array('gas'=>134.23,'telefono'=>421.35) */
 protected $vociDiSpesa;
 function __construct() {
  $this->vociDiSpesa=array();
 }
}
```

Si richiede di aggiungere due metodi :

- il metodo aggiungiSpesa($merce,$costo) : se la chiave $merce è già presente aggiunge il costo, altrimenti aggiunge la nuova chiave e relativo costo
- il metodo totaleSpese() che restituisce la somma totale delle spese

array_key_exists

array_key_exists — Checks if the given key or index exists in the array

Description

bool **array_key_exists** (mixed `$key` , array `$search`)

array_key_exists() returns **`TRUE`** if the given *`key`* is set in the array. *`key`* can be any value possible for an array index.

Parameters

`key` : Value to check.

`search` : An array with keys to check.

Return Values

Returns **`TRUE`** on success or **`FALSE`** on failure.

Example

Example #1 array_key_exists() example

```
<?php
$search_array = array('first' => 1, 'second' => 4);
if (array_key_exists('first', $search_array)) {
  echo "The 'first' element is in the array";
}
?>
```

Esempio di svolgimento

Q1

```
class Contatore {
  /** @var Int il valore del conteggio */
  protected $valore;
  /** set a 0 (zero) il valore del conteggio */
  function __construct(){
    $this->valore=0;
  }
  /** @return Int il valore di conteggio */
  function getValore() {
    return $this->valore;
  }
  /** aggiunge 1 (uno) al valore di conteggio */
  function incrementa() {
    $this->valore++;
  }
}
```

Q2

```
require_once 'Contatore.php';
class ContatorePasso extends Contatore {
  /** @var Int il passo di incremento */
  protected $passo;
  /**
   * assegna $passo al corrispondente attributo
   * @param Int $passo
   */
  function __construct($passo) {
    $this->passo = $passo;
    parent::__construct();
  }
  /** porta a zero l'attributo valore */
  function azzera(){
    $this->valore=0;
  }
  /** aggiunger all'attributo valore tante unità quanto è il valore di passo */
  function incrementa() {
    $this->valore+=$this->passo;
  }
}
```

Q3

Principali proprietà degli arrayi in PHP:

- non hanno una dimensione statica: il numero di elementi nell'array non viene dichiarato per poi rimanere costante per tutto il tempo di esecuzione, ma può essere variabile durante l'esecuzione dello script:
- l'accesso agli elementi è ottenuto tramite una chiave che può essere di tipo intero o string, diversamente da altri linguaggi come Java in cui agli elementi si accede tramite un indice posizionale
- conseguenza della proprietà precedente è che se anche abbiamo chiavi numeriche non è necessario che siano presenti tutti i valori fra i due estremi minimo e massimo
- il tipo di dato contenuto nei diversi elementi può essere eterogeneo, cioè diverso fra un elemento e un altro
- la scansione di tutti gli elementi può essere compiuta utilizzando il costrutto foreach che prevede due forme: scansione dei soli valori contenuti negli elementi oppure scansione delle coppie chiave, valore.

Q4

```
class Spese {
 /** @var Float[] le diverse voci di spesa, es array('gas'=>134.23,'telefono'=>421.35) */
 protected $vociDiSpesa;
 /** inizializza con array vuoto */
 function __construct() {
  $this->vociDiSpesa=array();
 }
 /**
  * aggiunge questa spesa
  * @param type $merce la merce
  * @param type $costo il relativo costo
  */
 function aggiungiSpesa($merce,$costo){
   if (array_key_exists($merce, $this->vociDiSpesa)){
     $this->vociDiSpesa[$merce]+=$costo;
   }
   else {
     $this->vociDiSpesa[$merce]=$costo;
   }
 }
 /**
  * Calcola il totale spese
  * @return Float il totale delle spese
  */
 function totaleSpese(){
   $tot=0;
   foreach ($this->vociDiSpesa as $value) {
     $tot+=$value;
   }
   return $tot;
 }
}
```