Nama : Aulia Husnul Khotimah
NIM : 1900018228
Kelas : E
UAS PRPL

1. **Prinsip Single Responsibility**, yaitu prinsip dalam pemrograman komputer yang menyatakan bahwa setiap modul, kelas maupun fungsi yang ada pada program komputer harus memiliki satu tanggung jawab saja. keuntungan prinsip ini terletak pada cara penyelesaian tugasnya yang hanya terfokus pada tiap – tiap tugas yang diberikan saja. **Prinsip Open/Closed**, yaitu sebuah prinsip pada OOP yang mengharuskan kita untuk dapat merancang sebuah entitas yang dapat dirubah tanpa harus memodifikasi source code program yang sudah ada. Keuntungan yang didapatkan dari prinsip ini adalah program yang telah kita buat dapat kita ulang pemakaiannya tanpa harus memodifikasi ulang.

2.
```php
<?php
class User
{
  public $nama;
  public $email;
  public $dob;

  public function __construct($data)
  {
    $this->nama = $data['nama'];
    $this->email = $data['email'];
    $this->dob = $data['dob'];
  }
}

class UserRequest
{
  protected static $rules = [
    'nama' => 'string',
    'email' => 'string',
    'dob' => 'string'
  ];

  public static function validate($data){
    foreach (static::$rules as $property => $type){
      if (gettype($data[$property]) != $type){
        throw new \Exception("User property {$property} must be of type {$type}" );
      }
    }
  }
}

class Json{
  public static function from ($data){
    return json_encode($data);
  }
}
```

```
class Age{
   public static function now($data){
      $dob = new DateTime($data['dob']);
      $today = new Datetime(date('d.m.y'));
      return [
         'year' => $today->diff($dob)->y,
         'month' => $today->diff($dob)->m,
         'day' => $today->diff($dob)->d,
      ];
   }
}

$data = [
   'nama' => 'AuliaHusnulKhotimah',
   'email' => 'aulia1900018228@webmail.uad.ac.id',
   'dob' => '12.9.2001'
];

UserRequest::validate($data);
$user = new User($data);
print_r(Json::from($user));
echo '<br>';
print_r(Age::now($data));
```

3. a. Jika kita menambahkan sebuah class class baru yng mengimplementasikan interface maka Interface akan membuat si objek bisa menggunakan objek apapun, dengan syarat objek tersebut harus mengimplementasikan method dari interface
Cara memanggil tiap class
1. interface MyInterface
2. public abstract class Entitas
3. public class Mahasiswa extends Entitas implements MyInterface

b. <?php

```
interface Shape{
   public function area();
}
class PersegiPanjang implements Shape{
   public $panjang;
   public $lebar;

   public function area(){
      return $this->panjang * $this->lebar;
   }
}

class Bola implements Shape{
   public $jarijaribola;

   public function area(){
      return 4/3 * (pi() * $this->jarijaribola * $this->jarijaribola * $this->jarijaribola );
```

```
    }
}

class Kerucut implements Shape{
    public $jarijarikerucut;
    public $tinggikerucut;

    public function area(){
        return 1/3 * (pi() * $this->jarijarikerucut * $this->jarijarikerucut * $this->tinggikerucut );
    }
}

class Kubus implements Shape{
    public $sisi;

    public function area(){
        return $this->sisi * $this->sisi * $this->sisi;
    }
}

class Lingkaran implements Shape{
    public $jarijarilingkaran;

    public function area(){
        return 2 * (pi() * $this->jarijarilingkaran);
    }
}

class AreaKalkulator {
    public function kalkulator(Shape $shapes)
    {
        $area = [];

        foreach ($shapes as $shape) {
            $area[] = $shape->area();
        }

        return array_sum($area);
    }
}

?>
```