TUGAS PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK

PERTEMUAN 1

10111002 – AHMAD RAMDANI

SISTEM INFORMASI 2B

Dosen Pengampu

Sari Azhariyah, M.Pd.T.

Usep Abdul Rosid, S.T., M.Kom

DAFTAR ISI

BAB I
ISI

1.1 Sintak Program Pencatatan barang

```
<?php
class Database {
    protected $data = [];
}

class Barang extends Database {
    public function tambahBarang($nama, $jumlah) {
        $this->data[] = ["id"=>count($this->data)+1,"nama"=>$nama,"jumlah"=>$jumlah];
    }
    public function tampilBarang() { return $this->data; }
}

class BarangElektronik extends Barang {
    public function info() { return "Barang Elektronik bergaransi"; }
}

class BarangMakanan extends Barang {
```

```
    public function info() { return "Barang Makanan ada kadaluarsa"; }
}
$barang = new Barang();
$barang->tambahBarang("Laptop",5);
$barang->tambahBarang("Meja",10);
$elektronik = new BarangElektronik();
$makanan = new BarangMakanan();
print_r($barang->tampilBarang());
echo $elektronik->info();
echo $makanan->info();
?>
```

1.2 Analisis Sintak Program Pencatatan Barang

1. Class
   ```
   class Database { ... }
   class Barang extends Database { ... }
   class BarangElektronik extends Barang { ... }
   class BarangMakanan extends Barang { ... }
   ```

   Alasannya karena Class adalah cetak biru (blueprint) untuk membuat object. Database sebagai induk, Barang sebagai class utama, sedangkan BarangElektronik dan BarangMakanan adalah turunan khusus.

2. Object
   ```
   $barang = new Barang();
   $elektronik = new BarangElektronik();
   $makanan = new BarangMakanan();
   ```

   Alasannya karena Object adalah instance nyata dari class. $barang dibuat dari class Barang, $elektronik dari BarangElektronik, dan $makanan dari BarangMakanan.

3. Properties
   ```
   protected $data = [];
   ```

   Alasannya karena Property adalah variabel yang ada di dalam class. $data digunakan untuk menyimpan daftar barang yang dimasukkan.

4. Method

```
public function tambahBarang($nama, $jumlah) { ... }
public function tampilBarang() { ... }
public function info() { ... }
```

Alasannya karena Method adalah fungsi yang ada di dalam class. tambahBarang() untuk menambah data, tampilBarang() untuk menampilkan data, dan info() untuk memberikan keterangan khusus pada masing-masing class turunan.

5. Enkapsulasi

```
protected $data;
```

Alasannya karena Modifier protected membuat properti $data hanya bisa diakses oleh class itu sendiri dan class turunannya, bukan dari luar class. Ini melindungi data agar tidak diubah sembarangan.

6. Abstraksi

```
$barang->tambahBarang("Laptop",5);
$barang->tambahBarang("Meja",10);
$barang->tampilBarang();
```

Alasannya karena Abstraksi menyembunyikan detail implementasi. User cukup memanggil method tambahBarang() dan tampilBarang(), tanpa perlu tahu bagaimana data disimpan (dalam array).

7. Inheritance

```
class Barang extends Database { ... }
class BarangElektronik extends Barang { ... }
class BarangMakanan extends Barang { ... }
```

Alasannya karena Inheritance adalah pewarisan. Barang mewarisi property $data dari Database. Sedangkan BarangElektronik dan BarangMakanan mewarisi method dari Barang.

8. Polymorphism

```
class BarangElektronik extends Barang {
   public function info() { return "Barang Elektronik bergaransi"; }
}

class BarangMakanan extends Barang {
   public function info() { return "Barang Makanan ada kadaluarsa"; }
}
```

```
echo $elektronik->info();
echo $makanan->info();
```

Alasannya karena Polymorphism ditunjukkan dari method info() yang sama pada dua class berbeda (BarangElektronik dan BarangMakanan) tetapi implementasinya berbeda.

BAB II
PENUTUP

2.1 Kesimpulan

Pada pertemuan kali ini saya bisa memahami bahwa class adalah rancangan dasar untuk membuat object yang dapat menyimpan data melalui property dan menjalankan aksi melalui method. Lalu saya juga sekarang sudah megenal apa saja 4 pillar di OOP itu. Dengan enkapsulasi data lebih aman, sedangkan abstraksi membuat penggunaan method lebih sederhana. Inheritance memudahkan penggunaan kembali kode, dan polymorphism memungkinkan method yang sama menghasilkan perilaku berbeda. Secara keseluruhan, OOP membuat program lebih terstruktur, aman, dan fleksibel untuk dikembangkan.