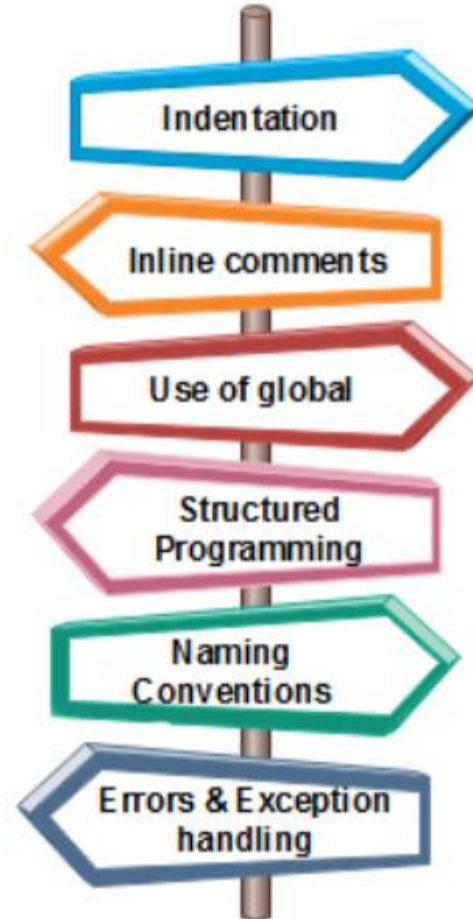


# Motivasi

So if you want to go fast, if you want to get done quickly, if you want  
your code to be easy to write, make it easy to read.

–Robert C. Martin

# Apersepsi

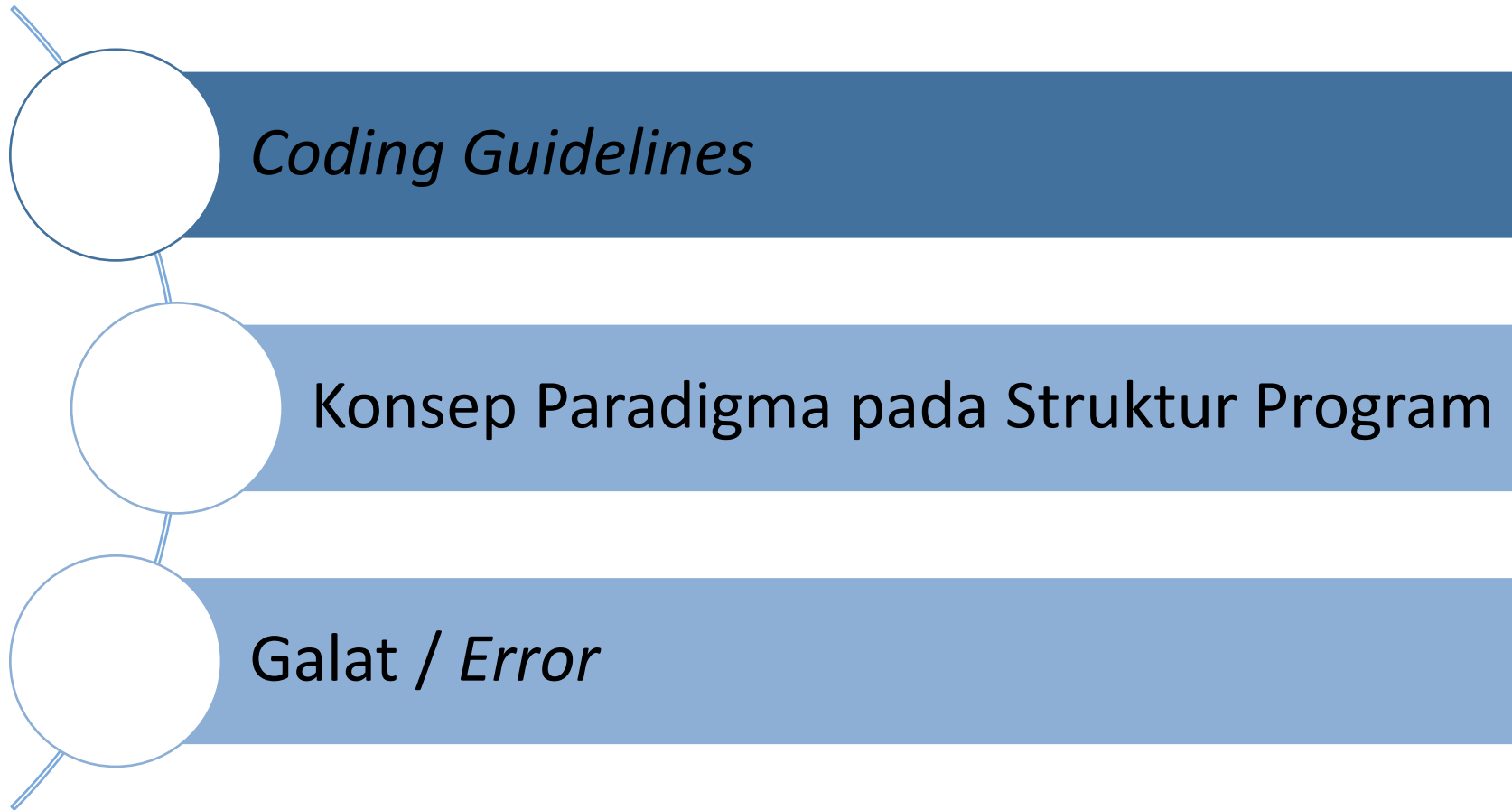


# Menerapkan *Coding Guidelines* dan *Best Practices* dalam Penulisan Program (Kode Sumber)

# Ringkasan Mata Pelatihan

- Unit Kompetensi Acuan: Menulis kode dengan prinsip sesuai *guidelines* dan *best practices*
- Kode Unit Kompetensi Acuan: J.620100.016.01
- Deskripsi singkat: Mata pelatihan ini menentukan kompetensi, pengetahuan dan sikap kerja yang diperlukan dalam menerapkan penulisan kode.
- Tujuan Pembelajaran: Peserta dapat menerapkan penulisan kode yang baik agar kode tersebut dapat dirawat (*maintainability*).

# Agenda



# *Coding Guidelines*

# *Coding Guidelines*

*Coding Guidelines* adalah acuan bagi *developer* untuk membuat kode program yang lebih mudah dibaca dan dipelihara.

Tujuan dari *coding guidelines*:

1. Menyeragamkan kode program yang dibuat oleh *developer* yang berbeda-beda
2. Mempermudah pemahaman isi kode program dan mengurangi kompleksitas program
3. Membantu kode program untuk bisa digunakan kembali (*reuse*)
4. Mempermudah mendeteksi kesalahan / *error*

# Penamaan

Penamaan *local variable*, *global variable*, *constant*, *function* dan *method* sebaiknya memperhatikan:

- Mudah dipahami
- Menghindari penggunaan angka
- Mendeskripsikan isi dengan singkat dan jelas



# Contoh Penamaan

- **Variable**

*Variable* menggunakan *camel case* dimana diawali dengan huruf kecil, sedangkan *global variable* diawali dengan kapital.

```
class FirstClass{  
    //global variable  
    InputProgram = 10;  
  
    function testVariable()  
    {  
        //local variable  
        inputTest = 1;  
    }  
}
```

# Contoh Penamaan (2)

- ***Constant***

*Constant* menggunakan huruf kapital semua atau huruf kapital dengan garis bawah sebagai pengganti spasi

```
class FirstClass{  
    const INPUT = 100;  
    const TEST_VERSION = 2;  
}
```

# Contoh Penamaan (3)

- **Function**

*Function* menggunakan *camel case* dimana diawali dengan huruf kecil.

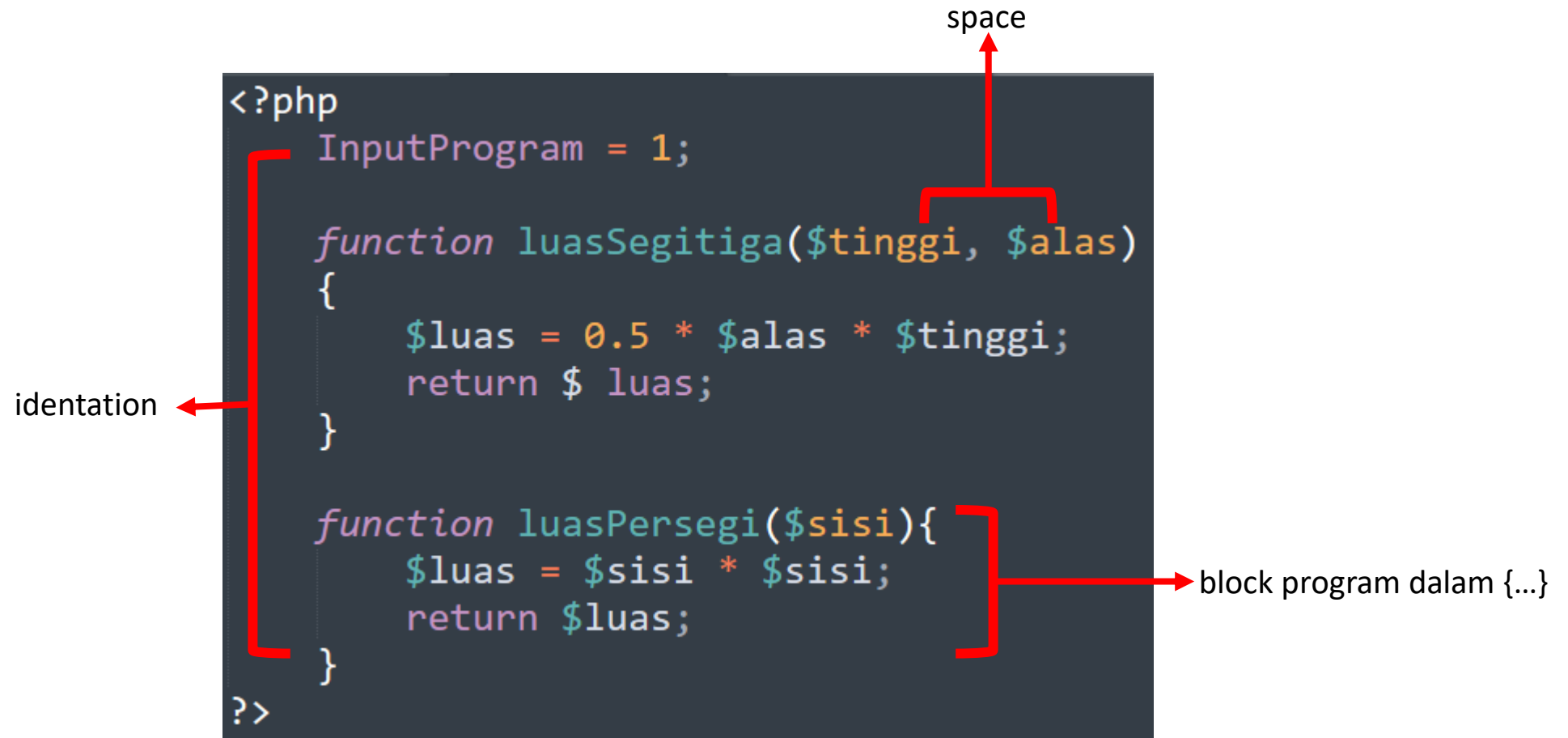
```
function luasSegitiga($tinggi, $alas)
{
    $luas = 0.5 * $alas * $tinggi;
    return $ luas;
}
```

# *Indentation*

*Indentation* sangat penting untuk mempermudah pembacaan kode program. Beberapa bagian dari *indentation*:

- Menambahkan spasi setelah koma diantara dua *argument function*
- Setiap *nested block* harus dilakukan *indentation*
- *Indentation* dilakukan pada awal dan akhir setiap *block* program
- Setiap *block* program ditulis didalam tanda kurung kurawal {...}

# Indentation (2)



The diagram shows a code editor with PHP code. A large red bracket on the left side of the code block is labeled "indentation". A red arrow points from the word "space" to the space between the opening curly brace and the first line of the `luasSegitiga` function. Another red arrow points from the text "block program dalam {...}" to the body of the `luasPersegi` function.

```
<?php
InputProgram = 1;

function luasSegitiga($tinggi, $alas)
{
    $luas = 0.5 * $alas * $tinggi;
    return $ luas;
}

function luasPersegi($sisi){
    $luas = $sisi * $sisi;
    return $luas;
}

?>
```

# *Structured Programming*

*Structured* atau *Modular Programming* harus digunakan dalam penulisan kode program. “**GO TO**” statements sebaiknya tidak digunakan untuk memudahkan pemahaman isi kode program.

## Menghindari penggunaan *identifier* untuk kebutuhan yang berbeda

Setiap *variable* yang digunakan harus mendiskripsikan kegunaannya. Penggunaan satu *variable* yang sama untuk beberapa tujuan yang berbeda harus dihindari untuk menghindari kesulitan di masa depan.

# Kode Program Harus Didokumentasikan

Setiap kode program harus didokumentasikan dalam bentuk komen yang mudah dipahami.

## *Error Return Values*

Setiap *function* dalam program ditangani dengan standard yang terorganisasi, misalnya setiap kesalahan (*error*) mengembalikan nilai 0 atau 1 untuk menyederhanakan *debugging*.

# Konsep Paradigma pada Struktur Program



# Paradigma Pemrograman

Paradigma pemrograman adalah cara untuk mengklasifikasikan kode program berdasarkan fitur program yang dibuat.

Jenis-jenis paradigma pemrograman:

1. *Procedural Programming*
2. *Logical Programming*
3. *Functional Programming*
4. *Object-Oriented Programming*

# *Procedural Programming*

*Procedural programming* merupakan paradigma pemrograman berdasarkan konsep bagaimana prosedur dipanggil, dimana kode program disusun dalam bentuk *list* instruksi yang menjelaskan tahap-tahapan yang harus dikerjakan.

Kelebihan *procedural programming*:

- Kode program *portable*.
- Program dapat digunakan kembali pada program lain tanpa perlu menyalinnya (*copy*).
- Alur program dapat ditelusuri dengan metode *top-down approach*.

# *Logical Programming*

*Logical programming* mempunyai dasar pada logika matematika dimana *program statements* mengekspresikan fakta dan aturan tentang pemecahan masalah.

Contohnya saat kita memesan kopi, *imperative approach* yang akan dilakukan adalah:

1. Masuk ke dalam toko kopi
2. Antri untuk memilih menu yang akan dibeli
3. Memesan menu yang dipilih
4. Memilih apakah kopi akan dinikmati di tempat atau dibungkus (*take away*)
5. Membayar
6. Mengambil pesanan dan meninggalkan tempat pemesanan

# *Functional Programming*

*Functional programming* mempunyai ciri dengan membangun struktur dan elemen program. *Functional programming* terdiri dari ***pure function*** dimana *function* berisi argument yang menerima inputan dan akan mengembalikan nilai.

Beberapa contoh *functional programming* adalah:

- *Pure function*
- *Recursion*

# Functional Programming (2)

- **Pure function**  
*Output* fungsi tergantung pada inputan yang diberikan.

```
function addNum($firNum, $secNum){  
    $sumNum = $firNum + $secNum;  
    return $sumNum;  
}
```

# Functional Programming (3)

- **Recursion**

Fungsi yang akan memanggil dirinya sendiri saat proses eksekusi.

```
function countInt ($input){  
    if($input <= 0){  
        return "Input a positive integer  
        ";  
    }  
    else if($input > 10){  
        return "Counting complete";  
    }  
    else{  
        return countInt($input + 1);  
    }  
}
```

# *Object-Oriented Programming*

*Object-oriented programming* merepresentasikan objek dalam bentuk ***class***. Setiap *class* akan menyimpan seluruh data dan fungsi serta dapat berinteraksi dengan *class* lain.

*Features* pada OOP:

- *Encapsulation*  
Fitur mendasar berfungsi untuk menyembunyikan detail yang tidak ingin ditampilkan. Konsep ini membungkus data dan *internal method* pada objek tersebut.
- *Inheritance*  
Mekanisme untuk mengambil sifat dari kelas lain dan digunakan dalam pembentukan hirarki yang saling berbagi atribut dan *method*.

# Object-Oriented Programming (2)

- *Data abstraction*  
Reduksi untuk menyederhanakan representasi keseluruhan data. *Data abstraction* biasanya adalah tahap pertama dari *database design*.
- *Polymorphism*  
Konsep OOP yang mengambil kemampuan dari *variable*, fungsi atau objek untuk membentuk berbagai bentuk (*forms*).



# Galat / *Error*

# Galat / *Error*

- ***Parse Errors (Syntax Error)***
  - Kesalahan *syntax* (kutip, kurung, titik koma, dll) dalam *script* yang biasanya diakibatkan oleh kesalahan pengetikan.
  - Pesan kesalahan akan muncul pada *outputnya* ketika dijalankan.
  - *Parse error* akan menghentikan proses eksekusi *script*.

```
15  echo "Junior Web Developer";  
16  echo "BPPTIK"  
17  echo "Kementerian Kominfo";
```

**Parse error:** syntax error, unexpected 'echo' (T\_ECHO), expecting ';' or ';' in C:\xampp\htdocs\test.php on line 17

```
15  echo "Junior Web Developer";  
16  echo "BPPTIK";  
17  echo "Kementerian kominfo";
```

Kesalahan terjadi karena kurangnya  
tanda titik koma (;) di baris ke 16

# Galat / *Error* (2)

- ***Fatal Errors***

- PHP mengerti kode program yang ditulis, namun apa yang diminta tidak dapat dilakukan.
- Misalnya program memanggil fungsi yang tidak terdefinisi.
- *Fatal error* akan menghentikan eksekusi *script*.

```
10     function luasPersegi($sisi){
11         $luas = $sisi * $sisi;
12         return $luas;
13     }
14
15     luasSegitiga();
```

Fungsi luasSegitiga() yang dipanggil pada baris 15 harus didefinisikan terlebih dahulu

**Fatal error:** Uncaught Error: Call to undefined function luasSegitiga() in C:\xampp\htdocs\test.php:15 Stack trace: #0 {main} thrown in C:\xampp\htdocs\test.php on line 15

# Galat / *Error* (3)

- ***Warning Errors***
  - File yang tidak ada atau jumlah parameter yang tidak pas saat pemanggilan suatu fungsi.
  - *Warning error* tidak akan menghentikan eksekusi dari *script*.

```
15 include ("connect.php");
```

File connect.php tidak ditemukan

**Warning:** include(connect.php): failed to open stream: No such file or directory in C:\xampp\htdocs\test.php on line 15

**Warning:** include(): Failed opening 'connect.php' for inclusion (include\_path='C:\xampp\php\PEAR') in C:\xampp\htdocs\test.php on line 15

# Galat / *Error* (4)

- ***Notice Errors***
  - Variable yang diakses belum didefinisikan.
  - *Notice error* tidak menghentikan *script*.

```
15     $lokasi = "BPPTIK Kementerian Kominfo";  
16     echo $waktu;
```

Variable \$waktu belum didefinisikan

Notice: Undefined variable: waktu in C:\xampp\htdocs\test.php on line 16

# Kesimpulan

# Kesimpulan

- *Coding Guidelines* adalah acuan bagi *developer* untuk membuat kode program yang lebih mudah dibaca dan dipelihara.
- Paradigma pemrograman adalah cara untuk mengklasifikasikan kode program berdasarkan fitur program yang dibuat.
- Galat/*error* adalah pesan yang akan muncul sesuai dengan kesalahan pada kode program.

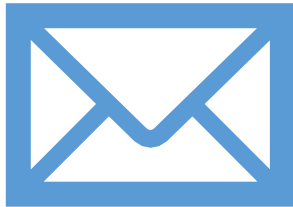
# Referensi / Bacaan Lebih Lanjut



# Referensi / Bacaan Lebih Lanjut

- <https://www.javatpoint.com/software-engineering-coding>
- <https://www.geeksforgeeks.org/coding-standards-and-guidelines/>
- <https://hackr.io/blog/programming-paradigms>

# Terima Kasih



Kantor:

Balai Pelatihan dan Pengembangan  
Teknologi Informasi dan Komunikasi  
Kementerian Kominfo

Website: <https://bpptik.kominfo.go.id>

Email: [bpptik@kominfo.go.id](mailto:bpptik@kominfo.go.id)

Twitter: @bpptik

Facebook: @bpptik

Instagram: @bpptik

Google Plus: +bpptikkemkominfo