DESAIN DAN PEMROGRAMAN BERORIENTASI OBJEK

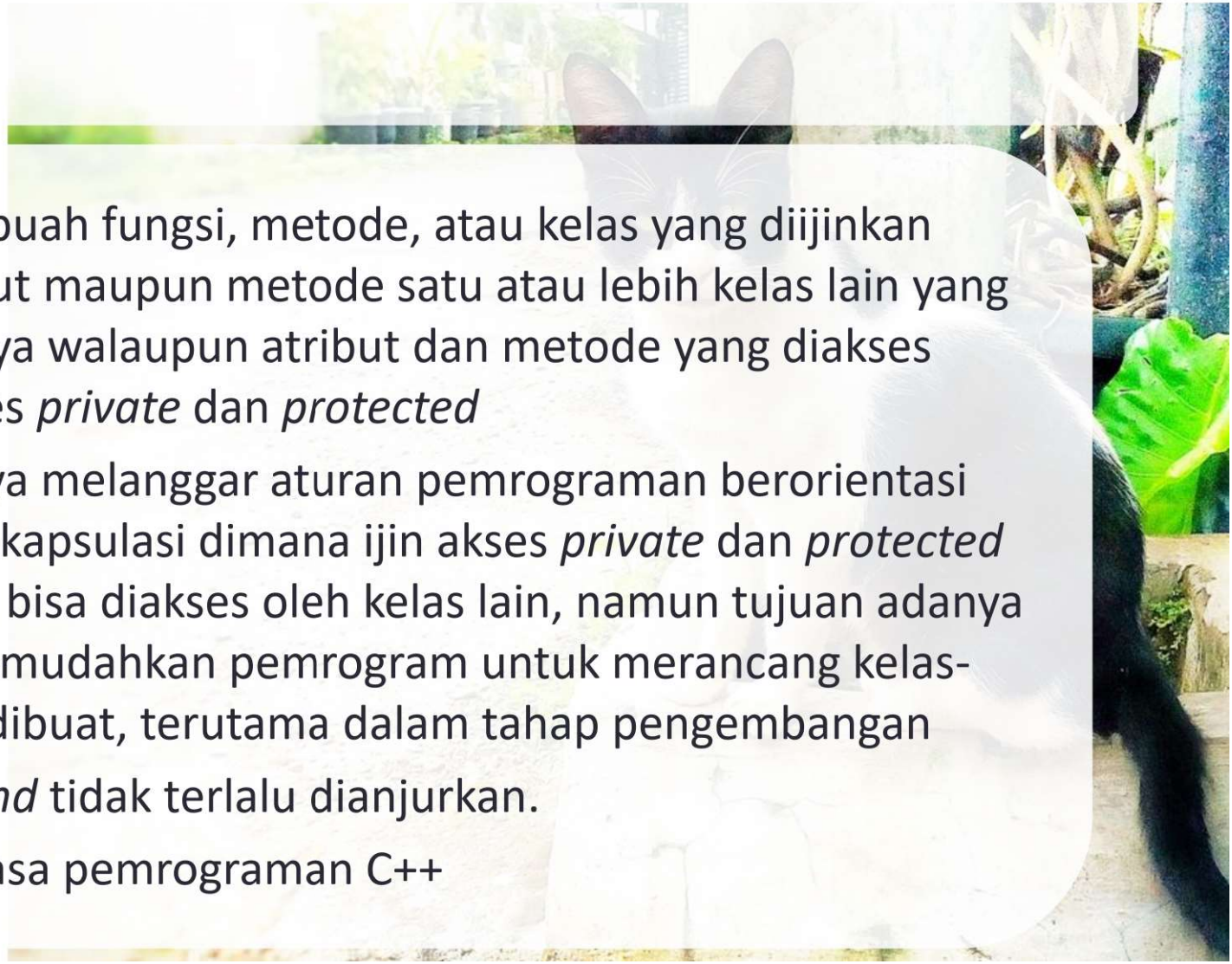
Friend

Rosa A. S.

Rosa Ariani Sukamto

- Blog: http://hariiniadalahhadiah.wordpress.com
- Facebook: https://www.facebook.com/rosa.ariani.sukamto
- Email: rosa.ariani@upi.edu
- Website: https://rosa-as.id
- Youtube: https://www.youtube.com/c/RosaArianiSukamto

Friend



- *Friend* adalah sebuah fungsi, metode, atau kelas yang diijinkan mengakses atribut maupun metode satu atau lebih kelas lain yang menjadi temannya walaupun atribut dan metode yang diakses memiliki ijin akses *private* dan *protected*
- *Friend* sebenarnya melanggar aturan pemrograman berorientasi objek tentang enkapsulasi dimana ijin akses *private* dan *protected* tidak sembarang bisa diakses oleh kelas lain, namun tujuan adanya *friend* adalah memudahkan pemrogram untuk merancang kelas-kelas yang akan dibuat, terutama dalam tahap pengembangan
- Penggunaan *friend* tidak terlalu dianjurkan.
- Hanya pada bahasa pemrograman C++

Friend – Fungsi / Prosedur

Kelas Titik

Atribut Private:
int x;
int y;

friend void
tampilkanTitik(Titik t1,
Titik3D t2);

Kelas Titik3D – anak Titik

Atribut Private:
int z;

Protected:
getZ()

friend void tampilkanTitik(Titik
t1, Titik3D t2);

```
void tampilkanTitik(Titik t1,
Titik3D t2)

t1.x
t1.y

t2.getZ()
```

Implementasi Friend - Fungsi/Prosedur (1)

Titik.cpp

```
class Titik3D;
class Titik{
  private:
    int x; /*koordinat x*/
    int y; /*koordinat y*/

  public:
    Titik(){
      /*konstruktor*/
      x = 0;
      y = 0;
    }
    Titik(int xp, int yp){
      /*konstruktor*/
      x = xp;
      y = yp;
    }
    void setX(int xp){
      /*mengembalikan nilai x*/
      x = xp;
    }

    void setY(int yp){
      /*mengembalikan nilai y*/
      y = yp;
    }
   /*fungsi friend*/
    friend void tampilkanTitik(Titik t1,
    Titik3D t2);
};
```

Implementasi Friend - Fungsi/Prosedur (2)

Titik3D.cpp

```
class Titik3D : public Titik{

  private:
   int z; // koordinat z

  public:
   Titik3D(){
     /*konstruktor*/
     z = 0;
   }

   Titik3D(int xp, int yp, int zp){
     /*konstruktor*/
     setX(xp);
     setY(yp);
     z = zp;
   }

   /*fungsi friend*/
   friend void tampilkanTitik(Titik
   t1, Titik3D t2);


protected:
    int getZ(){
      /*mengembalikan nilai z*/
      return z;
    }
};
```

Implementasi Friend - Fungsi/Prosedur (3)

main.cpp

```
#include <iostream>
using namespace std;
#include "Titik.cpp"
#include "Titik3D.cpp"

/*fungsi friend*/
//bisa hanya satu masukan
void tampilkanTitik(Titik t1, Titik3D
t2){
//bisa mengakses atribut private
  cout << "Fungsi Friend" << endl;
  cout << "Titik : x : " << t1.x << "  y
: " << t1.y << endl;
  cout << "Titik3D : x : " << t2.x << "
y : " << t2.y << "  z : " << t2.getZ()
<< endl;
  cout << "-----------------------------
-----------" << endl;
}
```

```
int main(){

  Titik t1(28, 1);
  Titik3D t2(28, 1, 7);

  tampilkanTitik(t1, t2);

  return 0;
}
```

Friend – Metode (hanya untuk metode tertentu)

```
Kelas Titik

Atribut Private:
int x;
int y;


friend void
SahabatTitik::printTitik(const
Titik &t)
```

```
Kelas SahabatTitik

Atribut Private:
int atributSahabatTitik;


void printTitik(const Titik &t)
t1.x
t1.y
```

Implementasi Friend - Metode (1)

SahabatTitik.cpp

```
class Titik;
class SahabatTitik{

  private:
    int atributSahabatTitik;

  public:
    SahabatTitik();

  void printTitik(const Titik &t);
  };
```

```
SahabatTitik::SahabatTitik(){
  /*konstruktor*/
  atributSahabatTitik = 89;
}
```

Implementasi Friend - metode (2)

Titik.cpp

```
class Titik{
  private:
    int x; /*koordinat x*/
    int y; /*koordinat y*/

  public:
    Titik(){
      /*konstruktor*/
      x = 0;
      y = 0;
    }

    Titik(int xp, int yp){
      /*konstruktor*/
      x = xp;
      y = yp;
    }

    void setX(int xp){
      /*mengembalikan nilai x*/
      x = xp;
    }

    void setY(int yp){
      /*mengembalikan nilai y*/
      y = yp;
    }
  /*metode friend*/
    friend void
    SahabatTitik::printTitik(const Titik
    &t);
};
```

Implementasi Friend - metode (3)

Titik.cpp

```
/*metode friend*/
void SahabatTitik::printTitik(const Titik &t){
  //bisa mengakses atribut private dari Titik
  cout << "Metode Friend" << endl;
  cout << "Titik : x : " << t.x << "  y : " << t.y << endl;
  cout << "---------------------------------------" << endl;
}
```

Implementasi Friend - metode (4)

main.cpp

```
#include <iostream>
using namespace std;
#include "SahabatTitik.cpp"
#include "Titik.cpp"

int main(){

  Titik t(28, 1);
  SahabatTitik f;

  f.printTitik(t);

  return 0;

}
```

Friend – kelas (Bisa untuk semua metode dalam kelas friend)

```
Kelas Sahabat

Atribut Private:
int atributSahabat


friend class TemannyaSahabat
```

```
Kelas TemannyaSahabat


void aksesSahabat(Sahabat tm)

tm. atributSahabat
```

Implementasi Friend - kelas (1)

Sahabat.cpp

```
class TemannyaSahabat;

class Sahabat{

  private:
    int atributSahabat;

  public:
    Sahabat(){
      /*konstruktor*/
      atributSahabat = 89;
    }
   /*kelas teman*/
    friend class TemannyaSahabat;

};
```

Implementasi Friend - kelas (2)

TemannyaSahabat.cpp

```
class TemannyaSahabat{

  public:
   TemannyaSahabat(){
     /*konstruktor*/

   }
   void aksesSahabat(Sahabat tm){
    //mengakses atribut private
     cout << "Kelas Friend" << endl;
     cout << "Nilai teman : " << tm.atributSahabat << endl;
     cout << "---------------------------------------" <<
endl;
   }

};
```

Implementasi Friend - kelas (3)

main.cpp

```
#include <iostream>
using namespace std;
#include "Sahabat.cpp"
#include "TemannyaSahabat.cpp"

int main(){

  Sahabat t;
  TemannyaSahabat kt;

  kt.aksesSahabat(t);

  return 0;

}
```

Contoh Soal (1)

Diberikan dua buah kelas:

Segiempat

memiliki atribut p yang berupa integer

memiliki atribut l yang berupa integer

memiliki metode untuk membuat segiempat sesuai masukan p dan l

misal p = 5 dan l = 4

*****

*****

*****

*****

dan

Segitiga

memiliki atribut n yang berupa integer

memiliki metode untuk membuat segitiga dengan bintang-bintang sesuai masukan

masukan 3

*

***

*****

Contoh Soal (2)

Dan diberikan pula sebuah kelas

BentukCampur

BentukCampur memiliki hubungan friend dari kelas Segiempat dan Segitiga dimana Bentuk campur dapat memiliki metode bentukRumah yang merupakan gabungan dari segitiga dan segiempat

```
    *
   ***
  *****
 *****
*****
*****
*****
```

dan menampilkan atribut n milik segitiga,dan atribut p dan l milik segiempat

Daftar Pustaka