

UML - Class Diagrams



What is UML Diagram?

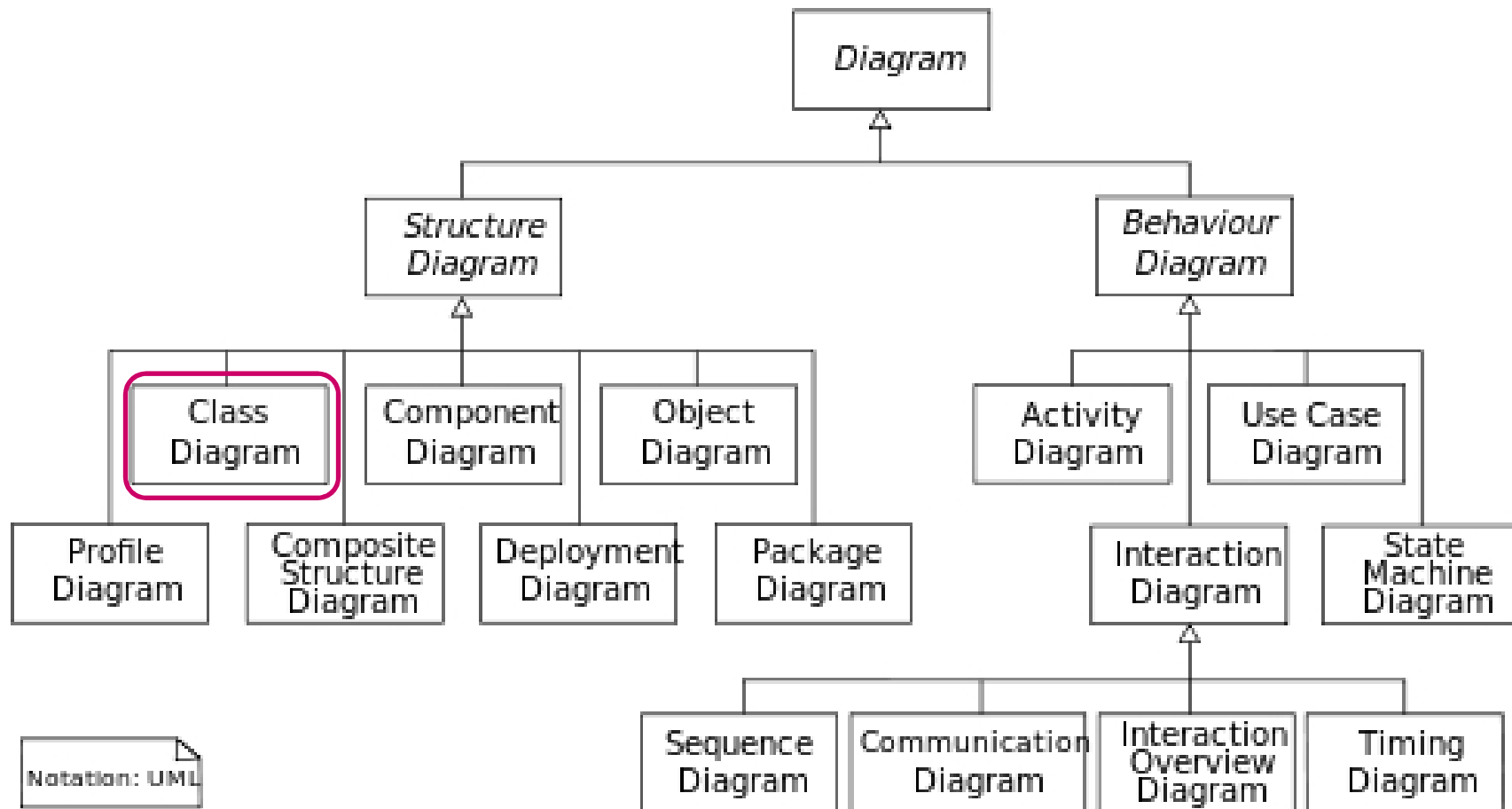
- Unified Modeling Language (UML)
- Standardized general-purpose **modeling language** in the field of **object-oriented software engineering**.
- The standard is managed, and was created, by the Object Management Group.



UML Diagram

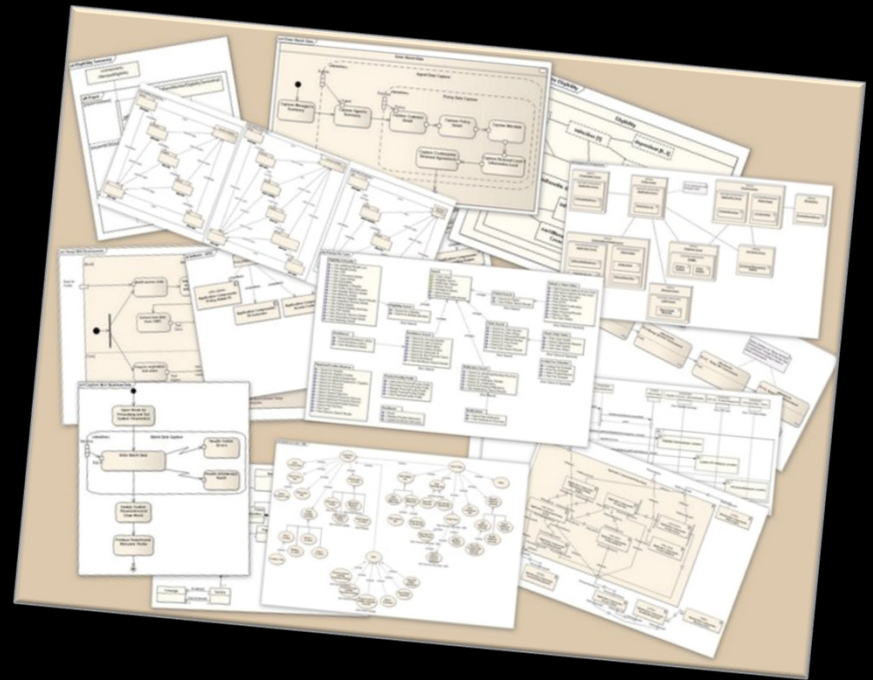
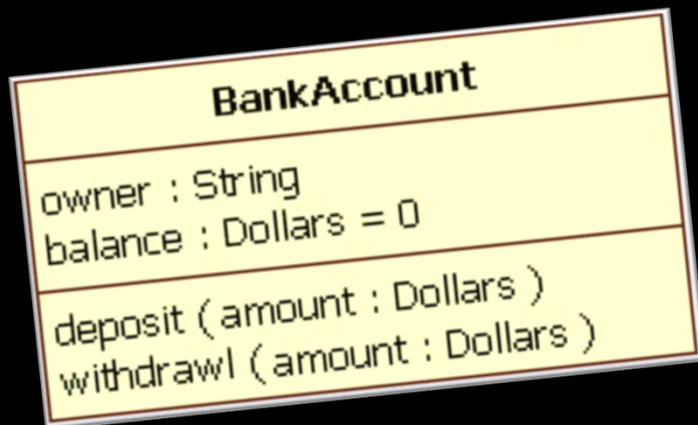
- UML 2.2 has **14 types of diagrams** divided into two categories.
- Seven diagram types represent **structural information**.
- Seven represent **general types of behavior**.

UML Diagram Types



What is class diagram?

- Is a type of **static structure diagram** that **describes the structure** of a system **by showing the system's classes**, their **attributes**, operations (or **methods**), and **the relationships** among the classes.



Class Diagram Models



Class Name

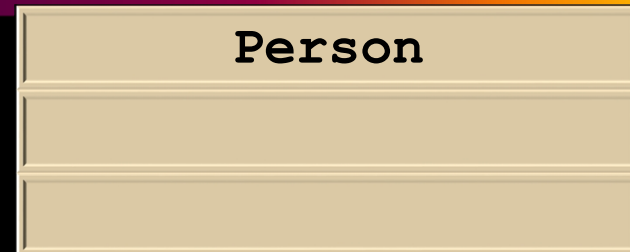
Attributes

Methods

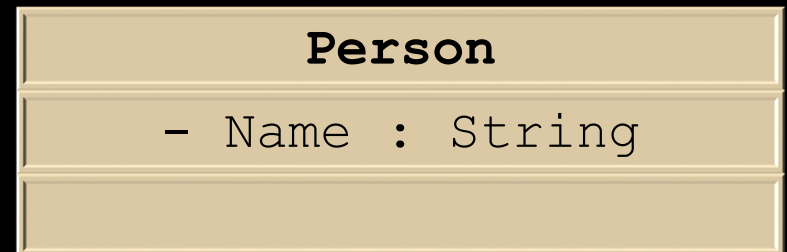
Example:

CD Models (Class and Attribute)

```
public class Person{  
  
}
```



```
public class Person{  
    private String name;  
}
```



Modifier

+ : Public
- : Private
: Protected
~ : Package

Example 2:

CD Models (Class, Attributes, Method)

MyCar.java

```
1 public class MyCar{
2     public String Color;
3     public int yearOfProduction;
4     public void printMyCar(){
5         System.out.println("Color: " + Color);
6         System.out.println("Year Of production: " + yearOfProduction);
7     }
8 }
```

MyCar

+ Color : String
+ yearOfProduction : int
+ printMyCar () : void

Example 3:

CD Models (Class with main method)

```
MyCar.java  MyCarInAction.java
```

```
1  public class MyCarInAction{
2      public static void main(String[] args){
3          MyCar myCar = new MyCar();
4
5          myCar.Color = "Black";
6          myCar.yearOfProduction = 2006;
7          myCar.printMyCar();
8      }
9  }
```

MyCarInAction

+ main(String[]): void

Exercise 1:

Create code from UML class diagram below!

MyCar2

```
+ Color : String  
+ yearOfProduction : int  
  
+ printMyCar (): void  
+ turnOnMechine (): void  
+ changeGear (): void  
+ turnOffMechine (): void
```

MyCarInAction2

```
+ main(String[]): void
```

Administrator: Command Prompt

```
D:\DOKUMEN\UDINUS\FASILKOM\PBO\_latihan\m7>javac MyCarInAction2.java
```

```
D:\DOKUMEN\UDINUS\FASILKOM\PBO\_latihan\m7>java MyCarInAction2
```

```
Color: Black
```

```
Year Of production: 2006
```

```
Mechine is turned ON
```

```
Changed gear
```

```
Mechine is turned OFF
```

```
D:\DOKUMEN\UDINUS\FASILKOM\PBO\_latihan\m7>
```

Example 4:

CD Models (Class with main constructor)

Constructor



Students

- nim : String
- nama : String

+ Students(nim : String, nama : String)
+ getNim () : String
+ getNama () : String

StudentsCard

+ main(String[]): void

Example 4:

CD Models (Class with constructor) Result

```
Students.java StudentsCard.java
1 public class Students{
2     private String nim;
3     private String nama;
4
5     public Students(String nim, String nama){
6         this.nim=nim;
7         this.nama=nama;
8     }
9
10    public String getNim(){
11        return nim;
12    }
13
14    public String getNama(){
15        return nama;
16    }
17 }

StudentsCard.java
1 public class StudentsCard{
2     public static void main(String[] args){
3         Students SiAdin = new Students("A11.2012.0001","SiAdin");
4         System.out.println("Student Name : " +SiAdin.getNama());
5         System.out.println("Student Number : " +SiAdin.getNim());
6     }
7 }
```

Exercise 2:

Create code from UML class diagram below!

Students2

- nim : String
- nama : String

+ Students2(nim : String)
+ setNim(String): void
+ setNama(String): void
+ getNim(): String
+ getNama(): String

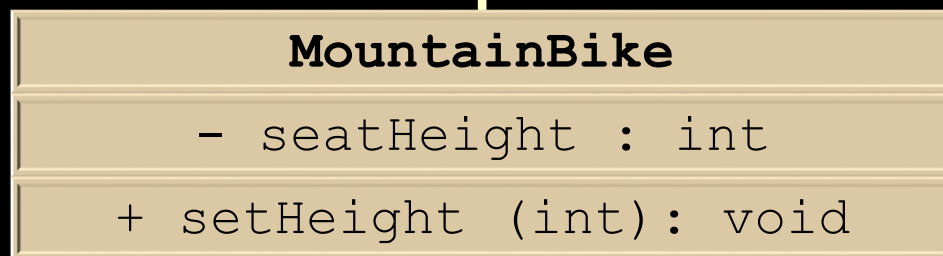
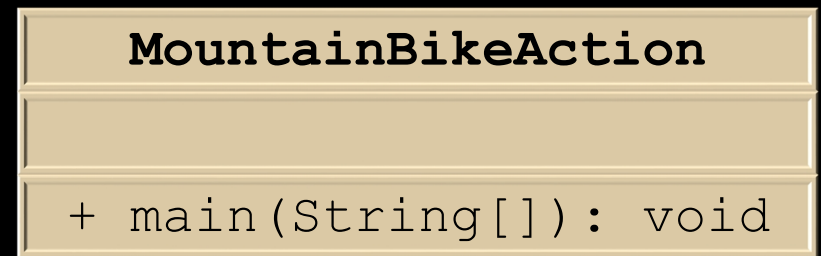
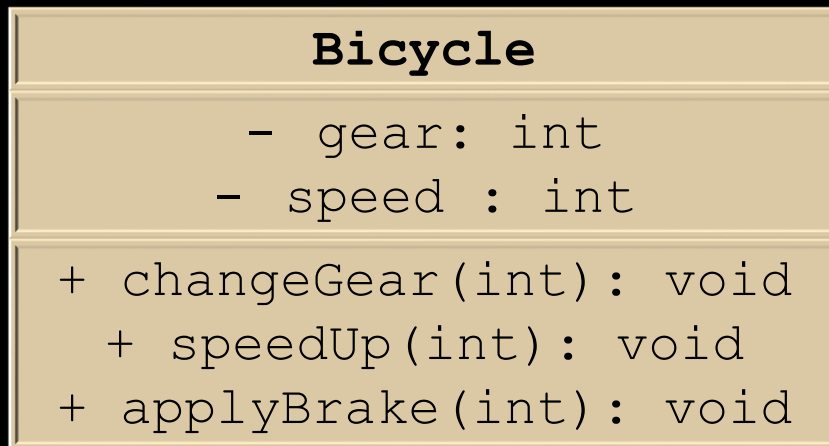
StudentsCard2

+ main(String[]): void

```
Administrator: Command Prompt
D:\DOKUMEN\UDINUS\FASILKOM\PBO\_latihan\m7>javac StudentsCard2.java
D:\DOKUMEN\UDINUS\FASILKOM\PBO\_latihan\m7>java StudentsCard2
Student Name = Si Adin
Student Number Format =xxx.xxxx.xxxxx
Student Number = A11.2012.00001
-----
Student Name : Si Adin
Student Number : A11.2012.00001
D:\DOKUMEN\UDINUS\FASILKOM\PBO\_latihan\m7>
```

With scanner
import java.util.scanner

Example 5: CD Models (Inheritance)



Example 5: CD Models (Inheritance) Result

```
Bicycle.java MountainBike.java MountainBikeAction.java
1 public class Bicycle{
2     //state
3     private int speed ;
4     private int gear ;
5
6     // method
7     public void changeGear(int newValue) {
8         gear = newValue;
9         System.out.println("Gear:" + gear);
10    }
11    public void speedUp(int increment) {
12        speed = speed+ increment;
13        System.out.println("Speed:" + speed);
14    }
15    public void applyBrake(int decrement) {
16        speed -= decrement;
17        System.out.println("Speed:" + speed);
18    }
19 }
```

Bicycle

- gear: int
- speed : int

+ changeGear(int): void
+ speedUp(int): void
+ applyBrake(int): void

Example 5:

CD Models (Inheritance) Result cont'd

The image shows a Java IDE with three tabs: Bicycle.java, MountainBike.java, and MountainBikeAction.java. The MountainBike.java tab is active, showing the following code:

```
1 class MountainBike extends Bicycle {
2     // the MountainBike subclass adds one f
3     private int seatHeight;
4
5     // the MountainBike subclass adds one m
6     public void setHeight(int newValue) {
7         seatHeight = newValue;
8         System.out.println("Seat height:" + seatHeight);
9     }
10 }
```

A callout box for the **MountainBike** class is shown to the right, containing the following information:

- MountainBike**
- seatHeight : int
- + setHeight (int): void

The MountainBikeAction.java tab is also visible, showing the following code:

```
1 class MountainBikeAction {
2     public static void main(String[] args) {
3         // Create a object
4         MountainBike mBike= new MountainBike ();
5
6         // Calling method
7         mBike.speedUp(10);
8         mBike.changeGear(2);
9         mBike.setHeight(20);
10    }
11 }
```

A callout box for the **MountainBikeAction** class is shown to the left, containing the following information:

- MountainBikeAction**
- + main(String[]): void

Exercise 3:

Create code from UML class diagram below!

Calc

```
# result: double = 0
# operand1: double = 0
# operand2: double = 0

# setOperand1(double): void
# setOperand2(double): void
# getOperand1(): double
# getOperand2(): double
# addition(): double
# subtraction(): double
# multiplication(): double
# division(): double
```

AdvancedCalc

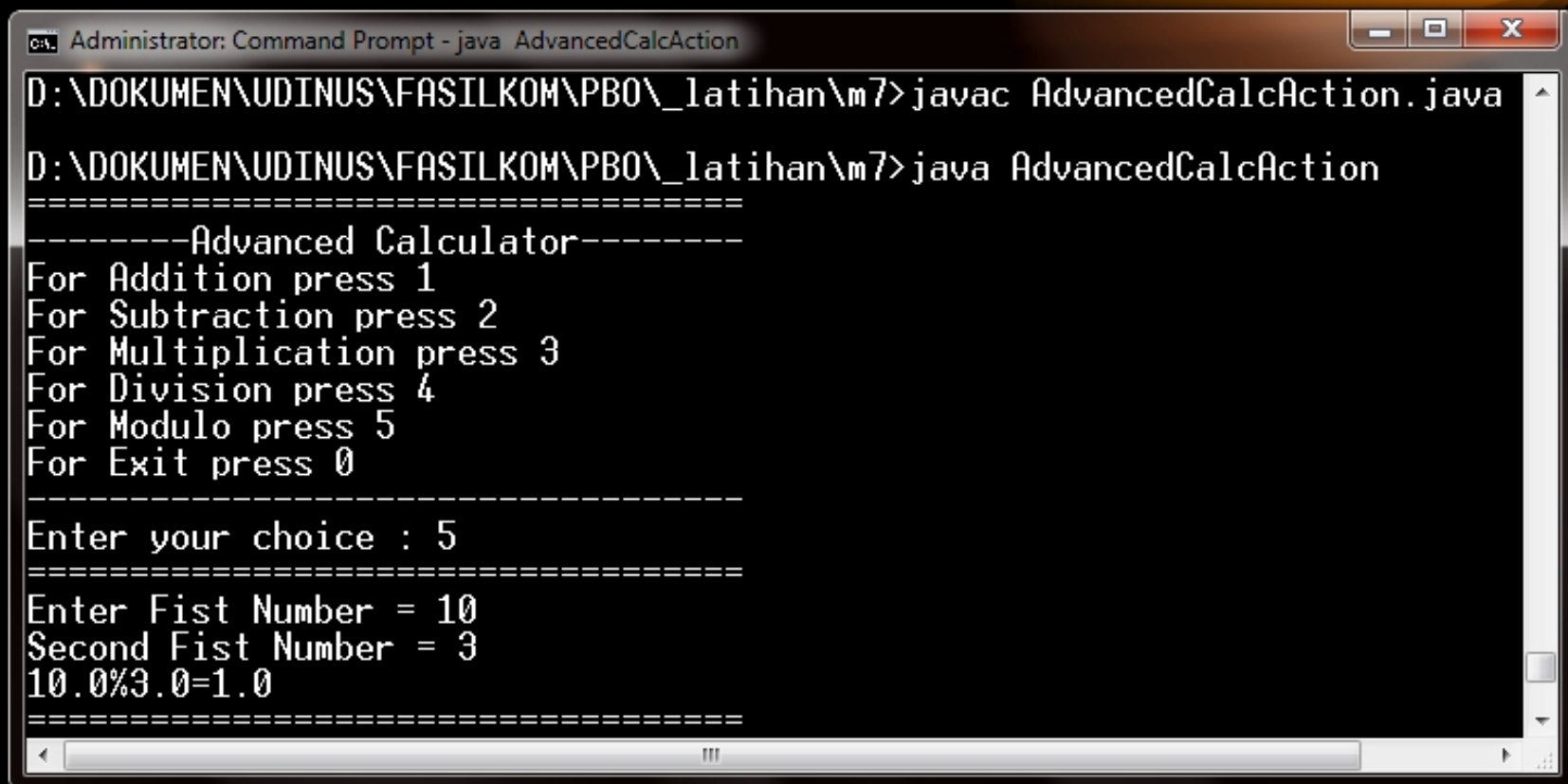
```
+ modulo(double, double): double
```

AdvancedCalcAction

```
+ main(String[]): void
```

Exercise 3:

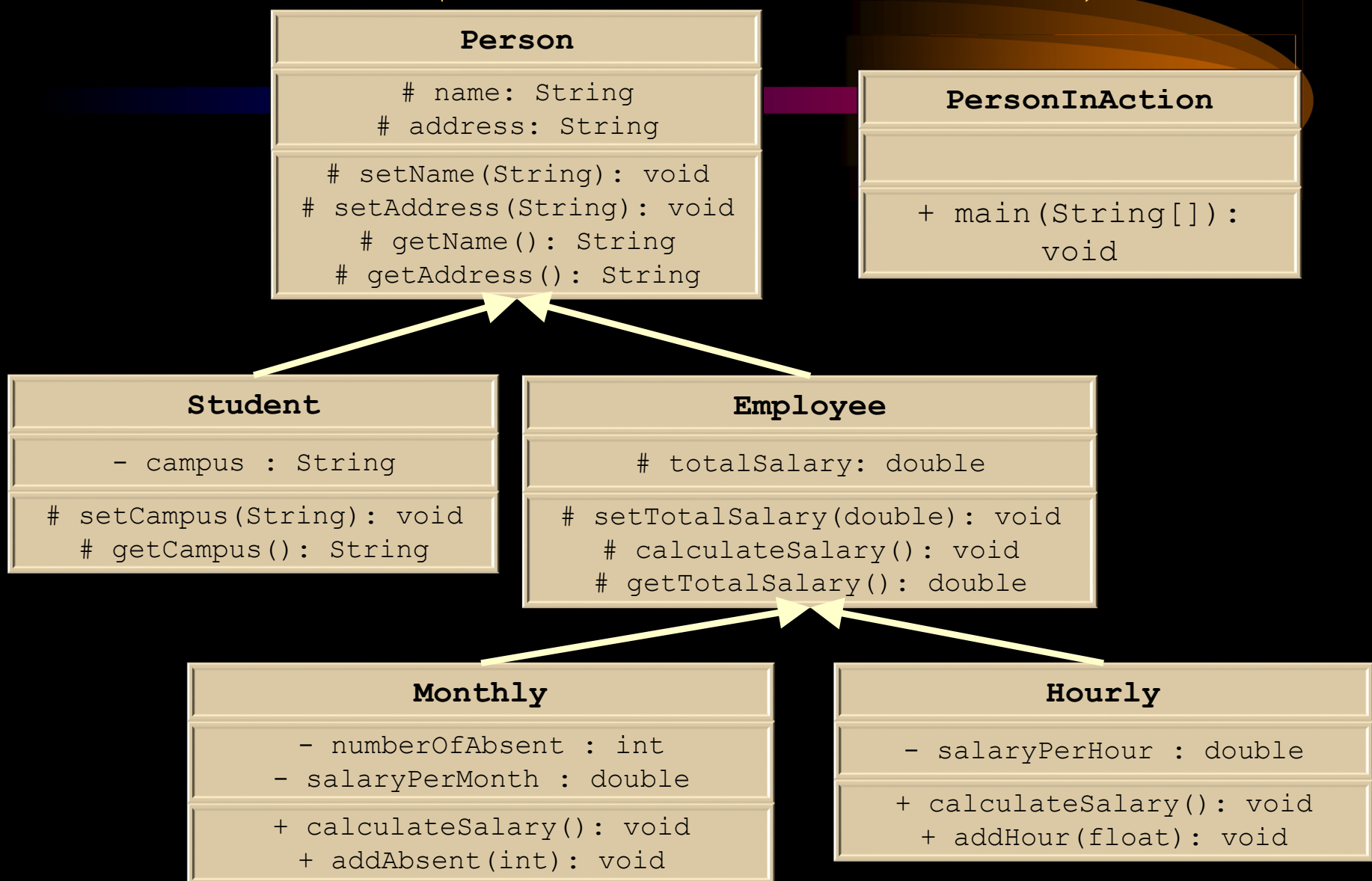
Result



```
Administrator: Command Prompt - java AdvancedCalcAction
D:\DOKUMEN\UDINUS\FASILKOM\PBO\_latihan\m7>javac AdvancedCalcAction.java
D:\DOKUMEN\UDINUS\FASILKOM\PBO\_latihan\m7>java AdvancedCalcAction
=====
-----Advanced Calculator-----
For Addition press 1
For Subtraction press 2
For Multiplication press 3
For Division press 4
For Modulo press 5
For Exit press 0
-----
Enter your choice : 5
=====
Enter Fist Number = 10
Second Fist Number = 3
10.0%3.0=1.0
=====
```

Example 6:

CD Models (Multilevel Inheritance)



Exercise 4:

Create code from class diagram in the previous slide

- Clue:
 - calculateSalary is polymorphism method

```
public class Person{  
}  
  
public class Student extends Person{  
}  
  
public class Employee extends Person{  
}  
  
public class Monthly extends Employee{  
}  
  
public class Hourly extends Employee{  
}
```

THANKS

