Week 5

User-Defined Class

(Class Definition)

Norida Muhd Darus

Outline

- Design class using UML notation
  - Class component
  - Example of class and its objects design using UML notation
  - Example of class definition
- Class field and method definition
  - Class header definition
  - Instance variable(s) definition
  - Method(s) definition
    - Method with and without return value
    - Method parameter(s)

Learning Objectives

- To understand the importance for defining programmer's own classes.
- To design class using UML graphical notation.
- To define class with attribute(s) and method(s).

Importance of User-Defined Classes

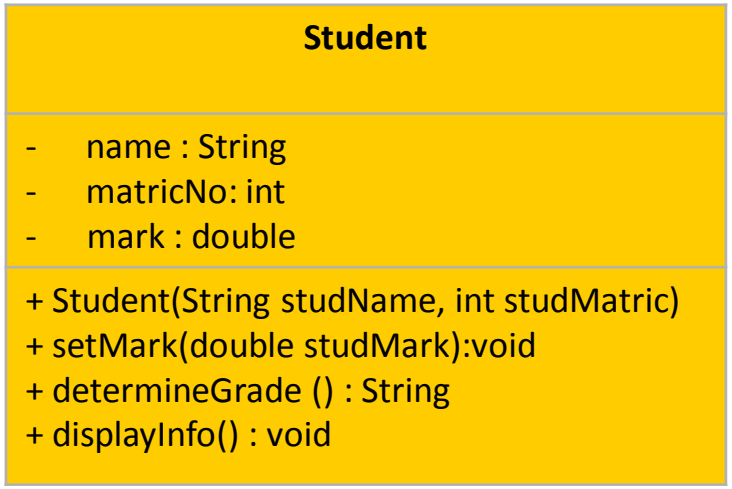
- The programs written in previous topics have used classes defined in the Java standard class library.
- In this topic, we will start to design programs that use classes that we define ourselves.
- A real object-oriented programming is depended on class definitions that represent objects with clear attributes and methods.

Class Component

- A class has:
  - Attributes (instance variables/data)
  - Methods (actions/operations)
- Example
  - **`Student`** class
    - Attributes : `name, matricNo, mark`
    - Methods : `setMark(), determineGrade(), displayInfo()`
  - **`UGStudent`** object
    - `name = "Ahmad", matricNo = 112233`

UML Class Diagram – Student class

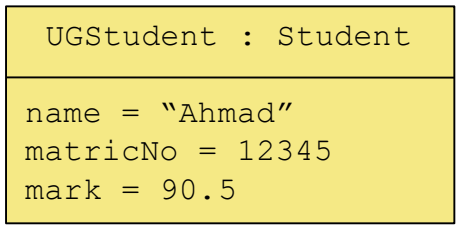
UML notation for Class ←

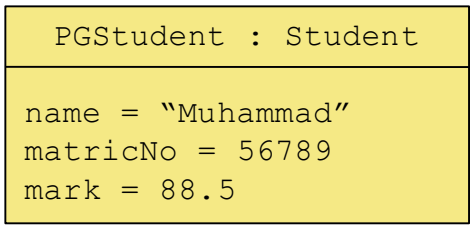


→ Class name

→ Data fields

→ Constructor and member methods





→ UML notation for objects

Example: Student Class

```
public class Student {
  //variables
  private String name;
  private int matricNo;
  private double mark;

  //methods
  public Student(String studName, int studMatric){
      name=studName;
      matricNo=studMatric;
  }

  public void setMark(double studMark){
     mark=studMark;
  }

  public String determineGrade(){
      String grade;
      if(mark > 39)
          grade="PASS";
      else
          grade="FAIL";
      return grade;
  }
```

Class name

Data fields

member methods

Define Class

- Use **`class`** keyword to define a class and put the class members inside curly braces - { }.
- Three steps to define a class:
  - STEP 1
    - Define class header
  - STEP 2
    - Define instance variables
  - STEP 3
    - Define member methods

Step 1 Define class header

- Syntax to define class header:

```
<accessModifier> final class <ClassName>
{
        <instance variable(s)>
        <member method(s)>
}
```

where:

- ***<accessModifier>*** – determine access rights for the class and its members.
- ***<className>*** – class name that we want to declare.

- Class naming tips:
  - Use a noun for the class name.
  - Begin the class name with a capital letter.
- **final** keyword indicates a class cannot have subclasses and it is optional.

Access Modifiers

Access Modifier	Class or member can be referenced by…
public	methods of the same class, and methods of other classes
private	methods of the same class only
protected	methods of the same class, methods of subclasses, and methods of classes in the same package
No access modifier (package access)	methods in the same package only

Step 1 Define class header: Example

- Examples:

```
public class Student
{
  ...
}
```

- The above class is an empty class and have no functionality.
- To make it works, we must include instance variables and member methods in curly braces (Step 2 and Step 3).
- Yet, we can compile and create an object from the empty class.

Step 2 Define instance variable(s)

- Syntax to define instance variable(s):

```
<accessModifier> final/static <dataType> <identifierList>;
```

where:

- ***<accessModifier>*** – determine access rights for the class and its members.
- ***<dataType>*** – can be primitive data type or a class type.
- ***<identifierList>*** – can contain one or more variable names of the same data type with or without initialization.

- The **final** keyword indicates the value for the instance variable is permanent and it is optional.
- The **static** keyword indicates the variable is a class variable. *Refer to the next topic for detail of static variable.*
- Define instance variable(s) before it can be used in member method(s).

Step 2 Define instance variable(s): Examples

- Examples:

```
private String name = " ";

private final int TOTAL_MARK = 100;

public int startX, startY, width, height;

public Student UGStudent, PGStudent;
```

Step 3 Define member method(s)

- Syntax to define member method(s):

```
<accessModifier> static <returnType> <methodName>(<parameter(s)>)
 {
      <method body>
 }
```

where:

- ***<accessModifier>*** – determine access rights for the member method(s).
- ***<returnType>*** – can be primitive data type, a class type or `void`.
- ***<methodName>*** – user-defined method name.
- **<parameter(s)>** - a comma-separated list of data types and variable names.

- The **static** keyword indicates the method is a static or class method. *Refer to the next topic for detail of static method.*
- Writing method name tips:
  - Use verb for method name.
  - Begin the method name with a lowercase letter and capitalize internal words.

Step 3 Define member method(s): Examples

```
//methods
public Student(String studName, int studMatric){
    name=studName;
    matricNo=studMatric;
}

public void setMark(double studMark){
   mark=studMark;
}

public String determineGrade(){
    String grade;
    if(mark > 39)
        grade="PASS";
    else
        grade="FAIL";
    return grade;
}
```

Define member methods

Types of Method

- Return a value
  - Must have `return` statement.
- Do not return a value (`void` method)
  - Has a `void` keyword as a returnType
  - Does not have `return` statement

Method That Return a Value

- Syntax:

```
public <returnType> <methodName>(<parameter(s)>)
{
    <statement(s)> with return expression.
 }
```

where:

- The return variable data type must have a type that matches the <returnType>.

- Example:

```
public String determineGrade(){
 String grade;
    if (mark > 39)
     grade="PASS";
 else
     grade="FAIL";
  return grade;
}
```

void Method

- Syntax:

```
public void <methodName>(<parameter(s)>)
{
  <statement(s)> without return expression.
}
```

- Example:

```
public void displayInfo(){
    System.out.println("Your name is"+name);
    System.out.println("Your matric number is"+matricNo);
    System.out.println("Your mark is"+mark);
    System.out.println("Your grade is"+determineGrade());
}
```

Method Parameters

- Methods can either have parameter or without parameter (with/without pass value)
- Example of method without parameter:

```
public String determineGrade(){
 String grade;
    if (mark > 39)
     grade="PASS";
 else
     grade="FAIL";
  return grade;
}
```

```
// Invocation of the method
...
   System.out.println("Your grade is"+determineGrade());
}
```

Method Passing Values - Parameters

- Formal parameters
  - Part of method definition
  - Put within parentheses after method name
    - type (e.g. int, float,…)
    - name (e.g. age, salary, …)
- Actual parameters (arguments)
  - Calling a method with an object within the parentheses
    - Must match data type
    - Must be in the same order

Parameter Passing: Example

```
// Definition of method to assign student's mark.
 public void setMark(double studMark){
     mark=studMark;
 }
```

What is the formal parameter in the method definition?

studMark

```
// Invocation of the method (somewhere in main method)
...
   double mark = read.nextDouble();
   UGStudent.setMark(mark);
}
```

What is the actual parameter in the method invocation?

mark

Complete Class Definition: Example

```
public class Student {
  //variables
  private String name;
  private int matricNo;
  private double mark;

  //methods
  public Student(String studName, int studMatric){
      name=studName;
      matricNo=studMatric;
  }

  public void setMark(double studMark){
     mark=studMark;
  }

  public String determineGrade(){
      String grade;
      if(mark > 39)
          grade="PASS";
      else
          grade="FAIL";
      return grade;
  }
```

STEP 1 – Define class header

STEP 2 – Define instance variables

STEP 3 – Define member methods

Complete Class Definition: Example (continue)

```
public void displayInfo(){
    System.out.println("Your name is"+name);
    System.out.println("Your matric number is"+matricNo);
    System.out.println("Your mark is"+mark);
    System.out.println("Your grade is"+determineGrade());
}
}
```

STEP 3 – Define member method

Summary

- In addition to predefined class defined in the Java standard class, we also can define our own class – user-defined class.
- UML class diagram is commonly used to design class and its objects.
- There are three (3) steps in defining our own class:
  - Step 1 - Define class header
  - Step 2 - Define instance variables
  - Step 3 - Define member methods
- Method can return a value or do not return a value (`void` method).