OBJECT ORIENTED PROGRAMMING

User-Defined Class

by
Dr. Nor Saradatul Akmar Zulkifli
Faculty of Computer Systems & Software Engineering
saradatulakmar@ump.edu.my

OER Object Oriented Programming by Dr. Nor Saradatul Akmar Binti Zulkifli work is under licensed Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.

Content Overview

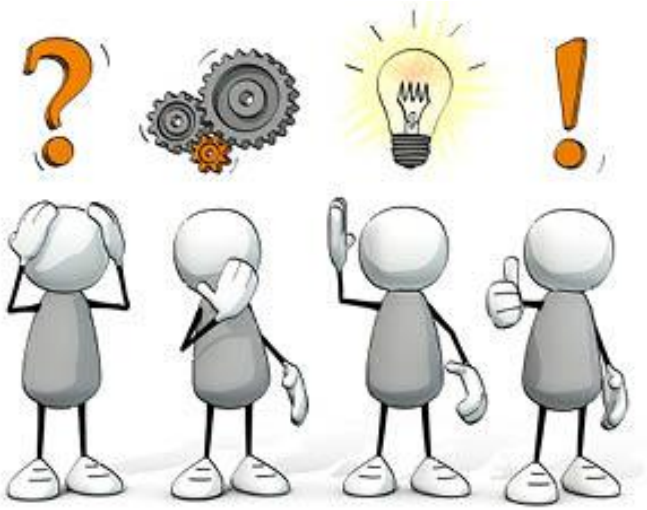
You will learn about:

- Create Classes & Objects
- Anatomy of Class Definition
- Variables in OOP (data member, argument, parameter & local variable)
  - Private and public (Accessibility modifiers)
  - General syntax for data member and method declarations
  - Arguments and parameters

Learning Objectives

Student should be able to:

- Define objects and classes
- Declare a class, data field/instance variable and methods
- Define a class with multiple methods and data members
- Create an object and invoke methods of a class
- Differentiate the local and instance variables

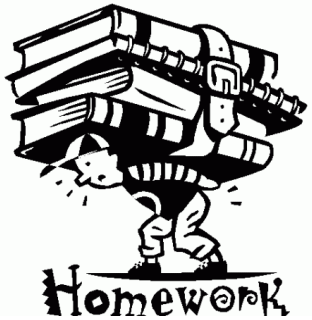


Universiti Malaysia PAHANG
Engineering • Technology • Creativity

PROCEDURAL VS. OBJECT-ORIENTED

ABSTRACTION VS. ENCAPSULATION

What Are The Differences??

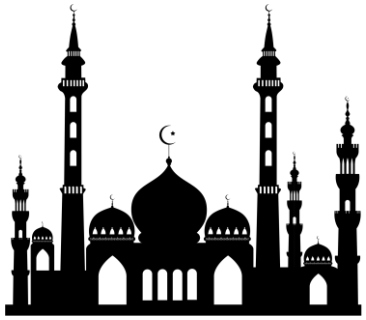


ReVISION

Universiti Malaysia PAHANG

Class
Definition of objects that share structure, properties and behaviors

Skyscrapers
Class



Mosques
Class

Berries
Class

Instance
Concrete Object, Created from a certain class

KL Tower
Instance of ***Skyscraper***

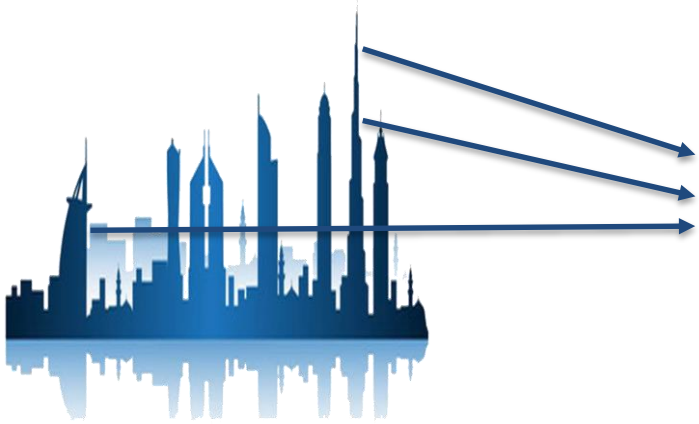
Beth Mosque
Instance of ***Mosque***

Strawberry
Instance of ***Berry***

CLASS VS. OBJECT

Universiti Malaysia PAHANG

Class Skyscraper



Instances

Attributes / Properties

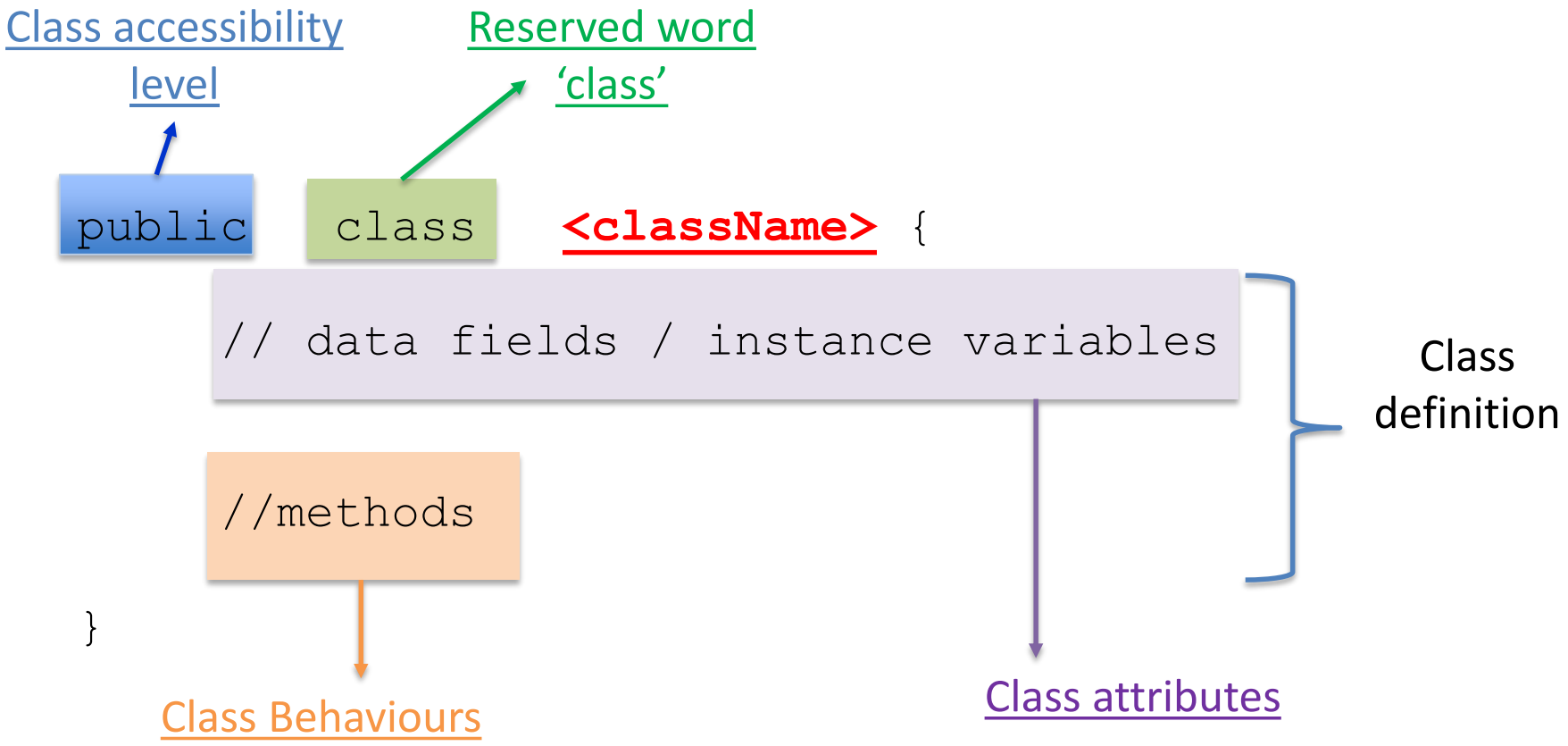
- name
- built
- country

Behaviours

- set ticket
- update ticket
- print ticket
- register visitor

Skycraper1	skycraper2	skycraper3
name = “Burj Khalifa” built = 2010 country = “UAE” set ticket update ticket print ticket register visitor	name = “Taipei 101” built = 2004 country = “China” set ticket update ticket print ticket register visitor	Name = “Petronas Twin Tower” built = 1996 country = “Malaysia” set ticket update ticket print ticket register visitor

ANATOMY OF JAVA CLASS



ANATOMY OF JAVA CLASS

Class accessibility level

Reserved word 'class'

```
public class <className> {

    <AccessLevel> <DataType> <variableName>

    <AccessLevel> <ReturnType> <methodName> (<parameters>)

    //code
}
```

Where;

- Access Level – public, private, default and protected
- DataType – data type of variable e.g. int, double, String etc
- ReturnType – data type returned by the method

CLASSES

Pre-Defined Classes

Classes defined in JAVA standard class library

Most commonly used :

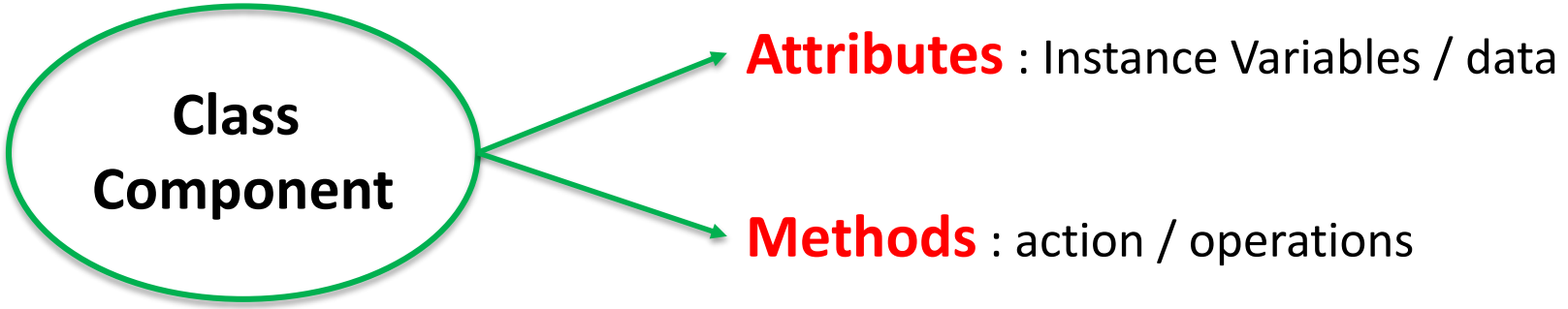
- ❑ String
- ❑ Math
- ❑ Random
- ❑ Scanner
- ❑ Primitive type wrapper

User-Defined Classes

Classes that defined by the programmer

Define own classes is the first step towards mastering the skill necessary in building large programs.

PROGRAMMER-DEFINED CLASS

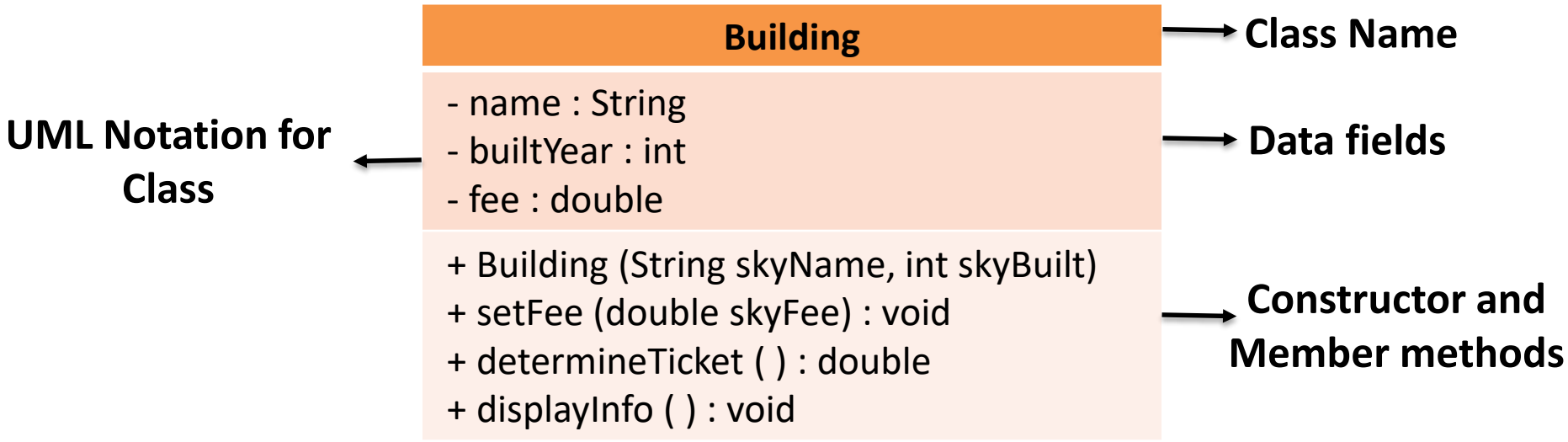


Example:

1. `Building` Class
   a. Attributes : `name`, `country`, `built`
   b. Methods : `setFee ( )`, `determineNationality ( )`, `displayInfo ( )`

2. `Skyscrapper` Class
   a. `name = "Petronas Twin Tower", country = "Malaysia", built = 1996`

PROGRAMMER-DEFINED CLASS : UML CLASS DIAGRAM




```
SkyScraper : Building
name = "Petronas Twin Towers"
builtYear = 1996
fee = 50.0
```

```
Mosque : Building
name = "Masjid Zahir"
builtYear = 1912
fee = 0.0
```

UML Notation for Objects

EXAMPLE : Class Bicycle

```
class Bicycle {
    // Data member
    private String ownerName;

    //Constructor: Initializes the data member
    public Bicycle(){
        ownerName = "Unknown";
    }
    //Returns the name of this bicycle's owner
public String getOwnerName(){
    return ownerName;
}
//Assigns the name of this bicyle's owner
public void setOwnerName(String name){
    ownerName = name;
    }
}
```

Class Name

Data fields

Constructor and Member methods

EXAMPLE : MAIN CLASS

```
class BicycleRegistration {

    public static void main(String[] args) {
        Bicycle bike1, bike2; //Create instances of class Bicycle
        String owner1, owner2;

        bike1 = new Bicycle(); //Create and assign values to bike1
        bike1.setOwnerName ("Sara Zulkifli");

        bike2 = new Bicycle(); //Create and assign values to bike2
        bike2.setOwnerName ("Ben Jones");

        owner1 = bike1.getOwnerName ( );  //Output the information
        owner2 = bike2.getOwnerName ( );
        boolean owner;

        System.out.println(owner1 + " owns a bicycle.");
        System.out.println(owner2 + " also owns a bicycle.");
    }
```

EXAMPLE : CLASS BICYCLE

- The dependency diagram between both programming

- The key differences : The use of bicycle class instead of standard classes
- Two classes created : BicycleRegistration (main class) and Bicycle


BicycleRegistration.java

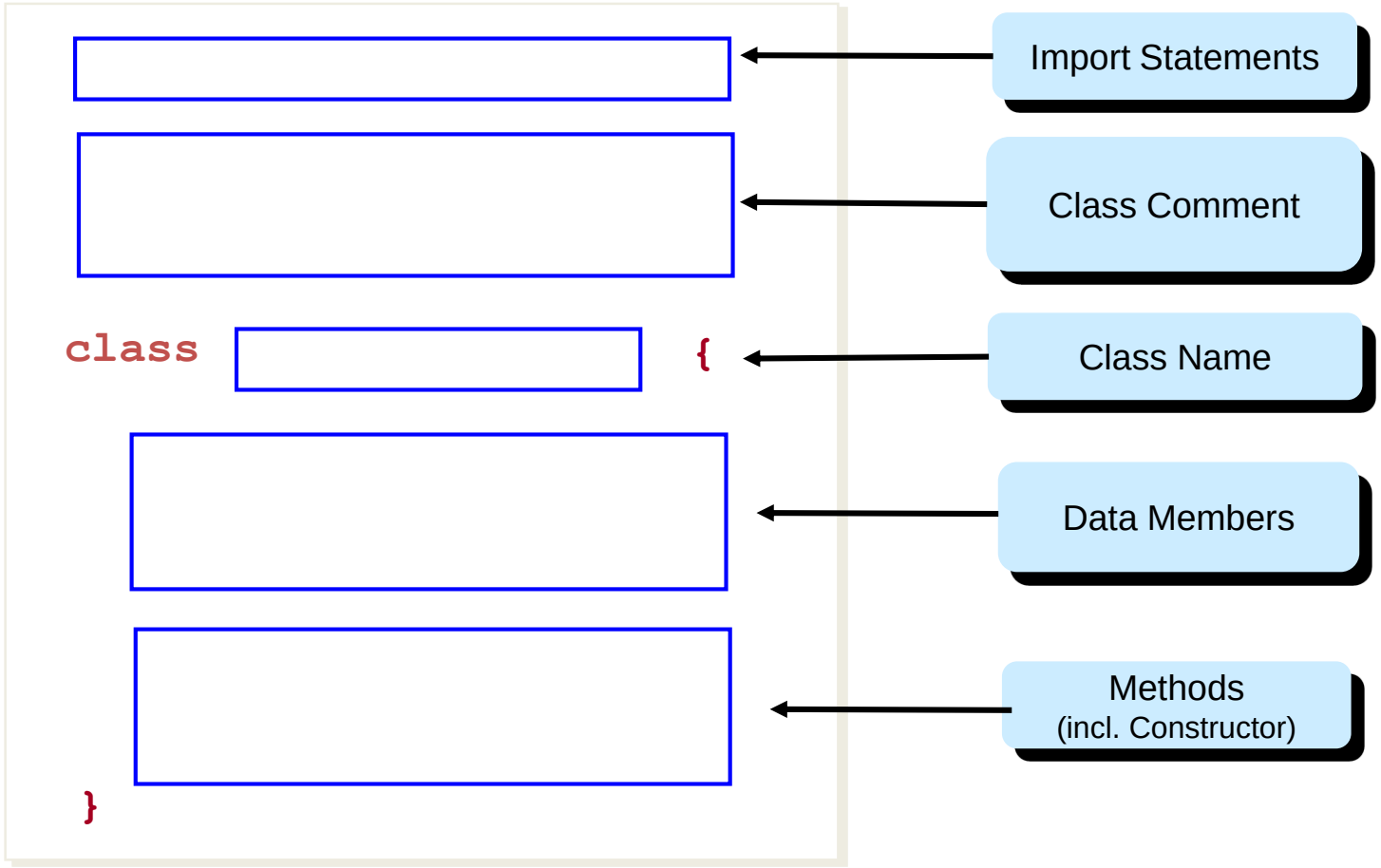

Bicycle.java

There are two source files. Each **class definition** is stored in a separate file.

To run the program: 1. javac Bicycle.java (compile)
2. javac BicycleRegistration.java (compile)
3. java BicycleRegistration (run)

REMEMBER!
Place all source files for a program in the same folder (directory)

ANATOMY OF CLASS DEFINITION



CLASS DEFINITION

Universiti Malaysia PAHANG

✓ To define a class, use `class` keyword and put the class members inside curly braces { }

1 Define Class Header

- 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's optional

2 Define Instance Variables

- `final` keyword indicates the value for the instance variable is permanent and it's optional
- `static` keyword indicates the variable is a class variable
- Define before it can be used.

3 Define member methods

- Use verb for the method name
- Begin the method name with lowercase letter and capitalize internal words.
- `static` keyword indicates the method is a static or class method

CLASS DEFINITION : SYNTAX

Class Header

```
<accessModifier> final class <className> {

        < instance variable(s) >
        < member method(s) >
}
```

Instance Variable(s)

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

CLASS DEFINITION : SYNTAX

Member Method(s)

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

Where ;

<accessModifier>	- determine **access right** for the class and its members
<className>	- class name that a programmer want to declare
<dataType>	- can be **primitive data type** or a class type
<identifierList>	- contain one or more variables names
<returnType>	- either primitive data type, a class type or void
<parameter(s)>	- a comma separated list of data types and variable names.

CLASS DEFINITION : 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

Additional:
The implementation of Access Modifiers for attributes and methods will be discuss in Lecture 7: Class Member Accessibility

CLASS DEFINITION : DATA TYPES

A **variable** can be either a primitive type or an object reference

PRIMITIVE DATA TYPES

- Numerical data
- A primitive variable contains **the value itself**

REFERENCE DATA TYPES

- Known as object reference variable
- An object reference variable holds **the address of an object**
- The contents are addresses that refer to memory location where the objects are actually stored
- Or thought of as a pointer to the location of the object

CLASS DEFINITION : RETURN TYPES

```
public void setOwnerName(String name){
    ownerName = name;
    }
```

Void Method

- A method that does not return a value
- Declare as void

```
public String getOwnerName(){
    return ownerName;
}
```

Value-returning Method

- When it is called, its return a value to the caller
- For this sample, since getOwnerName( ) method returns a string value – the value of instance variable ownerName – so it's return type is declare as string
- MUST include a return statement

return <expression> ;

EXAMPLE :
THE DEFINITION OF THE CLASS

```
class Bicycle {
    // Data member
    private String ownerName;

    //Constructor: Initializes the data member
    public Bicycle(){
        ownerName = "Unknown";
    }
    //Returns the name of this bicycle's owner
public String getOwnerName(){
    return ownerName;
}
//Assigns the name of this bicyle's owner
public void setOwnerName(String name){
    ownerName = name;
    }
}
```

Template:

```
class Bicycle {
    //data members
    //methods
}
```

We define the data member ownerName of the Bicycle Class

EXAMPLE :
THE DEFINITION OF THE CLASS

```
class Bicycle {
    // Data member
    private String ownerName;

    //Constructor: Initializes the data member
    public Bicycle(){
        ownerName = "Unknown";
    }
    //Returns the name of this bicycle's owner
public String getOwnerName(){
    return ownerName;
}
//Assigns the name of this bicyle's owner
public void setOwnerName(String name){
    ownerName = name;
    }
}
```

The syntax for **Data Member Declaration**

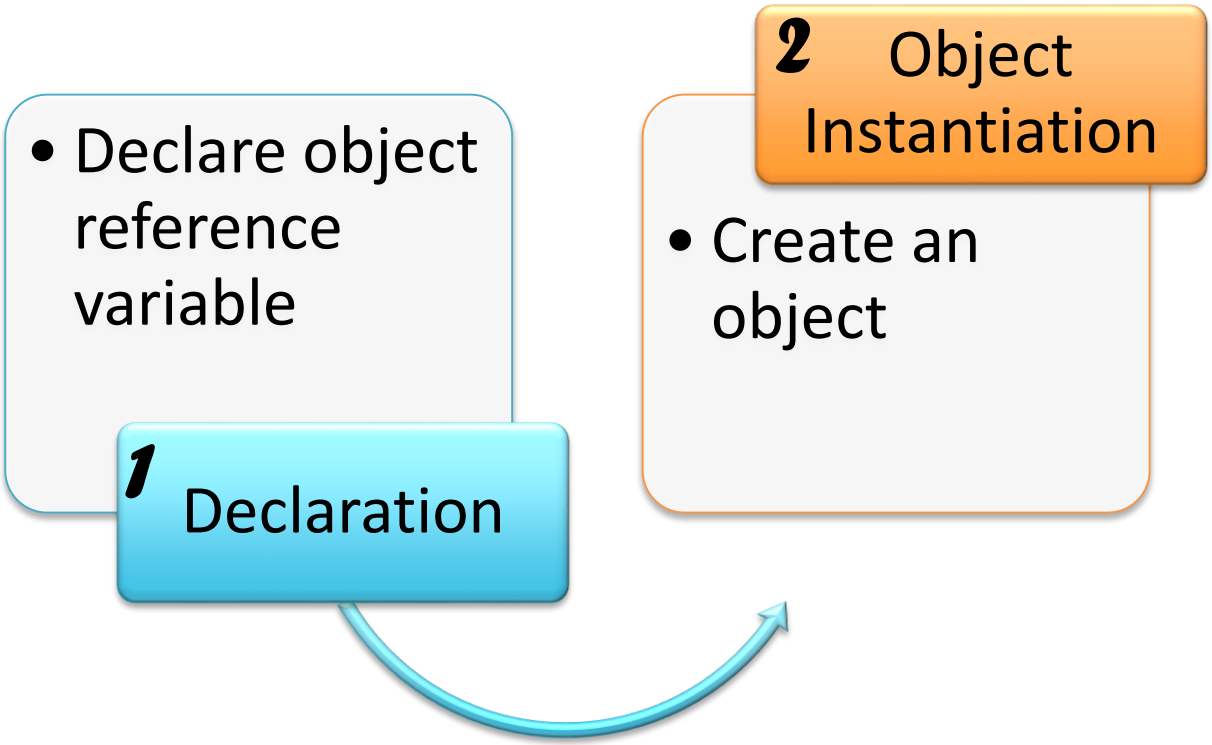
<modifiers> <data type> <name> ;

The syntax for **Method Declaration**

<modifiers> <return type>
<method name>
(<parameters>)
{
    <statements>
}

USER-DEFINED CLASS

Programmer need to create an object from the user-defined class in order to use it in a program



DECLARE AN OBJECT

Universiti Malaysia PAHANG

SYNTAX

```
<className>    <object reference variable>;
```

EXAMPLE:

```
Building     skyscraper1;
Person       student;
Vehicle      car;
Building     skyscraper1, skyscraper2;
```

Also known as declaring of an ***object reference***

Class Name
Must be defined before declaration can be stated

Object Reference Variable
One object is declare for 1 name

CREATE AN OBJECT

Universiti Malaysia PAHANG
Engineering • Technology • Creativity

SYNTAX

```
<object reference variable> = new <class name>
(<arguments>);
```

EXAMPLE:

Class Name
An instance of this class is created

Also known as assigning an ***object reference to the object***

```
skyscraper1 = new Building ("Burj Khalifa",2004);
Skyscraper2 = new Building ("KLCC", 1996);
Student     = new Person ("Ammar",01982);
```

Object reference variable

Arguments

DECLARE & CREATE AN OBJECT

Universiti Malaysia PAHANG
Engineering • Technology • Creativity

SYNTAX

```
<class Name> <object reference variable> = new
<class name> (<arguments>);
```

EXAMPLE:

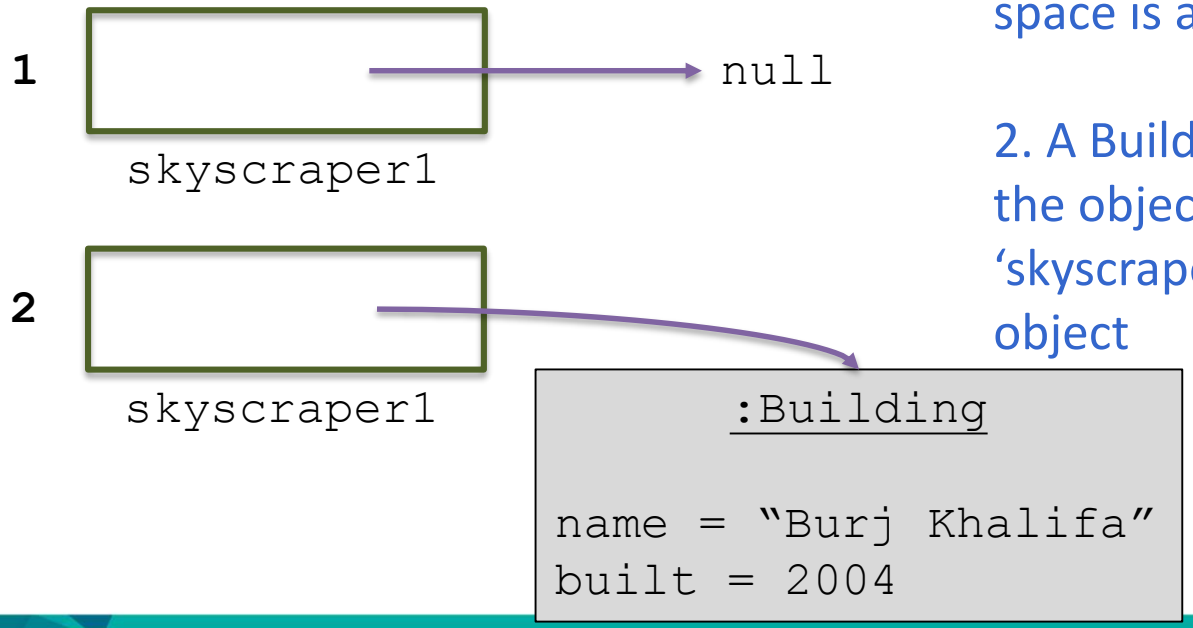
```
Building skyscraper1 = new Building ("Burj Khalifa",2004);
Building skyscraper2 = new Building ("KLCC", 1996);
Person student       = new Person ("Ammar",01982);
```

DECLARE Vs. CREATE AN OBJECT

EXAMPLE:

```
1. Building     skyscraper1;
2. skyscraper1 = new Building ("Burj Khalifa",2004);
```

MEMORY STATE



1. An object reference variable named 'skyscraper1' is declared and space is allocated in memory

2. A Building object is created and the object reference variable 'skyscraper1' is set to refer to the object

EXAMPLE : USING THE CLASS BICYCLE

Universiti Malaysia PAHANG

Memory State?

```
class BicycleRegistration {

    public static void main(String[] args) {
        Bicycle bike1, bike2; //Create instances of class Bicycle
        String owner1, owner2;

        bike1 = new Bicycle(); //Create and assign values to bike1
        bike1.setOwnerName ("Sara Zulkifli");

        bike2 = new Bicycle(); //Create and assign values to bike2
        bike2.setOwnerName ("Ben Jones");

        owner1 = bike1.getOwnerName ( );  //Output the information
        owner2 = bike2.getOwnerName ( );
        boolean owner;

        System.out.println(owner1 + " owns a bicycle.");
        System.out.println(owner2 + " also owns a bicycle.");
    }
```

INVOKING METHODS

```
owner1 = bike1.getOwnerName ( );
owner2 = bike2.getOwnerName ( );
```

- We send a *message* to the object to instruct an object to perform a task
- ONLY send a message that the objects can understand!
- To enable an object to handle the received message, it must posses a matching *method.* (The *behaviours* of an object implemented in classes are ***methods***)

Message `getOwnerName` is sent to a Bicycle object `bike1` → Bike1 : Bicycle

- Use dot operator (.) to invoked its method and to access a data field in the object

INVOKING METHODS

SYNTAX:

```
<objectReferenceVariable>.<methodName>(<arguments>);
```

EXAMPLE:

Method Name
The name of the message being send

```
skyscraper1.printTicket ( ) ;
Skyscraper1.editProfile ("BurjKhalifa", 2004);
```

Object reference variable

Argument(s)
Argument passing with the message

EXERCISE

> Imagine you are given the task of designing an airline reservation system that keeps track of flights for a commuter airline. Create and declare the objects involved in the system. Include the member methods and invoke the method necessarily.
>
> C.Thomas Wu
> Introduction to Object-Oriented Programming

The OBJECTIVE of this Exercise is to give you a continuous experience in building your own user-defined program.

Author Information

Dr. Nor Saradatul Akmar Binti Zulkifli

Senior Lecturer
Faculty of Computer Systems & Software Engineering
Universiti Malaysia Pahang