



THE UNIVERSITY OF
MELBOURNE

COMP90041

Programming and Software Development

Tutorial 4 The Anatomy of Classes & Methods

Chuang(Frank) Wang

Slides were developed by Chuang Wang
Copyright @ The University of Melbourne





Overview

1. Classes & Objects
2. Methods
3. Exercises & Lab



THE UNIVERSITY OF
MELBOURNE

1. Classes & Objects



Classes & Objects

Class (Template)

A class holds operations and data related to one concept.

Object (Instance)

An object(instance) is created based on template (a class's definition)



Create an object

Constructor

A special method that lives in the class instantiates an object(instance)

```
Classname myObject = new Classname(..);
```



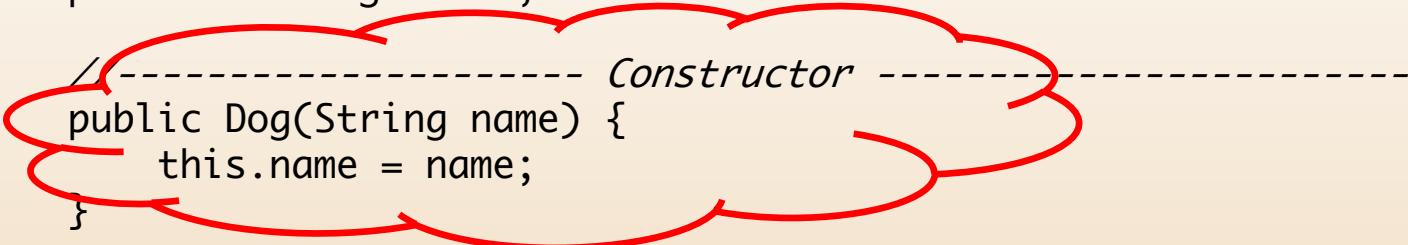
Java Class

```
public class Dog {  
    //----- All instance variables -----  
    private String name;  
    private int age;  
    private String breed;  
  
    //----- Constructor -----  
    public Dog(String name) {  
        this.name = name;  
    }  
  
    //----- All method definitions -----  
    public void bark(){  
        System.out.println("The dog " + name + " is barking");  
    }  
  
    public void wagTail(){  
        System.out.println("The dog is wagging tail");  
    }  
}
```



Create an Object

```
public class Dog {  
    //----- All instance variables -----  
    private String name;  
    private int age;  
    private String breed;  
  
    //----- Constructor -----  
    public Dog(String name) {  
        this.name = name;  
    }  
  
    //----- All method definitions -----  
    public void bark(){  
        System.out.println("The dog " + name + " is barking");  
    }  
  
    public void wagTail(){  
        System.out.println("The dog is wagging tail");  
    }  
}
```



The code is annotated with hand-drawn red circles and arrows. A large circle encloses the constructor definition, with an arrow pointing from the word 'Constructor' to its start. Another circle encloses the entire class body, with arrows pointing from the words 'All instance variables' and 'All method definitions' to their respective sections in the code.

```
Dog myDog = new Dog("Bella");
```



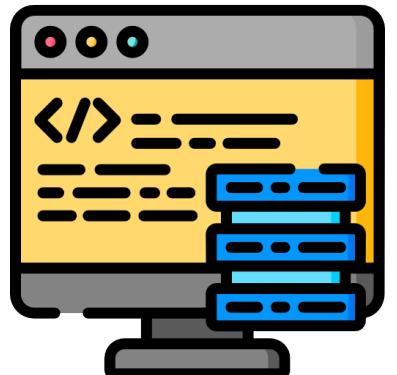
THE UNIVERSITY OF
MELBOURNE

2. Methods



Methods

- Static method
- Non-static method



Java Class

```
public class SampleClass {  
  
    public static void method1(){  
        method2();  
    }  
  
    public void method2(){  
        method1();  
    }  
}
```



Call a static method

SampleClass.method1();

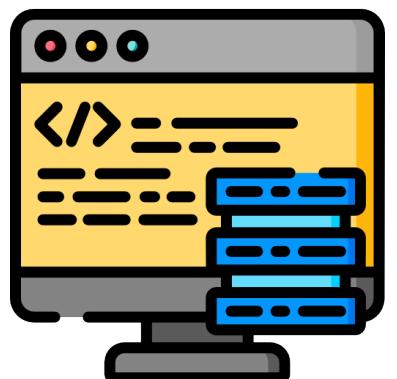
Call a non-static method

SampleClass myObject = new SampleClass();
myObject.method2;



Methods

- Return
- No return





Methods

Return a result

- `public double computeGPA(double sub1, double sub2)`

Just doing something

- `public void enrollStudent(int studentID)`

Use a returned value

```
public double computeGPA(double sub1, double sub2){  
    return System.out.println( (sub1 + sub2) / 2 );  
}
```



Callee

```
public void showGrade(){  
    double result = computeGPA(80.2, 75.6);  
    if (result >= 80){  
        System.out.println("h1");  
    }  
}
```

Caller



Overloading

Header

```
public double computeGPA(double sub1, double sub2)
```

Signature

```
public double computeGPA(int studentID)
```



Access modifier

➤ public

no restrictions on where the instance variable/method can be used

➤ private

the instance variable/method cannot be accessed by name outside of the class definition.

Access modifier

```
package somePackage;
```

```
public class A
{
    public int v1;
    protected int v2;
    int v3.//package
        //access
    private int v4;
```

```
public class B
{
    can access v1.
    can access v2.
    can access v3.
    cannot access v4.
```

```
public class C
    extends A
{
    can access v1.
    can access v2.
    can access v3.
    cannot access v4.
```

```
public class D
    extends A
{
    can access v1.
    can access v2.
    cannot access v3.
    cannot access v4.
```

```
public class E
{
    can access v1.
    cannot access v2.
    cannot access v3.
    cannot access v4.
```

In this diagram, "access" means access directly, that is, access by name.



THE UNIVERSITY OF
MELBOURNE

4. Exercise



Tutorial Exercise

1. Write a class method called `printMovies` that prints a list of your top 5 personal favourite movies. The output of this method should be in a numbered list, as follows:
 1. The Avengers
 2. Iron Man
 3. ...
2. *Overload* your `printMovies` method to accept two arguments - an integer and a `String`, which represent the rank and title of someone's favourite movie. Your method should then print the movie, with its rank, as above. For example, `printMovies(3, "Guardians of the Galaxy")` should output
 3. Guardians of the Galaxy

Revise your solution to Exercise 1 to use your method for this exercise to print out your 5 favourites.



Homework

3. Write a class method called `isAFavourite` which accepts a single `String` argument, and returns `true` if the argument is one of the movies contained in your list of favourites, and false otherwise.

Hint: Store your list as a single `String`

Hint: The `String` method `contains(arg)` returns `true` if the argument is contained in the string. For example, `"hello".contains("hell")` will return `true`.

4. **Challenge:** *Overload* your `printMovies` method again to accept a single `String` argument, where the string is a list of movies separated by commas. Your method should then print each movie on a separate line, numbered by the order you print it.

For example, `printMovies("The Avengers, Iron Man, Thor")` should output

1. The Avengers
2. Iron Man
3. Thor

Again revise your solution to Exercise 1 to use this new method for print out your 5 favourites.



Thank you





WARNING

This material has been reproduced and communicated to you by or on behalf of the University of Melbourne in accordance with section 113P of the *Copyright Act 1968 (Act)*.

The material in this communication may be subject to copyright under the Act.

Any further reproduction or communication of this material by you may be the subject of copyright protection under the Act.

Do not remove this notice