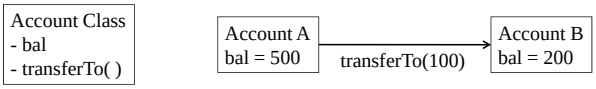
COMPS202F Java

Kelvin Lee

Basic Questions

- What is a program? Why do we write programs?
- What is
  - an object?
    - models real-world object, focus on **essential characteristics**
  - a class?
    - a class is a template to create objects of that class
  - the relationship between them?
    - class must be defined first, then create objects using the class



- What is object-oriented programming (OOP)?
- Why we learn OOP?

Advantages of Object-oriented Programming

- simplicity: software objects model real world objects
- modularity: internal workings decoupled from other parts
- modifiability: easy to make minor changes
- maintainability: objects can be maintained separately
- re-usability: objects can be reused in different programs
- source: http://eprints.ecs.soton.ac.uk/857/3/html/node3.html

Any disadvantages?

- small programs : slower to write

Class

- Need to create **class** before creating objects

```
// comment
public class ClassName {
  //attribute(s) : store value(s), if any
  //method(s)    : perform action(s), if any
}
```

- E.g. need to model mobile phone,
  - what are the values to store?
  - what actions need to be performed?

* attributes and methods can be in any order

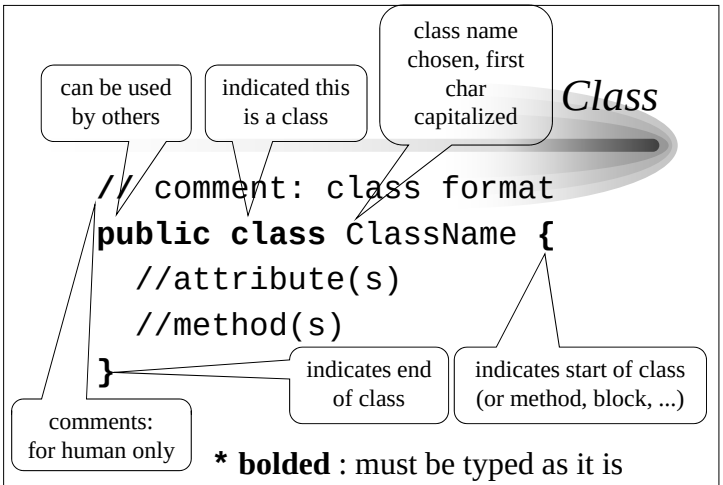
Mobile Phone Class

```
// Mobile Phone Class
public class MobilePhone {
  //attribute(s): store values
  ... emergencyNumber = 112;

  //method(s)   : perform actions
  ... dial(... phoneNumber) {
    // statement(s)
  }
}

where "..." represents something omitted
```


Class



*** bolded** : must be typed as it is

Class

- your turn: what is the meaning of each word or special character(s)?

```
// comment: class format
public class ClassName {
  //attribute(s)
  //method(s)
}
```

- now we discuss attributes

Attributes

usually private, some public attributes will be discussed later

attribute name chosen, first char lower case

```
// comments
public class Student {
  // format 1: private attType attName;
  private String name;
  private int    age;

  // format 2: private attType attName = value;
  private char   sex = 'M';
  private double height = 180;
  private boolean matureStudent = false;

  // methods omitted
  ...
}
```


Methods

??

optional word, method exists even no object created

type of object returned by method

??

```
public static returnType methodName(){
  statement(s)
}
```

??

method name chosen, first char lower case

parameter(s) can be put here

The `main()` *Method*

- the method `main()` will be executed when a class is executed using the command "**`java`** `Classname`". This method must be typed identical to the one given below (except some minor parts of the parameter)

nothing returned

a parameter: string array

```
e.g.
public static void main(String[] args) {
  System.out.println("How are you?");
}
```

prints something out

value to be printed: a string

A complete class with main() only

- the class `HowAreYou` (see below)
  - contains only one method, no attribute(s)
  - the method `main()` exists after the class is defined, no object is created
  - before run program need to compile first,: type "**`javac`** `HowAreYou.java`" in command prompt (or Ctrl-1 in Crimson editor)
  - to run, type "**`java`** `HowAreYou`" in command prompt (or Ctrl-2 in Crimson editor)

```
public class HowAreYou {
  public static void main(String[] args) {
    System.out.println("How are you?");
  }
}
```


Other methods

- a method containing `static`, such as `main()`, is special. In other words, most of the methods we will see do not contain `static`. The following are some examples:

```
// return square of a number
public double square(double number) {
  return number * number;
} //e.g. returns 9 if number is 3

// print a string twice
public void printString(String aString) {
  System.out.println(aString + aString);
} //e.g. prints "abcabc" if aString is "abc"
```