THE UNIVERSITY OF MELBOURNE

# SWEN20003 Workshop

## Week 07

**Demonstrator: Xulin Yang**

# Workshop tasks

1. Go through questions
2. Divide groups by records
3. Do questions Q1, 2, 3 together
4. Take attendance

# Interface


# What is an interface, and how is it different to inheritance?

# What is an interface, and how is it different to inheritance?

An interface is used to define the ***behaviors* or *actions*** of several classes that may not have anything else in common and represent a "can do" relationship;

- E.g.: Houses and Furniture can both be constructed, but they don't have much in common to warrant inheritance.

Inheritance is used to pass common information from one class (parent) to its subclasses (children) and represents an “is a” relationship; Dogs, Cats, and Birds are all Animals, so all share a *name*.

In my opinion:

1. interface only methods, inheritance have attributes + methods
2. Interface (can do relationship), inheritance (is a relationship)


# In what situations would we choose one (or more) interfaces over inheritance?

## Interface or Inheritance?

**Inheritance:**

- Represents *passing shared information* from a *parent* to a *child*
- Fundamentally an "Is a" relationship; a *child* **is a** *parent*, plus more; hierarchical relationship
- All `Dogs` are `Animals`

**Interface:**

- Represents the ability of a *class* to *perform an action*
- Fundamentally a "Can do" relationship; a *Comparable* object can be *compared* when sorting
- `Strings` can be compared and sorted


# In what situations would we choose one (or more) interfaces over inheritance?

Interfaces are the preferred abstraction when there are no (or few) common properties, or no logical parent-child relationship, between classes that share behaviors.

In my opinion:

1. No attributes only methods (i.e. no common property) -> interface
2. Can do relationship -> interface

Note that you can have constants defined in interface


Define an interface called Playable, that allows objects to be “played”. What objects do you think might use this interface, and why are we not using inheritance?


Define an interface called Playable, that allows objects to be “played”. What objects do you think might use this interface, and why are we not using inheritance?

```
public interface Playable {
        public void play();
}
```

Who use? Instruments, video games, music, and sports, are all examples of things you can play -> these class **implements Playable**

Why? there is no logical relationship between them, so inheritance doesn't make sense.


What is the Comparable interface?
What implementations of compareTo might we use to compare two: Students; Fruits; Players?


# What is the Comparable interface? What implementations of compareTo might we use to compare two: Students; Fruits; Players?

Implementing the Comparable interface allows a class to be ordered or sorted. There are plenty of different ways to sort a class, but here are some examples:

A class that implements `Comparable<ClassName>`

- Can (unsurprisingly) be compared with objects of the same class
- Must implement `public int compareTo(<ClassName> object)`
- Can therefore be **sorted** automatically

```
public int compareTo(Student student) {
    return this.studentID - student.studentID;
}
public int compareTo(Fruit fruit) {
    return this.colour.compareTo(fruit.colour); // Assume Colour implements Comparable
}
public int compareTo(Player player) {
    return this.winRate - player.winRate;
}
```

THE UNIVERSITY OF MELBOURNE

# Thank you