THE UNIVERSITY OF MELBOURNE

SWEN20003 Workshop

Week 06

Demonstrator: Xulin Yang

Workshop tasks

1. Go through questions
2. Divide groups by records
3. Do questions Q1, 2, 4, 5 together, Q3 individually on your own machine
4. Take attendance

Inheritance

Why does Object Oriented programming use inheritance?

Why does Object Oriented programming use inheritance?

Inheritance in Object Oriented programming is much the same as in genetics;

by using inheritance, **parent classes can pass on common information (methods and attributes) to their children**.

By using inheritance, we can define behavior that is common to several classes **in a single parent class**, and then pass that information on to children classes.

What advantages does inheritance give us as programmers?

What advantages does inheritance give us as programmers?

This means we don't need to write the same code multiple times, makes our classes simpler, and is often a better representation of important data in a problem.

1. **Reduce duplicate code**
2. **Better abstraction**

What relationship does inheritance represent?

What relationship does inheritance represent?

Inheritance represents an **“Is A” relationship**, where it makes sense to say a subclass object is a subtype of the superclass

for example, a Dog is a subtype of Animal, a Car is a subtype of Vehicle, but a Table is not a subtype of Chair.

What is the keyword used to access something in a parent class?

What is the keyword used to access something in a parent class?

We use **super** to reference an object's superclass, allowing us to access its protected or public attributes and methods.

What class do all other classes inherit from?

What class do all other classes inherit from?

All Java classes implicitly inherit from the Object class

What methods are inherited from the class Object, and why do we generally replace them?

What methods are inherited from the class Object, and why do we generally replace them?

Methods inherited: toString and equals methods.

The toString method should give a useful String representation of the object, and equals should be able to identify when objects are “identical”;

neither of the inherited methods work “correctly”, so we replace them.

Abstract Classes

What is an abstract class?

What is an abstract class?

An abstract class defines common behaviors or properties of other classes **but** doesn't have enough concrete information for it to be instantiated.

For example, we know that all Animal objects make noise, but if you are asked what noise an animal makes, there's no “correct” answer (no concrete information); --> Animal is abstract.

What is an abstract method?

What is an abstract method?

An abstract method means that a subclass will implement this method, we just don't know how;

E.g.: makeNoise is an abstract method of Animal (don’t know now), that is implemented by Dog to make it bark (know now).

How can we decide whether a class should be abstract or concrete?

How can we decide whether a class should be abstract or concrete?

Do any of these objects have common attributes?

If they both inherited from the same class, would that make sense?

| -Would it make sense to create an object of that superclass? If not, then it's **abstract**

My understanding: **Superclass shouldn’t be instantiated -> make it abstract**

Polymorphism

Define polymorphism.
In what ways does Java allow polymorphism?

Define polymorphism.
In what ways does Java allow polymorphism?

Polymorphism: the ability of an object or method to be used in different ways.

Java allows polymorphism through:

1. **Overloading** method used depends on the signature
2. **Overriding** method used depends on the class that was instantiated
3. **Substitution** subclasses taking the place of super classes
4. **Generics** defining parametrized methods/classes

THE UNIVERSITY OF MELBOURNE

Thank you