COMP90041
Programming and Software Development
2020 - Semester 2
Lab 10

Yuhao(Beacon) Song

yuhsong1@unimelb.edu.au

yuhsong@student.unimelb.edu.au

Introduction

- **Timetable**
  - Tue(18) 16:15-17:15 (Melbourne Time)
    - https://unimelb.zoom.us/j/94854648719?pwd=WUY0NmR6MkI5UVZBUWhGNWFIU216Zz09
  - Wed(19) 17:15-18:15 (Melbourne Time)
    - https://unimelb.zoom.us/j/93723434766?pwd=MGh4QkJuZnhJR21LZ0VqeXhJQU52UT09

- **GitHub Page (Tutorial Materials, Solutions, Additional Resources)**
  - https://github.com/Beaconsyh08/COMP90041-2020SEM2

- **PollEv**
  - https://pollev.com/yuhsong

Outline

- Assignment 2
- Lecture Review
- Topic to discuss next week
- Survey

Assignment 2 Feedback

- You could expect the feedback will be returned around SWOT vac (Nov 2nd)

- **Fun fact - SWOT vac:**
  - Study Week Or Take VACation
  - Study WithOut Teaching (or Tuition) VACation

Abstract Method

- An **abstract method** has a heading, but **no** method body(no implementation)
  - public **abstract** int sumOfTwo(int n1, int n2);
  - public **abstract** int sumOfThree(int n1, int n2, int n3);

- The body of the method is defined in the **derived classes**
  - public int sumOfTwo(int num1, int num2){return num1+num2;}
  - public int sumOfThree(int num1, int num2, int num3){return num1+num2+num3;}

- An **abstract method** is like a **placeholder** for a method that will be fully defined in a **descendent class**. It has a complete method heading with modifier **abstract**

Abstract Class

- If a **class** has an abstract method it should be declared **abstract**
- If a **regular class** extends an **abstract class**, then the class must have to implement **all** the **abstract methods** of abstract parent class or it has to be declared **abstract** as well.
- A class that has **no abstract methods** is called a **concrete class**
- abstract class and abstract method: **Cannot be private** (abstract class could not be protected as well)

You Cannot Create Instances of an Abstract Class

- An **abstract class** can only be used to **derive** more specialized classes
  - While it may be useful to discuss employees in general, in reality an employee must be a salaried worker or an hourly worker (Example from lecture)

- An **abstract class constructor** cannot be used to create an **object** of the abstract class
  - However, a derived class constructor will include an invocation of the abstract class constructor in the form of super
  - The constructor in an abstract class is only used by the constructor of its derived classes

Interfaces

- An **interface** is something like an **extreme case** of an **abstract class**, but not a class
- The syntax for defining an **interface** is use word interface

- It contains **method headings** and **constant definitions only**, and **no instance variables** nor any **complete method definitions**

- An **interface** and **all of** its **method headings** should be declared **public**

- When a class implements an interface, it must implement **all the methods** in the interface
- Any variables defined in an interface must be **public, static, and final**. Because this is understood, Java allows these modifiers to be **omitted**

Implementing Interfaces

- The classes may **implement** one or more interfaces, If more than one interface is implemented, each is listed, separated by commas
- It must include the phrase *implements Interface_Name* at the start of the class definition, if the class extends another class, **implements** go **after extends**.

- A **concrete class** must give definitions for **all** the **method headings** given in the abstract class and the interface
- A **abstract class** could also implements interface. Any method headings given in the interface that are not given definitions are automatically made into abstract methods

Derived Interfaces

- Like classes, an **interface** may be derived from a base interface
  - This is called **extending** the **interface**
  - The **derived interface** must include the phrase ***extends*** *BaseInterfaceName*

- A **concrete class** that implements a derived interface must have definitions for **any methods** in the derived interface as well as any methods in the base interface

Abstract class vs Interface

	Abstract class	Interface
Type of methods	abstract and non-abstract	only abstract
Final Variables	final or non-final	by default final
Type of variables	final, non-final, static and non-static	only static and final
Implementation	can provide the implementation of interface	can't provide the implementation of abstract class.
Inheritance vs Abstraction	extends	implements
Multiple implementation	extend another Java class and implement multiple Java interfaces	extend another Java interface only
Accessibility of Data Members	private, protected, etc	public by default

Final Project

- Read through the specification
- If you find any doubts, read a more carefully of the **specification**
- The first choice is a search on the **discussion board**, if not been answered, post on it. That will help your peers who have the same doubts. (avoid same questions been answered many times, but through email only)
- If you have more specific doubts, you could email us

What shall we discuss next week?

- The Final Project
- other peoples approach to the actual ethics of the final project
- how to attack the basic elements of the final project.
- collections

Survey

- https://forms.gle/AFGHbBigvTnwUh6q7

- No Official survey from Unimelb this year due to the pandemic
- This is created by myself, fill it if you willing to (not compulsory)
- I will remind you one more time next week :)