Day 1. Handler

Singleton

* Lazy loading
* Eager loading

//lazy loading

public class Singleton{

private static Singleton instance;

private Singleton(){

}

public static Singleton getInstance(){

if (instance==null){

instance=new Singleton;;

}

Return instance

}

}

Access Modifier: public, private, default\*, protected\*考點

* default is the default, gives package level accessibility for **field, method, class**
* protected=default+sub-class(gives parent/child access)
* private is only available in class

What is Class?

Class is a **template**

\* object/instance vs template ->objects should declare by **new** keyword

\* other templates include **Interface**, **Abstract Class**, (not a template but a annotation) **enum**

Static keyword

* It’s a method
* Access elements from template
* JVM calls it, and only executed once when loading, compile before run

\* JVM’s structure –> contains **stack, heap, PC register, method area, native method area**

* Static are sorted in method area

Final keyword

* It’s a modifier, can be apply on **field**, **method**, **class**
* For field: the value can only be assign once, means reference cannot be change, but the object itself can still be update
* For method: can not be override by sub-class
* For class: can not be extend, often used for Immutable class

\* Immutable:

* declare final
* declare private
* only use getters, no setters
* actively do **deep copy**

Override vs Overload

* both concepts of polymorphism

Run time <--> compile time

In between classes(parent child) <--> within the same class

Factory Design Pattern

* Scenario e-commerce is using 2 payment method, say paypal and cc
* Double vs double:

**double** is primitive, Double is referenced data type

\* 8 primitive byte, short, int, long, float, double, boolean, char

Java uses wrapper class and autoboxing(Double) to make primitive behave like objects, to achieve OOP

Exception Handling

* Throw vs Throws

Use in the function <-> use in the function signature

* “try-catch-finally”
* Types of exceptions: runtime, compile time
* All errors came from **throwable** class unless handled by try-catch
* Exceptions vs Error

Can be handle in code <->can not be handle in code(ex: StackOverFlow or OutOfMemory)

Collections: Lists

* ArrayList vs LinkedList

Access by index <-> Had to iterate through