Java Programming
OOP - Wrapping Up

The "static" Variables and Methods

- A static variable/method belongs to the class, and is shared by all instances.
- It is also called a class variable/method.
- The JVM allocates static variable during the class loading.
- The static variable exists even if no instance is created and regardless of the number of instances created.
- UML Notation: static variables/methods are underlined in the class diagram.

"final" Class/Variable/Method

- A final class cannot be sub-classed (or extended).
- A final method cannot be overridden in the subclass.
- A final variable cannot be re-assigned a new value.
- A "public final static" variable of primitive type is a global constant, whose value cannot be changed.
- A final variable of a reference type cannot be re-assigned a new value (reference).

Method Overloading vs. Overriding

An overriding method:

- must have the same parameter list as it original.
- must have the same return-type.
- cannot have more restrictive access modifier than its original.
- overriding a private method does not make sense, as private methods are not really inherited by its subclasses.

Method Overloading vs. Overriding

- You cannot override a non-static method as static, and vice versa.
- Technically, a subclass does not override a static method, but merely hides it. Both the superclass' and subclass' versions can still be accessed via the classnames.
- A final method cannot be overridden.

Method Overloading vs. Overriding

An overriding method:

- must be differentiated by its parameter list.
- it shall not be differentiated by return-type.
- It could have any return-type, exception list or access modifier, as long as it has a different parameter list than the others.

Reference

- https://www3.ntu.edu.sg/home/ehchua/programming/java/J3c_OOPWrappingUp.html