Chen Wang Assignment 12/12

Q1. What is the difference between a class and an object? Provide a real-world example.

- Class is designed as a blueprint for a purpose, while an object is the instance of the class. For example, we could see the library ID card as a class, and the cards hosted by concrete people can be seen as objects.

Q2. If you create a parameterized constructor in a class, what happens to the default constructor? What must you do if you still need it?

- If create a parameterized constructor, then the default won't be provided automatically. If we still need a non-parameter constructor, we need to write it ourselves.

Q3. What are the four access modifiers in Java? List them from most restrictive to least restrictive.

- Private: most restrictive one, can be accessed only within the same scope, like in the class.
- Default: Can be accessed within the same package, no keyword needed.
- Protected: can be accessed within the same package or the subclasses.
- Public: can be accessed anywhere in the project.

Q4. Explain the difference between method overloading and method overriding.

- Method overloading means a method has the same name with another method, but different number / type / order of parameters. It is static polymorphism, it is decided at compile time, and happens in the same class.
- Method overriding means an inherited method in a subclass modifies its behavior from its superclass. It is dynamic polymorphism, and is decided at run time.

Q5. Can you override a final **method? Can you override a** private **method? Explain why or why not.**

- We can't override a final method because the final keyword means nobody can modify the method, so overriding is impossible.
- We cannot override a private method as well, because we don't even have access to that method in the subclass, so overriding is impossible.

Q6. What is the difference between static polymorphism and dynamic polymorphism? When does each occur?

- static polymorphism means method overloading while dynamic polymorphism means method overriding.
- This question was answered in Q4.

Q7. Why does Java not support multiple inheritance with classes? How can you achieve multiple inheritance in Java?

- Java doesn't support multiple inheritance because it will avoid the diamond problem. If we have class A, and class B/C inherited A, then class D inherited B and C, and B/C both overrided a method from A. Then when class D is trying to call this method, D will have issues to determine which method from B/C to call.
- Also this will avoid the increase in complexity.

- To achieve multiple inheritance, we could use an interface in Java. interface is a blueprint of a purpose, it only have name and type of properties and methods, but no implementation.

Q8. Consider the following code:

```
List<Integer> lst = new ArrayList<>();
```

Which principle of OOP does this demonstrate? Explain.

- Polymorphism.
- Java knows that the variable type of lst is List, while the type of the instance is ArrayList.