Exercise 3:

a. Can an instance method override a static method?
No, an instance method cannot override a static method. Static methods belong to the class and are not associated with an instance, so they cannot be overridden but can be hidden.

b. Can a static method override (hide) an instance method?
No, a static method cannot override an instance method. Overriding applies only to instance methods. However, a static method with the same name in a subclass can hide an instance method in the superclass, which leads to method hiding.

c. Can you override a final instance method?
No, a final instance method cannot be overridden. The final modifier prevents method overriding to ensure the method's behavior remains unchanged in subclasses.

d. Can you override an instance method and make it final?
Yes, you can override an instance method and declare it as final in the subclass. This prevents further overriding in subsequent subclasses.

e. Can you override an instance method and change its return type?
Yes, you can override an instance method and change its return type, but only if the return type in the subclass method is a covariant of the return type in the superclass method (i.e., the subclass return type must be a subclass of the superclass return type).

f. Can you hide a final static method?
Yes, you can hide a final static method. Method hiding applies to static methods, and the final modifier doesn't prevent a subclass from defining a static method with the same name and signature.

g. Can an instance field hide a static field?
Yes, an instance field can hide a static field in the superclass. However, field hiding does not work like method overriding or hiding. Access to the hidden field depends on the type of reference used.

h. Can a static field hide an instance field?
Yes, a static field can hide an instance field in the superclass. Similar to instance field hiding, access depends on the reference type.

i. Can an instance method with public visibility override an instance method with default visibility?
Yes, an instance method with public visibility can override an instance method with default visibility because public has a broader access scope than default.

j. Can an instance method with default visibility override an instance method with public visibility?
No, an instance method with default visibility cannot override a public method. Overriding rules require the subclass method to have at least the same or broader access scope.

k. Can an instance method with protected visibility override an instance method with default visibility?
Yes, a protected method can override a default method because protected has a broader scope than default.

l. Can an instance method with default visibility override an instance method with protected visibility?
No, an instance method with default visibility cannot override a protected method because it has a narrower access scope.

m. Based on the last four questions, order the access visibility from the widest to the narrowest (weakest):

1. Public
2. Protected
3. Default (Package-private)
4. Private

Rule for overriding (instance methods) or hiding (static methods):

- For instance methods, the overriding method in the subclass must have the same or a broader access modifier than the overridden method in the superclass.
- For static methods, the hiding method does not need to follow access modifier rules, but the method in the subclass is not truly overriding—it is hiding the superclass method.