20251112 Wednesday

- 1 HW
- 2 current grade
  The percentage number on canvas means nothing.
- 3 final exam (Test 3)
  No more online test.
  ONLY Paper + Pen/Pencil
- 4 NO extra/bonus/redo
- 5 Grade/GPA is not important as for other majors.

Review Abstract Class/Method

- Abstract Art
- Abstract Method
- Abstract Class

True or False

- A class with abstract method
  does NOT need to be an abstract class
- An object can be instantiated directly from abstract class
- Abstract class must have abstract method inside
- Superclass of abstract class must be abstract class
- Abstract class can NOT be a type for objects
- Concrete methods can NOT be overridden as abstract

Interfaces

What is an interface?

Why is an interface useful?

How do you define an interface?

How do you use an interface?

- Java does NOT support "multiple inheritance" (a class can only inherit from one superclass).
- However, it can be achieved with interfaces, because the class can **implement** multiple interfaces.

What Is an Interface? Why Is an Interface Useful?

An interface is a classlike construct that contains only constants and abstract methods.

In many ways, an interface is similar to an abstract class, but the intent of an interface is to specify common behavior for objects.

For example, you can specify that the objects are comparable, edible, cloneable using appropriate interfaces.

- Interface methods are by default abstract and public
- Interface attributes are by default public, static and final

Define an Interface

To distinguish an interface from a class, Java uses the following syntax to define an interface:

```
public interface InterfaceName {
  constant declarations;
  abstract method signatures;
}
```

Example:

```
public interface Edible {
  /** Describe how to eat */
  public abstract String howToEat();
}
```

Interface Is a Special Class

An interface is treated like a special class in Java.

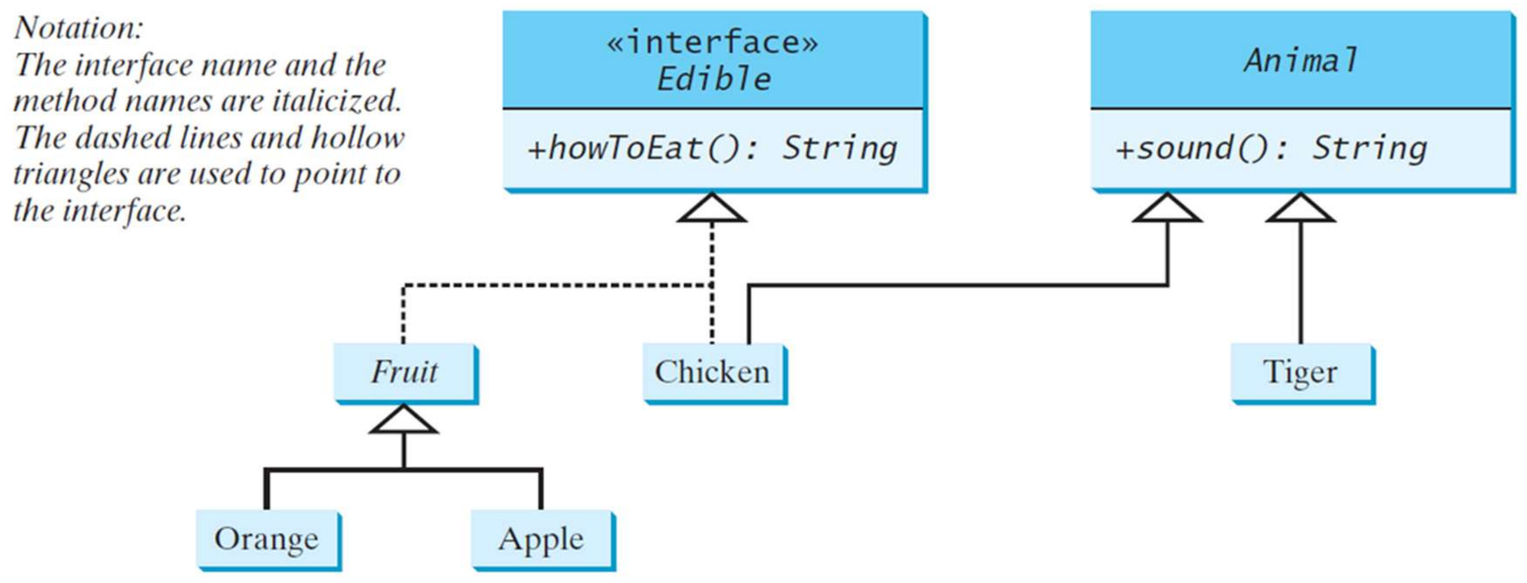
Each interface is compiled into a separate bytecode file, just like a regular class.

Like an abstract class, you cannot create an instance from an interface using the new operator, but in most case, you can use an interface the same way you use an abstract class.

For example, you can use an interface as a data type for a variable, as the result of casting, and so on.

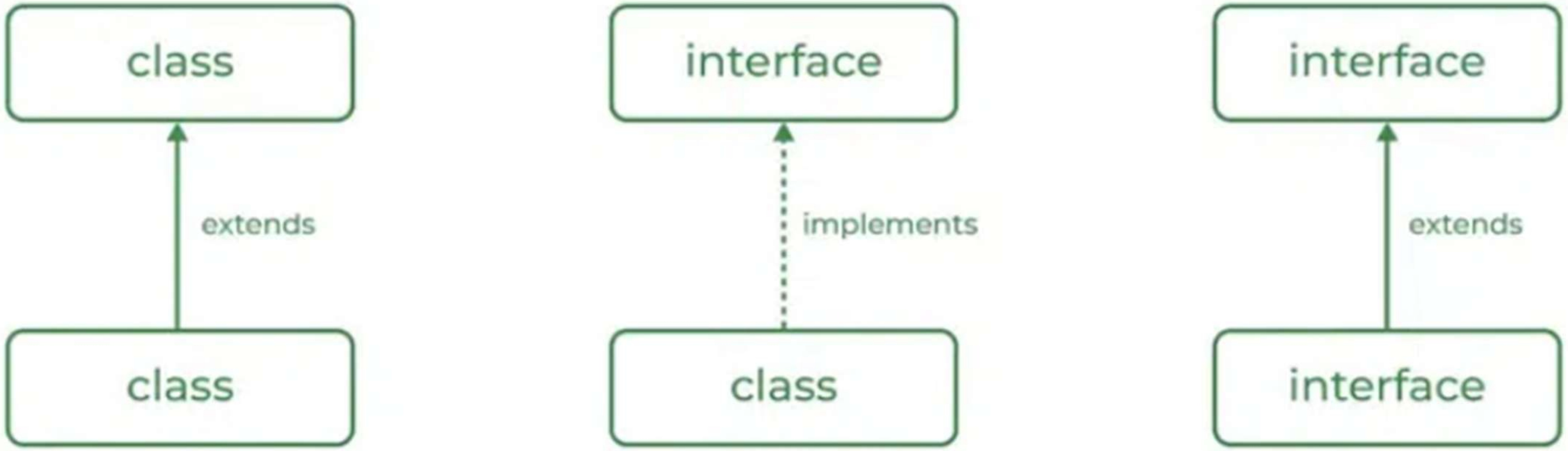
Example (1 of 2)

You can now use the Edible interface to specify whether an object is edible. This is accomplished by letting the class for the object implement this interface using the implements keyword. For example, the classes Chicken and Fruit implement the Edible interface (See TestEdible).

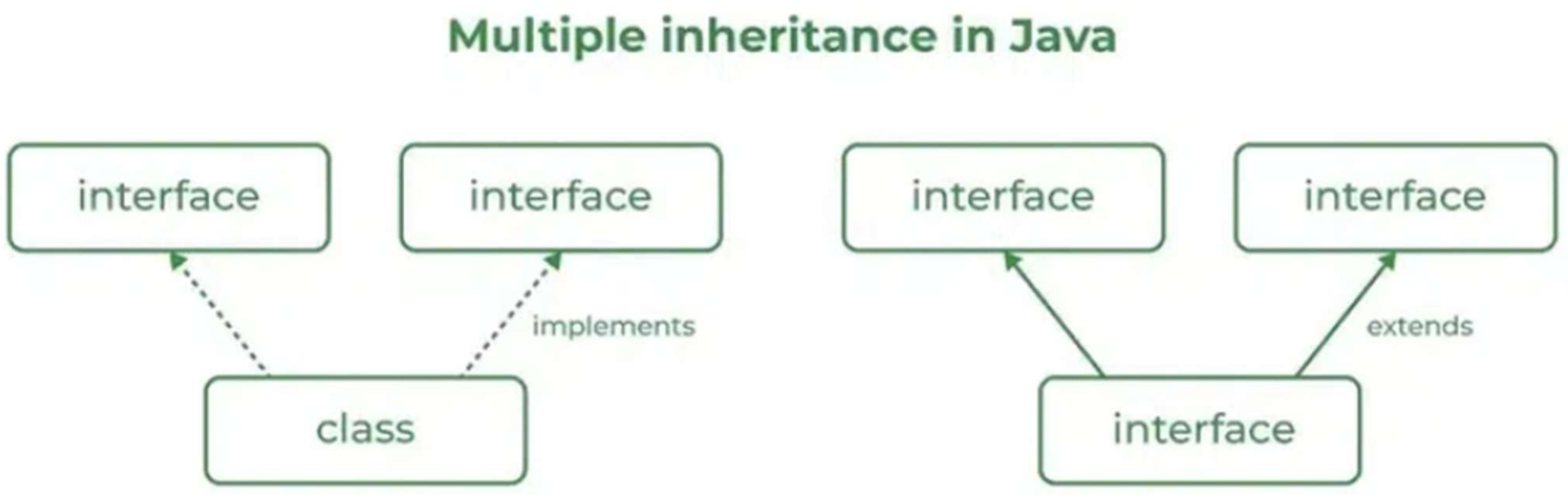


Notation:
The interface name and the method names are italicized. The dashed lines and hollow triangles are used to point to the interface.
«interface»
Edible
+howToEat(): String
Animal
+sound(): String
Fruit
Chicken
Tiger
Orange
Apple

Edible TestEdible



- geeksforgeeks.org



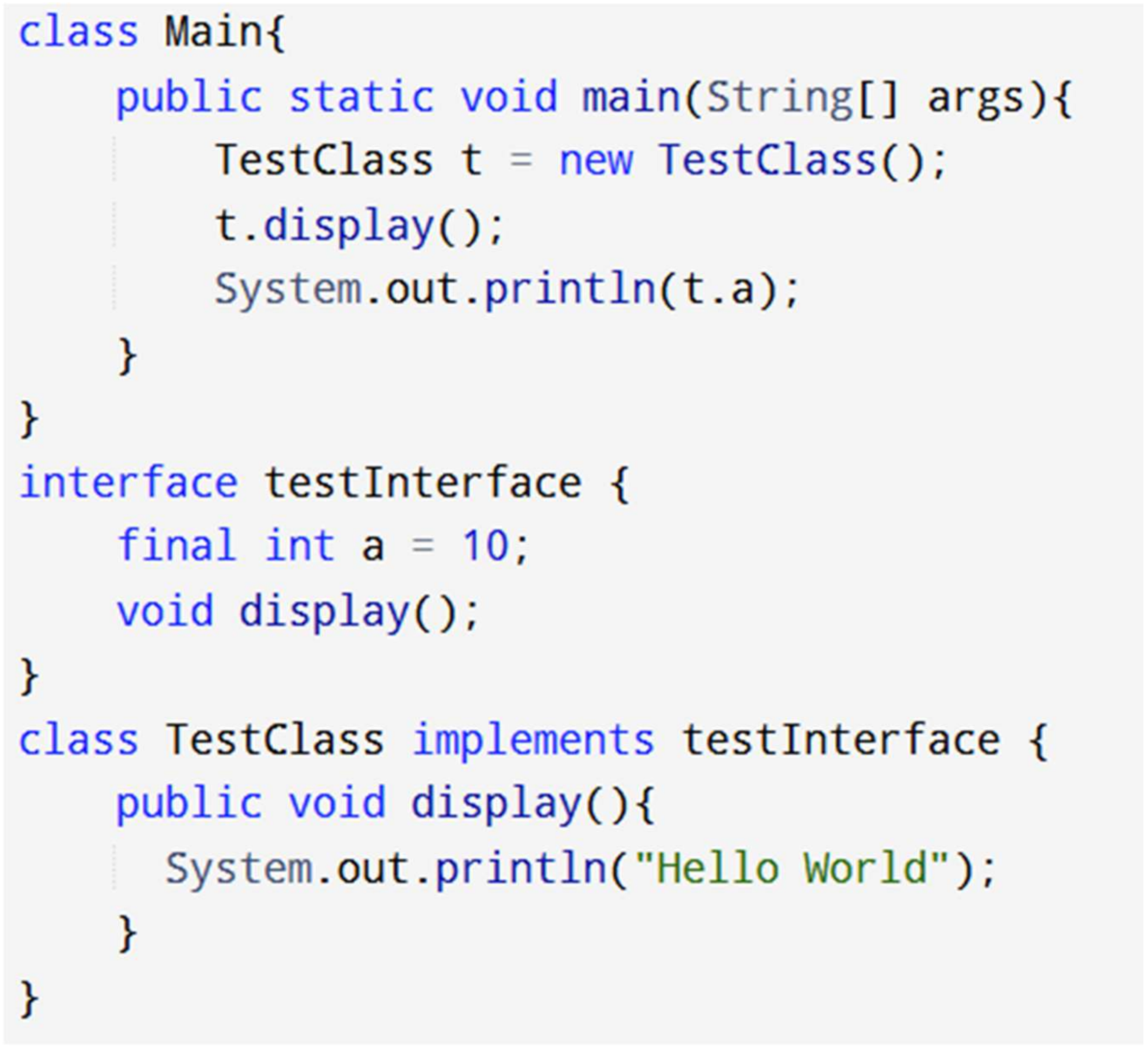
- Geeksforgeeks.org

Practice online

Programiz

Online Java Compiler

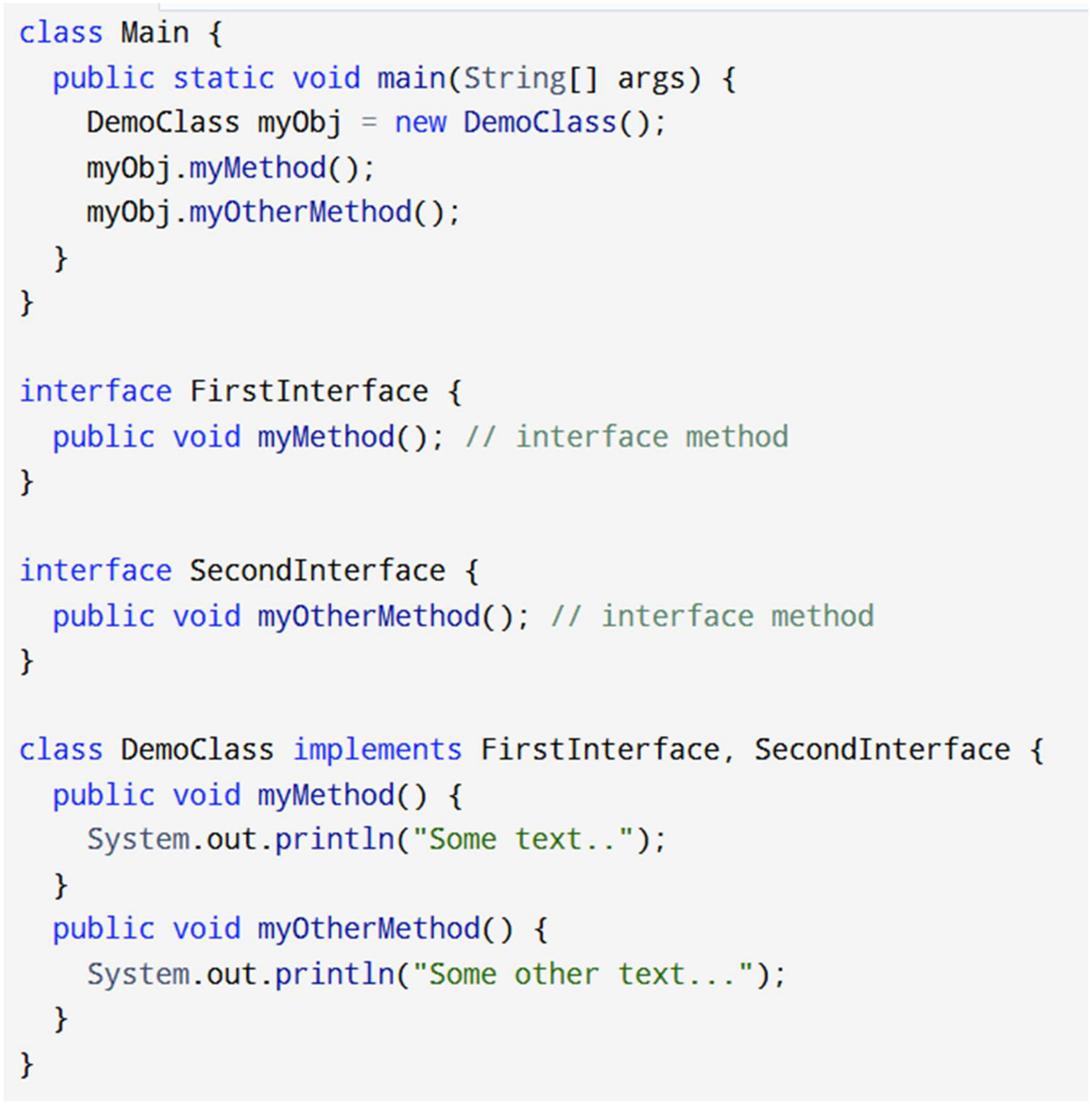
Practice



```java
class Main{
    public static void main(String[] args){
        TestClass t = new TestClass();
        t.display();
        System.out.println(t.a);
    }
}
interface testInterface {
    final int a = 10;
    void display();
}
class TestClass implements testInterface {
    public void display(){
      System.out.println("Hello World");
    }
}
```

- 1 Try create object based on interface
- 2 cast an object into interface type
- 3 define another interface based on one interface?

Practice



```java
class Main {
  public static void main(String[] args) {
    DemoClass myObj = new DemoClass();
    myObj.myMethod();
    myObj.myOtherMethod();
  }
}

interface FirstInterface {
  public void myMethod(); // interface method
}

interface SecondInterface {
  public void myOtherMethod(); // interface method
}

class DemoClass implements FirstInterface, SecondInterface {
  public void myMethod() {
    System.out.println("Some text..");
  }
  public void myOtherMethod() {
    System.out.println("Some other text...");
  }
}
```

- Practice, Practice, and more Practice.
- Next class: Chapter 14 “GUI”
- HW