OOP in Java Review

CS356 Object-Oriented Design and Programming
http://cs356.yusun.io
October 1, 2014

Yu Sun, Ph.D.
http://yusun.io
yusun@csupomona.edu

CAL POLY POMONA

Announcement

- Submit your GitHub username as soon as possible

Important

- The basic Java OOP features will be used through the whole course

- These features are frequently asked in tech interviews

Interface

- An *interface* in the Java programming language is an abstract type that is used to specify an interface (in the generic sense of the term) that classes must implement

Web iClicker

- http://answer.yusun.io
- Test Question:
- What grade do you think you can get from this course?

Interface – Question 1

- Which of the following variable declarations are correct?

```
public interface TestInterface1 {
  A     int var1;
  B     final int var2;
  C     public int var3 = 100;
  D     private int var4 = 100;
  E     public static final int var5 = 100;
}
```

Interface – Question 1

- Which of the following variable declarations are correct?

```
public interface TestInterface1 {
❌      int var1;

        final int var2;

        public int var3 = 100;

        private int var4 = 100;

        public static final int var5 = 100;

}
```

Only constants can be declared in an interface

Interface – Question 1

- Which of the following variable declarations are correct?

```
public interface TestInterface1 {

        int var1;

        final int var2;

        public int var3 = 100;

        private int var4 = 100;

        public static final int var5 = 100;

}
```

final requires an initial value to make a variable as constant.

Interface – Question 1

- Which of the following variable declarations are correct?

```
public interface TestInterface1 {

        int var1;

        final int var2;

    ✓   public int var3 = 100;

        private int var4 = 100;

        public static final int var5 = 100;

}
```

Interface – Question 1

- Which of the following variable declarations are correct?

```
public interface TestInterface1 {

        int var1;

        final int var2;

        public int var3 = 100;

        private int var4 = 100;

        public static final int var5 = 100;

}
```

Interface only contains public declarations, even without public keyword.

Interface – Question 1

- Which of the following variable declarations are correct?

```
public interface TestInterface1 {

      int var1;

      final int var2;

      public int var3 = 100;

      private int var4 = 100;

   ✓  public static final int var5 = 100;

}
```

Interface – Question 2

- What’s wrong with the following interface?

```
public interface TestInterface2 {

        void aMethod(int aValue) {
                System.out.println("Hello World!");
        }

}
```

Interface – Question 2

- What’s wrong with the following interface?

```
public interface TestInterface2 {

        void aMethod(int aValue) {
                System.out.println("Hello World!");
        }

}
```

Interface contains method signatures, not implementations.

Interface – Question 2

- How to fix it?

```
public interface TestInterface2 {

A       void aMethodFix0(int aValue);

B       protected aMethodFix1(int aValue);

C       abstract void aMethodFix2(int aValue);

D       static void aMethodFix3(int aValue) {
                System.out.println("Hello World!");
        }

E       default void aMethodFix4(int aValue) {
                System.out.println("Hello World!");
        }
}
```

Interface – Question 2

- How to fix it?

```
public interface TestInterface2 {

✓      void aMethodFix0(int aValue);

       protected aMethodFix1(int aValue);

       abstract void aMethodFix2(int aValue);

       static void aMethodFix3(int aValue) {
              System.out.println("Hello World!");
       }

       default void aMethodFix4(int aValue) {
              System.out.println("Hello World!");
       }
}
```

Interface – Question 2

- How to fix it?

```
public interface TestInterface2 {

        void aMethodFix0(int aValue);

        protected aMethodFix1(int aValue);

        abstract void aMethodFix2(int aValue);

        static void aMethodFix3(int aValue) {
                System.out.println("Hello World!");
        }

        default void aMethodFix4(int aValue) {
                System.out.println("Hello World!");
        }
}
```

Interface only contains public methods, even without the public keyword.

Interface – Question 2

- How to fix it?

```
public interface TestInterface2 {

    void aMethodFix0(int aValue);

    protected aMethodFix1(int aValue);

✓   abstract void aMethodFix2(int aValue);

    static void aMethodFix3(int aValue) {
        System.out.println("Hello World!");
    }

    default void aMethodFix4(int aValue) {
        System.out.println("Hello World!");
    }
}
```

Interface – Question 2

- How to fix it?

```
public interface TestInterface2 {

        void aMethodFix0(int aValue);

        protected aMethodFix1(int aValue);

        abstract void aMethodFix2(int aValue);

  ✓     static void aMethodFix3(int aValue) {
                System.out.println("Hello World!");
        }

  ✓     default void aMethodFix4(int aValue) {
                System.out.println("Hello World!");
        }
}
```

Start from Java 8, interface can contain static and default method bodies.

Interface – Question 3

- Is the following interface valid?

```
public interface TestInterface3 {

}
```

A. **Valid**
B. **Invalid**

Interface – Question 3

- Is the following interface valid?

✓

```
public interface TestInterface3 {

}
```

Empty interface is often used as a class marker. For instance,

java.io.Serializable

java.lang.Cloneable

Abstract Class

- Abstract classes are similar to interfaces. You cannot instantiate them.
- They may contain a mix of methods declared with or without an implementation.

Mark Rothko – "No. 13 (White, Red on Yellow)" – Oil and Acrylic on canvas -1958. "It was with the utmost reluctance that I found the figure could not serve my purposes. But a time came when none of us could use the figure without mutilating it."

Abstract Class vs Interface

- Use Abstract Class or Interface?
  - *Choose the ones that should use Abstract Class*

A. You want to share code among several closely related classes.
B. You want to take advantage of multiple inheritance of type.
C. You expect that classes that implement/extend your __ have many common methods or fields, or require access modifiers other than public (such as protected and private).
D. You want to declare non-static or non-final fields. This enables you to define methods that can access and modify the state of the object to which they belong.
E. You want to specify the behavior of a particular data type, but not concerned about who implements its behavior.
F. You expect that unrelated classes would implement/extends your __.

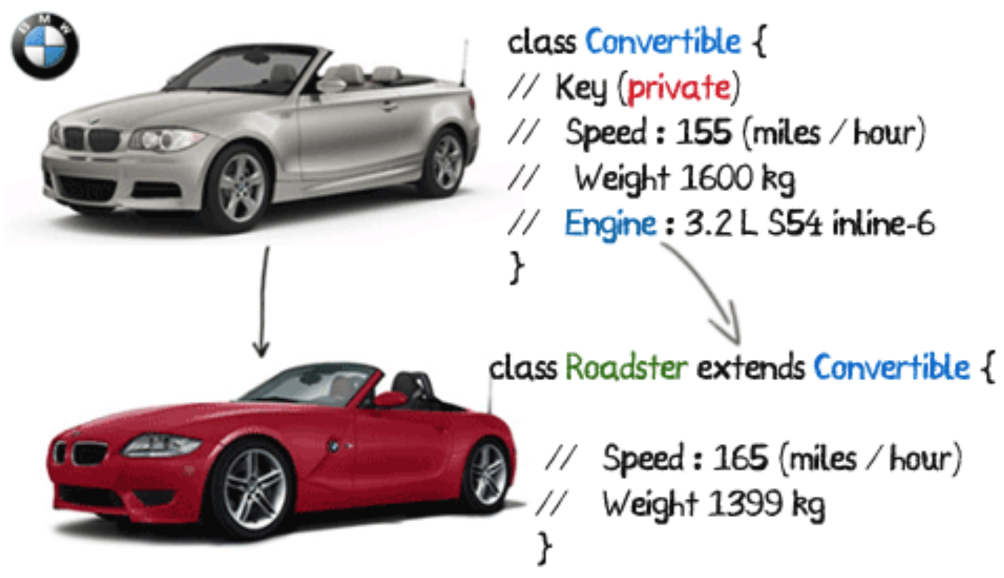
Abstract Class vs Interface

- Use Abstract Class or Interface?

A. You want to share code among several closely related classes.
B. You want to take advantage of multiple inheritance of type.
C. You expect that classes that implement/extend your __ have many common methods or fields, or require access modifiers other than public (such as protected and private).
D. You want to declare non-static or non-final fields. This enables you to define methods that can access and modify the state of the object to which they belong.
E. You want to specify the behavior of a particular data type, but not concerned about who implements its behavior.
F. You expect that unrelated classes would implement/extends your __.

Inheritance

- A class that is derived from another class is called a subclass (also a derived class, extended class, or child class).
- The class from which the subclass is derived is called a superclass (also a base class or a parent class).



```
class Convertible {
// Key (private)
//  Speed : 155 (miles / hour)
//  Weight 1600 kg
//  Engine : 3.2 L S54 inline-6
}
class Roadster extends Convertible {

  //  Speed : 165 (miles / hour)
  //  Weight 1399 kg
  }
```

Inheritance – Question 1

- Which of the the following are valid class/interface definitions?

```
interface InterfaceA { }
interface InterfaceB { }
interface InterfaceC { }

class ClassA { }
class ClassB { }
class ClassC { }
```

```
A. class TC1 implements InterfaceA, InterfaceB { }
B. class TC2 extends InterfaceA { }
C. class TC3 extends Class B, ClassC { }
D. class TC4 extends ClassB implements InterfaceA, InterfaceB { }
E. class TC5 extends Class B, ClassC implements InterfaceA, InterfaceB {
F. class TC6 implements InterfaceA, InterfaceB extends ClassB { }

G. interface TI1 implements InterfaceA, InterfaceB { }
H. interface TI2 extends InterfaceA, InterfaceB { }
I. interface Ti3 extends ClassB implements InterfaceA, InterfaceB { }
```

Inheritance – Question 1

- Which of the the following are valid class/interface definitions?

```
interface InterfaceA { }
interface InterfaceB { }
interface InterfaceC { }

class ClassA { }
class ClassB { }
class ClassC { }
```

```
A. class TC1 implements InterfaceA, InterfaceB { } ✓
B. class TC2 extends InterfaceA { }
C. class TC3 extends Class B, ClassC { }
D. class TC4 extends ClassB implements InterfaceA, InterfaceB { } ✓
E. class TC5 extends Class B, ClassC implements InterfaceA, InterfaceB {
F. class TC6 implements InterfaceA, InterfaceB extends ClassB { }

G. interface TI1 implements InterfaceA, InterfaceB { }
H. interface TI2 extends InterfaceA, InterfaceB { } ✓
I. interface Ti3 extends ClassB implements InterfaceA, InterfaceB { }
```

Inheritance – Question 2

- Which method overrides a method in the superclass?

```
class ClassParent {
        public void method1(int i) { }
        public void method2(int i) { }
        public static void method3(int i) { }
        public static void method4(int i) { }
        public void method5(int i) { }
        protected void method6(int i) { }
}
class ClassChild extends ClassParent {
A.      public static void method1(int i) { }
B.      public void method2(int i) { }
C.      public void method3(int i) { }
D.      public static void method4(int i) { }
E.      protected void method5(int i) { }
F.      public void method6(int i) { }
}
```

Inheritance – Question 2

- Which method overrides a method in the superclass?

```
class ClassParent {
        public void method1(int i) { }
        public void method2(int i) { }
        public static void method3(int i) { }
           blic static void method4(int i) { }
           blic void method5(int i) { }
        p otected void method6(int i) { }
}
clas  ClassChild extends ClassParent {
        public static void method1(int i) { }
        public void method2(int i) { }
        public void method3(int i) { }
        public static void method4(int i) { }
        protected void method5(int i) { }
        public void method6(int i) { }
}
```

Non-static method cannot be override with static.

Inheritance – Question 2

- Which method overrides a method in the superclass?

```
class ClassParent {
        public void method1(int i) { }
        public void method2(int i) { }
        public static void method3(int i) { }
        public static void method4(int i) { }
        public void method5(int i) { }
        protected void method6(int i) { }
}
class ClassChild extends ClassParent {
        public static void method1(int i) { }
     ✓  public void method2(int i) { }
        public void method3(int i) { }
        public static void method4(int i) { }
        protected void method5(int i) { }
        public void method6(int i) { }
}
```

Inheritance – Question 2

- Which method overrides a method in the superclass?

```
class ClassParent {
    public void method1(int i) { }
    public void method2(int i) { }
    public static void method3(int i) { }
    public static void method4(int i) { }
      blic void method5(int i) { }
     otected void method6(int i) { }

cl      ClassChild extends ClassParent {
    public static void method1(int i) { }
    public void method2(int i) { }
    public void method3(int i) { } ❌
    public static void method4(int i) { }
    protected void method5(int i) { }
    public void method6(int i) { }
}
```

Static method cannot be override with non-static.

Inheritance – Question 2

- Which method overrides a method in the superclass?

```
class ClassParent {
      public void method1(int i) { }
      public void method2(int i) { }
      public static void method3(int i) { }
      public static void method4(int i) { }
      public void method5(int i) { }
        otected void method6(int i) { }

      ClassChild extends ClassParent {
      public static void method1(int i) { }
      public void method2(int i) { }
      public void method3(int i) { }
      public static void method4(int i) { }
      protected void method5(int i) { }
      public void method6(int i) { }
}
```

Static method can hide the parent static method, not override.

Inheritance – Question 2

- Which method overrides a method in the superclass?

```
class ClassParent {
    public void method1(int i) { }
    public void method2(int i) { }
    public static void method3(int i) { }
    public static void method4(int i) { }
    public void method5(int i) { }
    protected void method6(int i) { }

      ssChild extends ClassParent {
    public static void method1(int i) { }
    public void method2(int i) { }
    public void method3(int i) { }
    public static void method4(int i) { }
    protected void method5(int i) { }
    public void method6(int i) { }
}
```

Override cannot reduce the visibility.

Inheritance – Question 2

- Which method overrides a method in the superclass?

```
class ClassParent {
        public void method1(int i) { }
        public void method2(int i) { }
        public static void method3(int i) { }
        public static void method4(int i) { }
        public void method5(int i) { }
        protected void method6(int i) { }
}
```

Override can increase the visibility.

```
    ssChild extends ClassParent {
        public static void method1(int i) { }
        public void method2(int i) { }
        public void method3(int i) { }
        public static void method4(int i) { }
        protected void method5(int i) { }
        public void method6(int i) { } ✓
}
```

Polymorphism

- Subclasses of a class can define their own unique behaviors and yet share some of the same functionality of the parent class

"Now Speak!"
"Quack"
"Woof"
"Meow"

Polymorphism - Basics

```
class Animal {
    public void makeNoise() {
        System.out.println("Some sound");
    }
}

class Dog extends Animal{
    public void makeNoise() {
        System.out.println("Bark");
    }
}

class Cat extends Animal{
    public void makeNoise() {
        System.out.println("Meawoo");
    }
}
```

- What's the output of the following piece of code?

```
Animal a1 = new Cat();
a1.makeNoise();
Animal a2 = new Dog();
a2.makeNoise();
```

Polymorphism - Basics

```
class Animal {
    public void makeNoise() {
        System.out.println("Some sound");
    }
}

class Dog extends Animal{
    public void makeNoise() {
        System.out.println("Bark");
    }
}

class Cat extends Animal{
    public void makeNoise() {
        System.out.println("Meawoo");
    }
}
```

- What's the output of the following piece of code?

```
Animal a1 = new Cat();
a1.makeNoise();
Animal a2 = new Dog();
a2.makeNoise();
```

```
Meawoo
Bark
```

Polymorphism – Tricky Question

```java
abstract class A {
        void test(A a) {
                System.out.println("You are in A");
        }
}
class B extends A {
        void test(B b) {
                System.out.println("You are in B");
        }
}
public class TrickyPoly {
        public static void main(String[] args) {
                A a1 = new B();
                A a2 = new B();
                B b1 = new B();
                a1.test(a2);
                b1.test(a2);
                a1.test(b1);
                b1.test(b1);
        }
}
```