The "type" of foo1 and foo2 is the same.

Consider the following class hierarchy:

 Interface Animal {...}
 class Dog implements Animal {...}
 class Poodle extends Dog {...}
 class Labrador extends Dog {...}

Which of the following lines (if any) will not compile?

- Poodle poodle = new Poodle();
- Animal animal = (Animal) poodle;
- Dog dog = new Labrador();
- animal = dog;
- poodle = dog;

Is there an error?

- No compilation error
- Compilation Error
  - Type mismatch: cannot convert
  - Animal to Dog
  - Animal to Poodle
  - Animal to Labrador
  - Dog to Animal
  - Dog to Poodle
  - Dog to Labrador
  - Poodle to Animal
  - Poodle to Dog
  - Poodle to Labrador
  - Labrador to Animal
  - Labrador to Dog
  - Labrador to Poodle
- No Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Exception
- Runtime Error
- No error
Method Overloading & Overriding

```java
public class A {
    public float foo(float a, float b) throws IOException {
        // Method implementation
    }
}

public class B extends A {
    // Method implementation
}
```

Which of the following methods can be defined in B:

1. `float foo(float a, float b){…}`
2. `public int foo(int a, int b) throws Exception{…}`
3. `public float foo(float a, float b) throws Exception{…}`
4. `public float foo(float p, float q) {…}`

Answer: 2 and 4

Inheritance

```java
public class A {
    public void foo() {
        System.out.println("A.foo()".);
    }
    public void bar() {
        foo();
    }
}

public class B extends A {
    public void foo() {
        System.out.println("B.foo()".);
    }
    public static void main(String[] args) {
        A a = new B();
        a.bar();
    }
}
```

The output is:

```
B
B.
```

Method Overriding

```java
public class A {
    public void print() {
        System.out.println("A");
    }
}

class B extends A implements C {
    interface C {
        void print();
    }
}
```

Is there an error?

Yes, the inherited package method `A.print()` cannot hide the public abstract method in C.

Method Overriding

```java
public class A {
    public void print() {
        System.out.println("A");
    }
}

class B extends A {
    public void print() {
        System.out.println("B");
    }
}
```

Does the code compile? If no, why?

No, the code compiles.

Does the code throw a runtime exception?

No, the code does not throw a runtime exception.

What is the output?

```
B
```

Method Overriding & Visibility

```java
public class A {
    protected void print() {
        System.out.println("A");
    }
}

class B extends A {
    public void print() {
        System.out.println("B");
    }
}
```

Does the code compile? If no, why?

Yes, the code compiles.

Does the code throw a runtime exception?

No, the code does not throw a runtime exception.

What is the output?

```
B
```

Method Overriding & Visibility

```java
public class A {
    public static void main(String[] args) {
        B b = new B();
        b.print();
    }
}
```

Does the code compile? If no, why?

No, the code compiles.

Does the code throw a runtime exception?

Yes, the code throws a runtime exception.

What is the output?

```
Compilation error: "Cannot reduce the visibility of the inherited method"
```

Method Overloading & Overriding

```java
public class A {
    public float foo(float a, float b) throws IOException {
        // Method implementation
    }
}

public class B extends A {
    // Method implementation
}
```

Which of the following methods can be defined in B:

1. `float foo(float a, float b){…}`
2. `public int foo(int a, int b) throws Exception{…}`
3. `public float foo(float a, float b) throws Exception{…}`
4. `public float foo(float p, float q) {…}`

Answer: 2 and 4
The output is:

Does the code compile? If no, why? Does the code throw a runtime exception? If yes, why? If no, what is the output?

The output is:

The output is:

The output is:

The output is:

How can you invoke the foo method of A within C? Answer: Not possible (super.super.foo() is illegal)

The output is:

The output is:
Inheritance & Constructors

```java
public class A {  
  public class B {  
    public class C extends A {  
      public C() { System.out.println("in C: no args."); }  
      public C(String s) { System.out.println("in C: s = " + s); }  
    }  
    public class D {  
      public static void main(String[] args) {  
        A a = new C("a");  
        a.foo();  
      }  
    }  
  }  
}
```

What will happen if we remove this line?

```java
public class A {  
  protected B b = new B();  
  public A() { System.out.println("in A: no args."); }  
  public A(String s) { System.out.println("in A: s = " + s); }  
}
```

Inheritance & Constructors

```java
public class B {  
  B() { System.out.println("in B: no args."); }  
}
```

Inheritance & Constructors

```java
public class A {  
  String bar = "A.bar";  
}
```

```java
public class B extends A {  
  String bar = "B.bar";  
  B() { foo(); }  
  public void foo() {  
    System.out.println("B.foo(): bar = " + bar);  
  }  
}
```

Will this compile?

Will there be a RTE?

What is the result?

בחינה באופה!

27.6 הבחרו ב 6 מה她们oinsכיסינו durante הסמסטר (שורשים,  
רטרולוביזיה, Java, DBC, IO, Iterator, stream, Collection Framework, DBC, Java,  
לפתור כמה שיותר בוחנים משנים שעברו,  
לא כל הסמסטרים זהים  
ב้าצלחה!