Given the following definition of class `Thing1`, what is the output of the Java application `Question1`?

```java
public class Thing1 {
    private int count;

    public Thing1( int count )
    {
        this.count = count;
    }

    public int getCount()
    {
        return this.count;
    }

    public void setCount( int count )
    {
        this.count = count;
    }

    public String toString()
    {
        if ( this.count == 1 )
            return "one";
        else if ( this.count == 2 )
            return "two";
        else if ( this.count == 3 )
            return "three";
        else
            return "too many";
    }

    public static void swap1( Thing1 t1, Thing1 t2 )
    {
        Thing1 temp;
        temp = t1;
        t1 = t2;
        t2 = temp;
    }
}

public class Question1 {
    public static void main( String[] args )
    {
        Thing1 first = new Thing1( 4 );
        Thing1 second = new Thing1( 2 );
        System.out.println( first.toString() );
        System.out.println( second.toString() );
        Thing1.swap1( first, second );
        System.out.println( first.toString() );
        System.out.println( second.toString() );
        Thing1 third = new Thing1( 1 );
        Thing1 fourth = new Thing1( 3 );
        second.setCount( third.getCount() );
        first = fourth;
        System.out.println( first.toString() );
        System.out.println( second.toString() );
        System.out.println( third.toString() );
        System.out.println( fourth.toString() );
        System.out.println( first.toString().equals( fourth.toString() ) );
        System.out.println( second.toString().equals( third.toString() ) );
        System.out.println( first == fourth );
        System.out.println( second == third );
    }
}
```

**Output**

```plaintext
one
two
three
four
first.toString().equals( fourth.toString() )
false
second.toString().equals( third.toString() )
false
first == fourth
false
second == third
false
```
Given the following definitions:

```java
public interface Speakable {
    public String speak();
}
```

```java
public class Puppy implements Speakable {
    private static final String PUPPY_SPEAK = "Bark";
    public Puppy() {
        // ctor initialization here
    }
    public String speak() {
        return PUPPY_SPEAK;
    }
    public void sleep( int time ) {
        // puppy sleeps for time seconds
    }
}
```

```java
public class Kitty implements Speakable {
    private static final String KITTY_SPEAK = "Meow";
    public Kitty() {
        // ctor initialization here
    }
    public String speak() {
        return KITTY_SPEAK;
    }
    public void wag() {
        // kitty wags its tail
    }
}
```

And the following variable definitions:

```java
private Puppy puppy;
private Kitty kitty;
private Speakable speakable;
```

Indicate which are valid Java statements. Consider each statement executed sequentially in the order it appears.

A) Valid Java statement – No Compiler Error
B) Invalid Java statement – Compiler Error

```java
puppy = new Puppy();
puppy.speak();
puppy.wag();
puppy.sleep( 1000 );
kitty = new Kitty();
kitty.speak();
kitty.wag();
kitty.sleep( 2000 );
speakable = puppy;
speakable.speak();
speakable.sleep( 3000 );
speakable = new Speakable();
speakable = kitty;
speakable.speak();
speakable.wag();
puppy = kitty;
```

**Hint:** What does the compiler know about any reference variable at compile time (vs. run time)?