What gets printed when the following program is run?

```java
public class While {
    public static void main( String[] args ) {
        final int MAX = 11;
        int i = 8, j = 8;

        while ( i <= MAX ) {
            j = i;
            while ( j < MAX ) {
                --j;
                System.out.println( i + " " + j);
                j += 3;
            }
            i++;
        }

        System.out.println( i + " " + j);
    }
}
```

By default, method headers in a Java interface definition are implicitly

______________________________ and ____________________________

Java interface definitions cannot have (list all that are applicable)

| A) method definitions |
| B) public static final constants |
| C) constructors |
| D) instance variables |

The rules for using ActiveObjects from the objectdraw library are (list all that are applicable)

| A) define a class that implements ActiveObject |
| B) define a class that extends ActiveObject |
| C) define a start() method |
| D) call start() as the first line in the constructor |
| E) call start() as the last line in the constructor |
| F) define a run() method |
| G) call run() from the constructor |
| H) call run() from the begin() method |
| I) pause() occasionally in start() |
| J) pause() occasionally in run() |
Given the following definitions:

```java
public interface Doable
{
    void doit();
}
```

```java
public class Thing1 implements Doable
{
    private static final String SPEAK = "Me";

    public Thing1()
    {
        // ctor initialization here
    }

    public String speak()
    {
        return SPEAK;
    }

    public void doit()
    {
        // Thing1 does its thing
    }
}
```

```java
public class Thing2 implements Doable
{
    public static final String SPEAK = "No, Me";

    public Thing2()
    {
        // ctor initialization here
    }

    public String speak( String s )
    {
        return SPEAK + s;
    }

    public void doit()
    {
        // Thing2 does its thing
    }
}
```

And the following variable definitions:

```
Thing1 thing1;
Thing2 thing2;
Doable doable;
```

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

1) Invalid Java statement – Compiler Error
2) Valid Java statement – No Compiler Error

```java
thing2 = new Thing2(); _______
thing2.speak(); _______
thing2.doit(); _______
thing2.speak( " Mine" ); _______
String s2 = Thing2.SPEAK; _______
thing1 = new Thing1(); _______
thing1.speak(); _______
thing1.doit(); _______
thing1.speak( " Mine" ); _______
```