Dealing with Modifiers

Checking modifiers

```
class Item {
    public static final int maxItems = 100;
    public static int inStock = 19;

    private String name;
    protected int basePrice;

    public Item(String name, int basePrice) {
        this.name = name;
        this.basePrice = basePrice;
    }

    public String getName() {
        return name;
    }

    public int getPrice() {
        return (int) (basePrice * getMarkUp());
    }

    protected double getMarkUp() {
        double markUp = 0.1;
        // ... connecting to the remote server
        return 1 + markUp;
    }
}
```

As you can see, this class contains different types of
fields: private, protected, public, static and final.

**The Modifier class is designed to work with such modifiers.
You can get all modifiers by calling the getModifiers() method on
a Field, Method or Constructor object.**

In fact, this method returns a simple int value that represents all the modifiers: the information on them is contained inside the single number.

Each number can be viewed in binary as 32 different bits, each of them being either 0 or 1.

Each bit's position is responsible for its own modifier, either
being true or false to indicate the presence or absence of the modifier.

Modifiers in source code	Modifiers as an integer
public	00000000 ... 00000001 = 1
private	00000000 ... 00000010 = 2
protected	00000000 ... 00000100 = 4
static	00000000 ... 00001000 = 8
final	00000000 ... 00010000 = 16
other modifiers	other bits

Modifiers in source code	Modifiers as an integer
public static	00000000 ... 00001001 = 9
private final	00000000 ... 00010010 = 18
protected static final	00000000 ... 00011100 = 28

There is also the zero modifier (00000000 ... 00000000 = 0) which is used for default constructors or package-private (or default) access modifier. Please, do not forget about it!

You don't really need to extract these bits to gather information about a particular modifier. The Modifier class has special static methods, such as isPublic or isStatic, which can check whether a field, method or constructor has a specific modifier.

Code example

```
Item item = new Item("apples", 500);

Class itemClass = item.getClass();
Field[] fields = itemClass.getDeclaredFields();

for (Field field : fields) {
   int modifiers = field.getModifiers();
   if (Modifier.isPublic(modifiers)) {
      System.out.print("public ");
   }
   if (Modifier.isProtected(modifiers)) {
      System.out.print("protected ");
   }
   if (Modifier.isPrivate(modifiers)) {
      System.out.print("private ");
   }
   if (Modifier.isStatic(modifiers)) {
      System.out.print("static ");
```

```
    }
    if (Modifier.isFinal(modifiers)) {
        System.out.print("final ");
    }

    System.out.print(field.getType() + " ");
    System.out.println(field.getName());
}
```

This code outputs modifiers of all the fields of the Item class, as well as the types of these fields and field names. The output is here:

```
public static final int maxItems
public static int inStock
private class java.lang.String name
protected int basePrice
```

As you can see, the output is as expected. You can also do the same with methods and constructors.