Analyzing an issue

Let’s analyze an example bug reported by FindBugs. FindBugs always gives us a short description of the bugs it detected, for example:

**Bug: Call to Person.equals(Integer) in Person.addFriend(Person)**

*This method calls equals(Object) on two references of different class types and analysis suggests they will be to objects of different classes at runtime. Further, examination of the equals methods that would be invoked suggest that either this call will always return false, or else the equals method is not be symmetric (which is a property required by the contract for equals in class Object).*

Double-clicking on the bug description takes us to the location the bug was found in the code, in this case:

```java
public void addFriend(Person person){
    if (person.equals(id)){
        System.out.println("One cannot be in his/her own friends list!");
    }
    else {
        friends.add(person);
    }
}
```

When analyzing an issue reported by FindBugs, the most important thing is to figure out whether an actual fault or a false positive was found. In our case FindBugs seems to be correct, since the method is trying to compare an integer value to an object from the class Person. Nevertheless, there is a chance the `equals()` method has been overridden in the class Person and the comparison might therefore be the intended implementation.

Let’s examine two possible implementations of the Person class:
Case 1

Code:

```java
import java.util.ArrayList;

public class Person {
    private String name;
    private int id;
    private ArrayList<Person> friends;

    public Person(String name, int id, ArrayList<Person> friends) {
        super();
        this.name = name;
        this.id = id;
        this.friends = friends;
    }

    public void addFriend(Person person){
        if (person.equals(id)){
            System.out.println("One cannot be in his/her own friends list!");
        }
        else {
            friends.add(person);
        }
    }
}

Conclusion:

Since the equals() method is not overridden in this class, it is used incorrectly in the addFriend() method and therefore FindBugs detected an actual fault.
Case 2

Code:

```java
import java.util.ArrayList;

public class Person {
    private String name;
    private int id;
    private ArrayList<Person> friends;

    public Person(String name, int id, ArrayList<Person> friends) {
        super();
        this.name = name;
        this.id = id;
        this.friends = friends;
    }

    @Override
    public boolean equals(Object object) {
        try {
            return (int)object == id;
        }
        catch (Exception e) {
            return false;
        }
    }

    public void addFriend(Person person) {
        if (person.equals(id)) {
            System.out.println("One cannot be in his/her own friends list!");
        } else {
            friends.add(person);
        }
    }
}
```

Conclusion:

Even though this code is breaking some Java coding conventions, the `equals()` method is overridden in a way that it accepts integer values as arguments and compares them to the person’s id. This suggests that the line reported by FindBugs was an intended implementation and the bug therefore a false positive. Nevertheless, for understandability and scalability reasons the `equals()` method should be fixed to match the default coding conventions.