CurriculaCourses Bugs

Path to find it:

\JavaSource\org\unitime\timetable\gwt\client\curricula.

Java File Name : CurriculaCourses.java

Bugzilla Link for all Bugs reports :

https://bugzilla.mozilla.org/buglist.cgi?bug_status=__open__&email1=hanaaymanyehia%40gmail.com&emailassigned_to1=1&emailreporter1=1&list_id=16529686

After running the sonar scanner analysis on that java file

On line 1189 :

Bug ID on Bugzilla -> https://bugzilla.mozilla.org/show_bug.cgi?id=1831569

```
return - (s0 == null ? Float.valueOf(0) : s0).compareTo(s1 == null ?
Float.valueOf(0) : s1);
```

The bug -> **Use the original value instead.**

More explanation :

It is possible for a call to hashCode to return Integer.MIN_VALUE. Take the absolute value of such a hashcode and you'll still have a negative number. Since your code is likely to assume that it's a positive value instead, your results will be unreliable.

Similarly, Integer.MIN_VALUE could be returned from Random.nextInt() or any object's compareTo method, and Long.MIN_VALUE could be returned from Random.nextLong(). Calling Math.abs on values returned from these methods is similarly ill-advised.

On line 1196:

Bug ID on Bugzilla -> https://bugzilla.mozilla.org/show_bug.cgi?id=1831571

```
return - (i0 == null ? Integer.valueOf(0) : i0).compareTo(i1 == null ?
Integer.valueOf(0) : i1);
```

The bug -> **Use the original value instead.**

More explanation :

It is possible for a call to hashCode to return Integer.MIN_VALUE. Take the absolute value of such a hashcode and you'll still have a negative number. Since your code is likely to assume that it's a positive value instead, your results will be unreliable.

Similarly, Integer.MIN_VALUE could be returned from Random.nextInt() or any object's compareTo method, and Long.MIN_VALUE could be returned

from Random.nextLong(). Calling Math.abs on values returned from these methods is similarly ill-advised.

On line 1706:

Bug ID on Bugzilla -> https://bugzilla.mozilla.org/show_bug.cgi?id=1831573

```
public boolean equals(Object o)
```

The bug -> **This class overrides "equals()" and should therefore also override "hashCode()".**

More explanation :

According to the Java Language Specification, there is a contract between equals(Object) and hashCode():
If two objects are equal according to the equals(Object) method, then calling the hashCode method on each of the two objects must produce the same integer result.
It is not required that if two objects are unequal according to the equals(java.lang.Object) method, then calling the hashCode method on each of the two objects must produce distinct integer results.

However, the programmer should be aware that producing distinct integer results for unequal objects may improve the performance of hashtables.

In order to comply with this contract, those methods should be either both inherited, or both overridden.