Assignment 4 Part 1: SpotBugs

Charles Ancheta

October 21 2020

1 Bugs found by SpotBugs

1. Possible null pointer dereference of input in ece325.CodingHorror.main(String[])
   on exception path Dereferenced at CodingHorror.java:[line 24]
2. Dereference of the result of readLine() without nullcheck in ece325.CodingHorror.main(String[])
   At CodingHorror.java:[line 24]
3. Return value of String.replace(char, char) ignored in ece325.CodingHorror.main(String[])
   At CodingHorror.java:[line 24]
4. Found reliance on default encoding in ece325.CodingHorror.main(String[]):
   new java.io.InputStreamReader(InputStream) At CodingHorror.java:[line
   16]
5. Comparison of String objects using == or != in method ece325.CodingHorror.main(String[])
   At CodingHorror.java:[line 25]

2 Reasons for the bugs

1. "input" was used without null checking the result of "br.readLine()" first. "input" may not have the intended value.
2. A method of "input" was called without null checking "input" first. Calling ".replace(char, char)" of the input might throw a NullPointerException.
3. The result of "input.replace(char, char)" was not used. Strings are immutable objects so its methods do not mutate the object themselves.
4. The InputStreamReader is relying on the default encoding format used by the machine that runs the program. The characters may have different byte values when used.
5. == and != compares object references. This kind of String comparison only works on literally defined Strings and will not work on Strings received from the InputStream.

3 Solution for the bugs

1. Check if "input" is null first before using it.
2. Same as 1.
3. Assign the result of "input.replace(char, char)" to a variable (e.g. input).
4. Specify a character set to pass on "new InputStreamReader(InputStream, CharSet)" (e.g. CharSet.forName("UTF-8")).
5. Use String.equals(String) method instead of == and != operators for a character-by-character comparison.