1. Equivalence relations are important because they are part of the contract that you have if you want to be stored in a collection like ArrayList or Map. If you do not correctly implement an equals method, you will experience subtle bugs.
2. The first layer checks for low-level errors like NullPointerException and ClassCastException by data flow analysis. The second layer actually recognizes abstractions for equality conditions from java code and turns it into an Alloy model.
3. This class breaks the promise of reflexivity. If the iort field in ORTI is null, the .equals will always return false. This means that there is a case where an ORTI is not equal to itself
4. Java types are modeled using Alloy signatures and inheritance is modeled using signature extension.
5. Fields are modeled as a field in the corresponding signature. Primitives are modeled as Ints in Alloy. The value 0 represents null.
6. Unknown methods are modeled as a nondeterministic function.
7. They are converted to Alloy predicates and setting the right side to 0 or 1 for true and false respectively based on which comparison operator.
8. This is an optimization. The loop unrolling prunes cases that makes that same logic run. If there are two different ways to make the same logic run, EQ will only look at once case in which that happens. This ensures a minimal set of paths through the control flow but also testing everything.
9.

Tomcat 6	Lucene 3.0	JFreeChart	JDK 1.5
13	11	19	128

10. They return false when (this == object). This is a reflexivity violation
11. EQ just modeled the java program. Alloy was what actually analyzed the system and produced contradictions.
12. The existing checkers do not check for violations of the equivalence relation.