Below are the rules about what makes a constant expression. They are taken directly from Java Language Specification and here is the direct link: http://docs.oracle.com/javase/specs/jls/se7/html/jls-15.html#jls-15.28. We already discussed the most common rules. So, don't worry if you are not understanding all the rules. Also, you can check out http://www.coderanch.com/t/454384/java/java/compile-time-constant. It is also about these rules and you can find it helpful and it is a great pointer on constant variables.

From Java Language Specification:

A compile-time *constant expression* is an expression denoting a value of primitive type or a String that does not complete abruptly and is composed using only the following:

- Literals of primitive type and literals of type String (§3.10.1, §3.10.2, §3.10.3, §3.10.4, §3.10.5)
- Casts to primitive types and casts to type String (§15.16)
- The unary operators +, -, ~, and ! (but not ++ or --) (§15.15.3, §15.15.4, §15.15.5, §15.15.6)
- The multiplicative operators *, /, and % (§15.17)
- The additive operators + and - (§15.18)
- The shift operators <<, >>, and >>> (§15.19)
- The relational operators <, <=, >, and >= (but not instanceof) (§15.20)
- The equality operators == and != (§15.21)
- The bitwise and logical operators &, ^, and | (§15.22)
- The conditional-and operator && and the conditional-or operator || (§15.23, §15.24)
- The ternary conditional operator ? : (§15.25)
- Parenthesized expressions (§15.8.5) whose contained expression is a constant expression.
- Simple names (§6.5.6.1) that refer to constant variables (§4.12.4).
- Qualified names (§6.5.6.2) of the form *TypeName* . *Identifier* that refer to constant variables (§4.12.4).

Other good pointers on final variables:

http://www.javapractices.com/topic/TopicAction.do?Id=23

http://stackoverflow.com/questions/154314/when-should-one-use-final-for-method-parameters-and-local-variables