Basics			
Working and compiling code		1.0	
Surface quality			
Language	Proper, idiomatic use of language. No while loop if foreach will do, etc. Generics. Proper use of standard library and frameworks. No reimplementation of stdlib features.	0.8	
Tools	Integrated use of Eclipse, Subversion, automation using build scripts (incl. parser generator). Proper use of APIs provided by tools. Unit test frameworks.	0.3	
Style	Consistency in indentation, coding convention and comments (JavaDoc). Code over comment. Comments for rationale.	0.3	
Names	Consistent naming convention. Intention revealing names. Small scopes, no name space pollution. Declarations close to use. No mutable globals. Proper packaging.	0.3	
Cruft	No debug print statements, commented out code. No dead code. Short methods/functions. Small classes. No God class.	0.3	
Design Quality			
Simplicity	Low cyclomatic complexity. No convoluted designs. Absence of boilerplate code. No work arounds to hide design flaws. No unneeded indirections.	2.0	
Encapsulation	Programming against interfaces. `List` vs. `ArrayList`. Proper use of inheritance (is_a). No fragile base-classes. Weak coupling, strong cohesion. Open for extension, closed for modification.	2.0	
Duplication	"Once and only once", DRY, single point of change. No parallel inheritance hiearchies. Factoring of methods.	1.5	
Separation of concerns	Single responsibility. Proper packaging of classes. No cyclic dependencies. Sub systems/components.	1.5	
Total:		10.0	