Appendix 2 - University comparison

Lecture	*AUT University*	*Stanford*	*Comment*
1	Procedural_Programming_Models Computer_Models Hardware_Models Stored_Program_Models Programming_as_a_Process_Models	Karel_the_Robot_Models	AUT introduces programming as being like the steps in making a cup of tea. Stanford is an introductory lecture with mostly administration and protocols.
2	Procedural_Programming_Models Karel_the_Robot_Models Immediate_Command_Models BlueJ_IDE_Models Call_Statements Language_Syntax_Models Conditional_Expressions Logical_Expressions. If_Structure Loop_Structure	Algorithm_Models Stored_Program_Models Karel_the_Robot_Models Conditional_Expressions	AUT: introduce program commands without mentioning variables. Illustrates the effect of immediate commands and conditional tests.
3	Hardware_Models Primitive_Types Java_Language Assembly_Language	Karel_the_Robot_Models Java_Language In-Line_Comments Conditional_Expressions Software_Engineering_Models Quality_Models	AUT: Comparison of languages and language attributes. Stanford: Introduction to Java and quality concepts.
4	Text_Identifiers Semi_Interpreted_Language_Models Debugging_Models	Historical_Background_Models Compiler_Models Object_Orientated_Models Inheritance_Models Java_Language Decomposition_Models	Stanford: Introduces object models and Java as an object-orientated language, including inheritance. They also examine their first complete Java program.
5	Language_Syntax_Models	Variable_Identifiers Data_Types Classes_as_Types_Models Objects_as_Variables_Models Method_Invocation_Models Graphic_Models Coordinate_Models Expressions	AUT: Why syntax and formatting is important. How it relates to control structures Standford: Objects and complex variables. Introduce methods that act on classes that manage graphics.
6	Control_Structure_Models Indentation Nesting	Precedence_Rules_Expressions Floating_Point Boolean Type_Conversion Variable_Scope_Models Conditional_Expressions If_Structure Nesting	AUT: Why indentation and formatting is important. How it relates to control structures. Nesting control structures and flow Stanford: variable types and operator precedence.
7	Program_State_Models Software_Requirements_Models	For_Loop_Structure While_Loop_Structure Method_Invocation_Models Return_Statements	AUT: How program commands affect the state of the program. Requirements Stanford: Method invocation and loop structures Returning data from method calls

Lecture	*AUT University*	*Stanford*	*Comment*
8	Program_State_Models Debugging_Models	Encapsulation_Models Data_Hiding_Models Local_Variable_Types	AUT: Tracing code while debugging Stanford: Data hiding and encapsulation.
9	Data_Types Variable_Identifiers Constant_Identifiers Logical_Expressions Variable_Initialisation_Models	Strings Data_Hiding_Models Call_By_Reference_Models Call_By_Value_Models	AUT: Detailed examination of variables and data types, constants. Comparing values. Initialising variables. Stanford: First introduction to call by value and call by reference.
10	Precedence_Rule_Expressions PostFix_PreFix_Expressions	Data_Hiding_Models Graphics_Models	Stanford: Lots of graphics examples
11	Debugging_Models Printing_Command_Models	Graphics_Models	AUT: Debugging techniques. Examining values by printing to the console. Stanford: Lots of graphics examples.
12	Software_Requirements_Models Junit_Testing_Model	Strings UniCode_Types	Stanford: Data representation including UniCode
13	Boolean Return_Statements	String_Expressions Encryption_Models	Stanford: Data encryption and string tokenising
14	Method_Invocation_Models	Data_Types Data_Type_Memory_Allocation_Models	Stanford: Different memory allocation models
15	Reference_Models Object_Orientated_Models String_Expressions	Pointer_Types Call_By_Reference_Models Call_By_Value_Models File_IO_Models	AUT: First mention of Karel as being a “complex variable” First mention of “Object-Orientation” ? Stanford: Discussion of C (?) pointers and parameter passing File I/O introduced
16	Software_Requirements Algorithm_Models	Arrays Data_Type_Memory_Allocation_Models Iteration_Models	AUT: Planning and design of an algorithm from requirements. Stanford: Arrays and iteration
17	Algorithm_Models	Multi_Dimensional_Arrays Array_List_Model Graphics_Models	AUT: Writing algorithms, best practice.
18	Variable_Scope_Models	Multi_Dimensional_Arrays Debugging_Models Eclipse_IDE_Models	AUT: Scope of variables.
19	Return_Values Variable_Scope_Models	HashMap_Access_Models	AUT: Extend variable scope concepts
20	Function_Identifiers	Window_GUI_Models Graphics_Models	Stanford: Introduction to Swing GUI’s

Lecture	*AUT University*	*Stanford*	*Comment*
21	Quality_Models	Event_Driven_Models	AUT: What is a “good” program Stanford: Interactors and listeners in event-driven applications.
22	Void_Return_Types Instance_Models Objects_as_Variables_Models	Event_Driven_Models	Stanford: More complex listener programs
23	Arrays	Searching_Models Sorting_Models	
24	For_Loop_Structures	Software_Engineering_Models Quality_Models	Stanford: Introduction to some complex program examples.
25	Program_Exceptions Throwing_Program_Exceptions Catching_Program_Exceptions	Software_Requirements_Models	AUT: Throwing and handling exceptions Stanford: Discussion of social networking
26	Object_Method_Contracts	Java_Libraries JAR_Program_Packaging	
27	Type_Conversion Constant_Identifiers		AUT: Type conversion methods and final (constant) data types Stanford: Review of subsequent courses
28	Object_Orientated_Methods		AUT: Everything that they had not been told before about methods. This is preparation for the next course and addresses some simplifications made in the course. Stanford: Review of course and wrapup