Preserving Behavior During Refactoring - Chapter 4

After a refactoring, a program must be syntactically correct. There are ways to preserve a program's behavior which rely on the compiler catching your mistakes but there are some errors that could change the behavior of the program but would not be picked up by the compiler. A particular set of syntactic and semantic properties of programs has been found to be easily violated if explicit checks are not made before a program is refactored. Those properties are:

- Unique superclass
- Distinct class names
- Distinct member names
- Inherited member variables not redefined
- Compatible signatures in member function redefinition
- Type-safe assignments
- Semantically equivalent references and operations

Semantic equivalence is defined as follows: taking any refactored section of a program, the inputs to that section should produce the same output from that section as the original version. The paper then summarizes the kinds of refactorings that can be made while preserving semantic equivalence. Then it defines the domains of the arguments to refactorings and functions for describing preconditions of refactorings.