The Liskov substitution principle states that if a parent class has subtype child class, then any objects of the parent class can be replaced by objects of the child subclass without modifying the behavior or correctness of the program. For this principle to be valid, the invariants must be inherited and preserved from the parent class, the preconditions must not be strengthened but can be weakened, and the postconditions can be strengthened but not weakened.

The principle relates to our contractual specifications by behaving as expected in extended classes. For example, if we take the BinaryTree class and any of its subclasses, we see that the invariant that disallows the height to be 0 of any tree is inherited throughout all its subclasses. Hence, if we try to set the height to 0 for any subclass of BinaryTree, it will be disallowed by this invariant within the contractual specifications. Therefore the invariants of the BinaryTree are preserved throughout all its subtypes. Hence the Liskov Substitution Principle relates to the contractual specifications.

It does not relate to our contractual specifications when an object is replaced with the instances of its subtype because this alters the correctness of the program and changes the behavior of the program. If we take for example the class BinaryTree and extended it to the class FullBinaryTree, we see that the behavior is changed. The contractual specification in Binary allow a node to have 0,1, or 2 nodes as children. However the FullBinaryTree does not allow a node to have only one child in its contractual specifications. We observe that the preconditions have been strengthened and that the methods for setting a single child node that define the BinaryTree are no longer valid within the FullBinaryTree subclass. Hence the contractual specifications change the behavior of the subclass, ultimately violating the Liskov Substitution Principle.