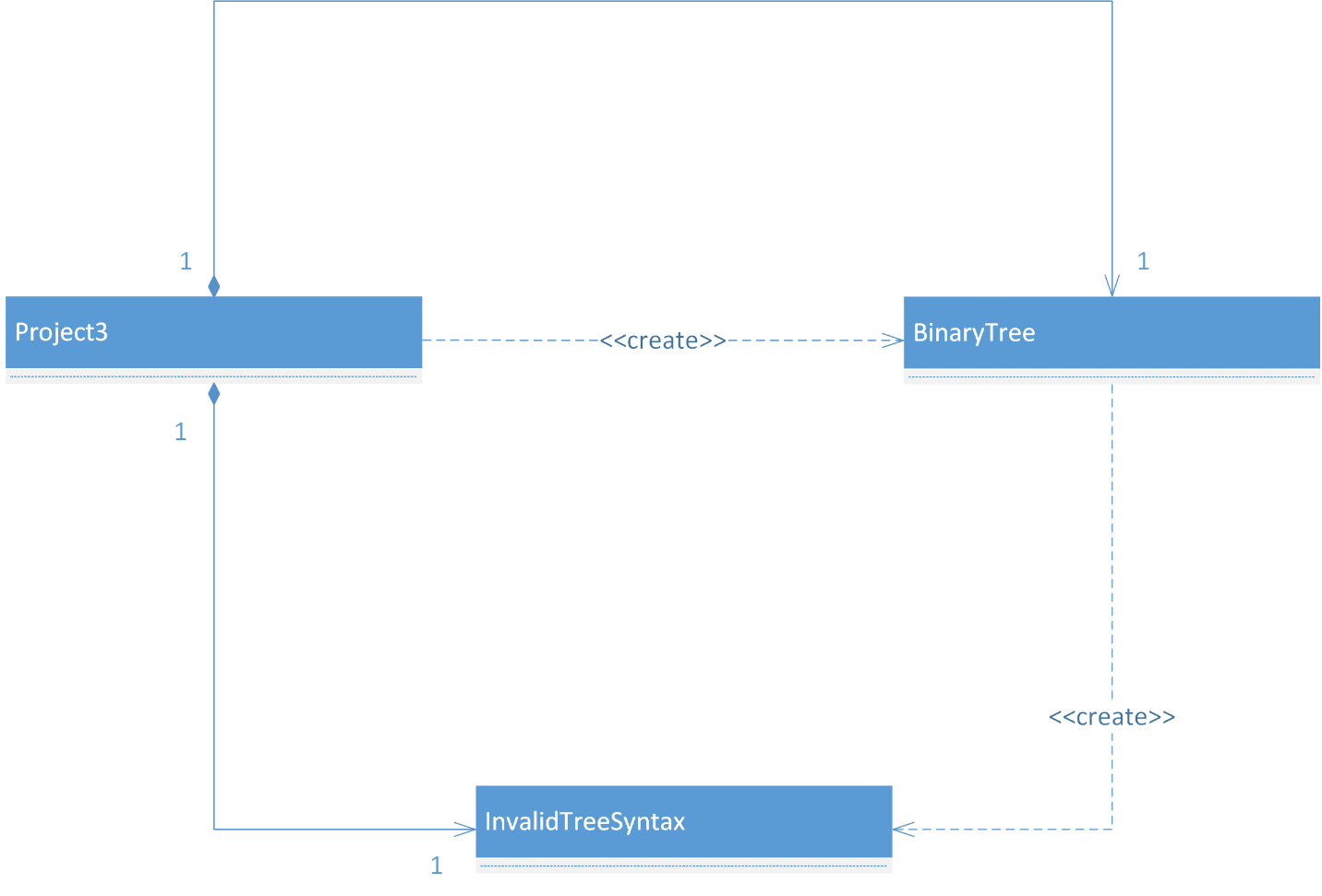
**UML Diagram**



**Test cases**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ser** | **Input** | **Function** | **Output** | **Correct** |
|  | (A(B(C)(D))(E(F)(G))) | Make tree | (A(B(C)(D))(E(F)(G)) | Yes |
|  | “ | Is Balanced? | True | Yes |
|  | “ | Is Full? | True | Yes |
|  | “ | Is Proper? | True | Yes |
|  | “ | Height | 2 | Yes |
|  | “ | Nodes | 7 | Yes |
|  | “ | In Order | (((C)B(D))A((F)E(G))) | Yes |
|  | (X(Y(Z)(I))) | Is Balanced? | False | Yes |
|  | (J(K(L)(M))(N(O))) | Is Full? | False | Yes |
|  | “ | Is Proper? | False | Yes |
|  | (X) | Height | 0 | Yes |
|  | “ | Nodes | 1 | Yes |
|  | Blank | Make tree | Error | Yes |
|  | (a(b)(c)(d)) | Make tree | Error | Yes |
|  | (1 | Make tree | Error | Yes |
|  | (9)( | Make tree | Error | Yes |

**Lessons learnt**

By using recursions in this exercise, since there were lot of recursions involved it made me get used to it and more comfortable with it. By implementing binary trees, I got more confident on how various operations on trees work. However, there was trouble generating nodes recursively as there were lot of parenthesis, in future more complex one would be implemented. Tried to keep the code as short as possible.