Experiment – 1 b: TypeScript

Name of Student	Spandan Deb
Class Roll No	13
D.O.P.	
D.O.S.	
Sign and Grade	

Aim: To study Basic constructs in TypeScript.

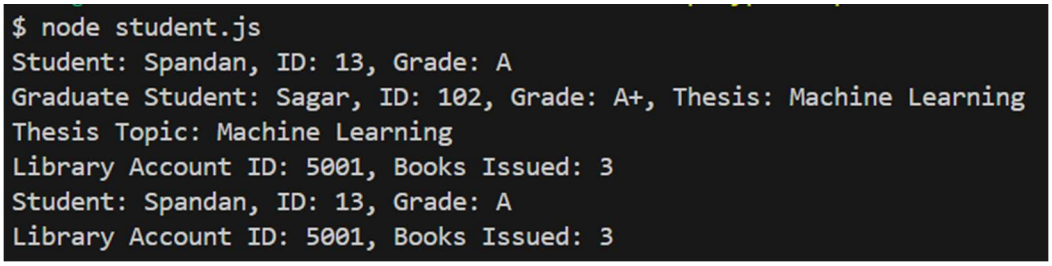
Overview of Tasks Performed:

This experiment demonstrated inheritance and method overriding in TypeScript by creating a `Student` base class and a `Graduate Student` subclass. Composition was showcased using a `Library Account` class associated with `Student`. Additionally, an employee management system was developed with an `Employee` interface, implemented by `Manager` and `Developer` classes, highlighting structured code organization and method customization.

GitHub Link- https://github.com/spandandeb/WebXEx1b

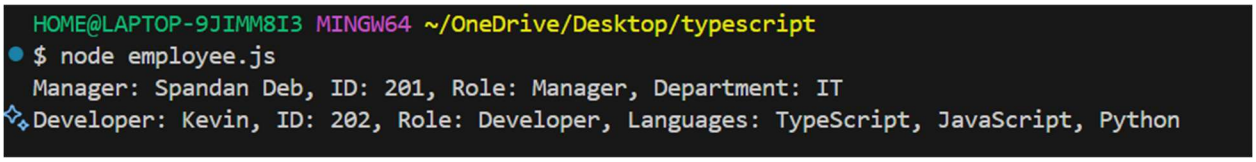
Output:

a) Student and Graduate Student with composition



This screenshot displays the output of the TypeScript program implementing inheritance. The program first prints details of a Student and a GraduateStudent, demonstrating method overriding and inheritance. Then, it prints the Thesis Topic of the GraduateStudent separately. The next lines show details of a LibraryAccount associated with a student, demonstrating composition. Finally, it displays a combined output of both Student and LibraryAccount, showcasing how composition works.

b) Employee Management System



This screenshot displays the output of the Employee Management System program. It shows details of an Employee interface, with two classes: Manager and Developer, implementing it. The output displays the details of a Manager named Spandan, including their ID, role, and department. It also shows the details of a Developer named Kevin, including their programming languages.

Conclusion:

This experiment demonstrated the fundamental concepts of TypeScript, such as inheritance, method overriding, and composition through the implementation of Student and GraduateStudent classes. Instead of using inheritance, composition was demonstrated by linking LibraryAccount with Student, emphasizing flexibility in design.

Furthermore, the Employee Management System utilized interfaces to enforce structure and type safety, highlighting the advantages of TypeScript in maintaining scalable and well-organized code. Overall, this experiment reinforced the benefits of TypeScript's object-oriented capabilities, improving code readability, reusability, and reliability.