

AIM: Perform the following:-

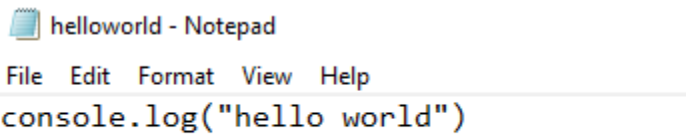
1. Small code snippets for programs like Hello World, Calculator using TypeScript.
2. Inheritance example using TypeScript
3. Access Modifiers example using TypeScript

IMPLEMENTATION:

Part 1:-Small code snippets for programs like Hello World, Calculator using TypeScript.

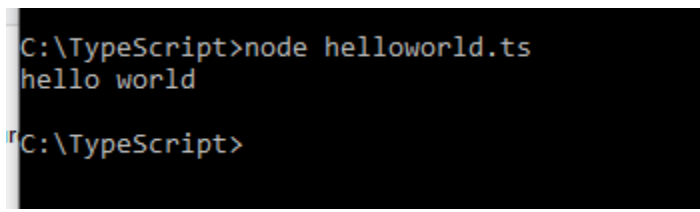
- **Hello world Example**

Code:-



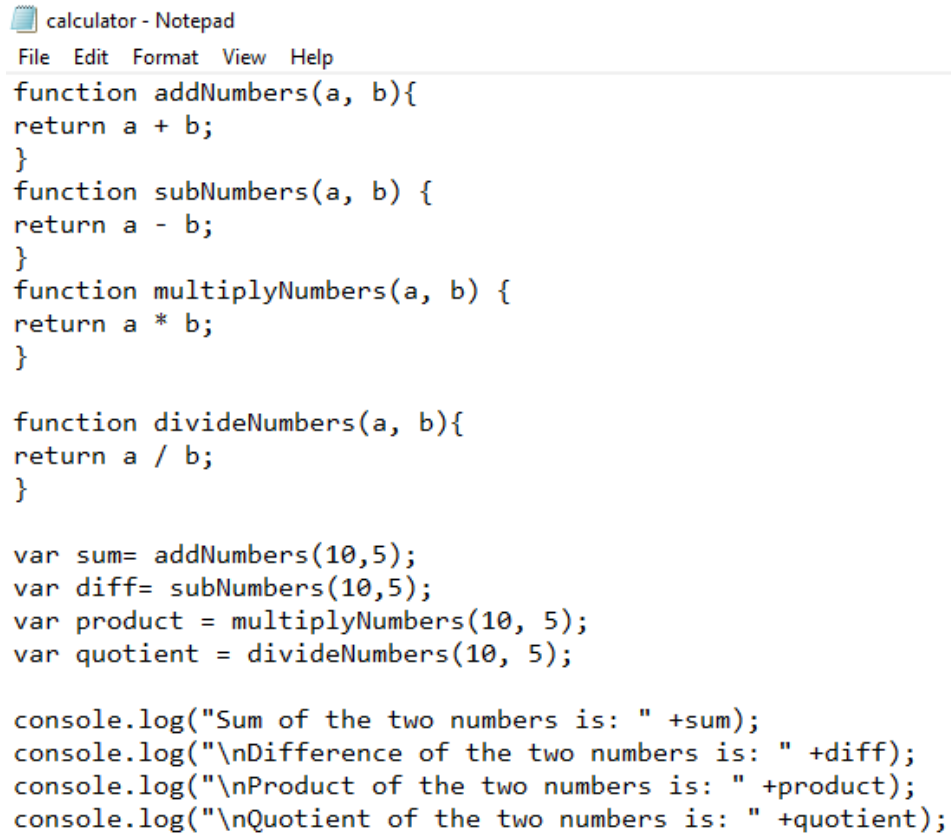
```
helloworld - Notepad
File Edit Format View Help
console.log("hello world")
```

Output:-



```
C:\TypeScript>node helloworld.ts
hello world
C:\TypeScript>
```

- **Calculator Example**

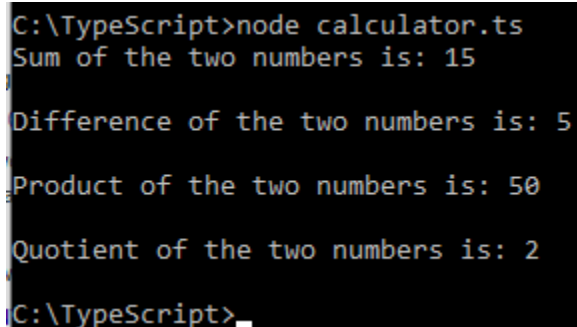
Code:-

```
calculator - Notepad
File Edit Format View Help
function addNumbers(a, b){
return a + b;
}
function subNumbers(a, b) {
return a - b;
}
function multiplyNumbers(a, b) {
return a * b;
}

function divideNumbers(a, b){
return a / b;
}

var sum= addNumbers(10,5);
var diff= subNumbers(10,5);
var product = multiplyNumbers(10, 5);
var quotient = divideNumbers(10, 5);

console.log("Sum of the two numbers is: " +sum);
console.log("\nDifference of the two numbers is: " +diff);
console.log("\nProduct of the two numbers is: " +product);
console.log("\nQuotient of the two numbers is: " +quotient);
```

Output:-

```
C:\TypeScript>node calculator.ts
Sum of the two numbers is: 15

Difference of the two numbers is: 5

Product of the two numbers is: 50

Quotient of the two numbers is: 2
C:\TypeScript>
```

Part 2:- Inheritance example using TypeScript

- **Single Inheritance**

Code:-

Single - Notepad

File Edit Format View Help

Name

Age

Gender

```
constructor(name, age, gender) {
```

```
  this.Name = name;
```

```
  this.Age = age;
```

```
  this.Gender = gender;
```

```
}
```

```
}
```

```
class Employee extends Person {
```

```
  EmployeeNumber
```

```
  Salary
```

```
  constructor(name, age, gender, empno, salary) {
```

```
    super(name, age, gender);
```

```
    this.EmployeeNumber = empno;
```

```
    this.Salary = salary;
```

```
  }
```

```
  display() {
```

```
    console.log("Employee Name: " + this.Name);
```

```
    console.log("Employee Age: " + this.Age);
```

```
    console.log("Employee Gender: " + this.Gender);
```

```
    console.log("Employee Number: " + this.EmployeeNumber);
```

```
    console.log("Employee Salary: " + this.Salary);
```

```
  }
```

```
}
```

```
let emp = new Employee("Mayuri Yerande", 20, "Female", 22512, 85000 );
```

```
emp.display();
```

Output:-

```
C:\TypeScript>node single.ts
Employee Name: Mayuri Yerande
Employee Age: 20
Employee Gender: Female
Employee Number: 22512
Employee Salary: 85000
```

- **Multiple Inheritance**

Code:-

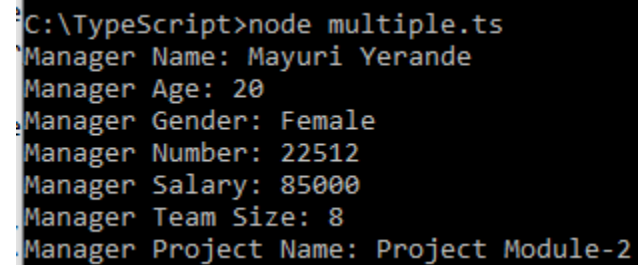
```
class Person {
  Name
  Age
  Gender
  constructor(name, age, gender) {
    this.Name = name;
    this.Age = age;
    this.Gender = gender;
  }
}
```

```
class Employee extends Person {
  EmployeeNumber
  Salary
  constructor(name, age, gender, empno, salary) {
    super(name, age, gender);
    this.EmployeeNumber = empno;
    this.Salary = salary;
  }
}
```

```
class Manager extends Employee {
  Teamsize
  Projectname
  constructor(name, age, gender, empno, salary,
    teamsize, projectname) {
    super(name, age, gender, empno, salary);
    this.Teamsize = teamsize;
  }
}
```

```
this.Projectname = projectname;
}

display(){
console.log("Manager Name: " + this.Name);
console.log("Manager Age: " + this.Age);
console.log("Manager Gender: " + this.Gender);
console.log("Manager Number: " + this.EmployeeNumber);
console.log("Manager Salary: " + this.Salary);
console.log("Manager Team Size: " + this.Teamsize);
console.log("Manager Project Name: " + this.Projectname);
}
}
let manager = new Manager("Mayuri Yerande", 20, "Female", 22512,
85000, 8, "Project Module-2"
);
manager.display();
```

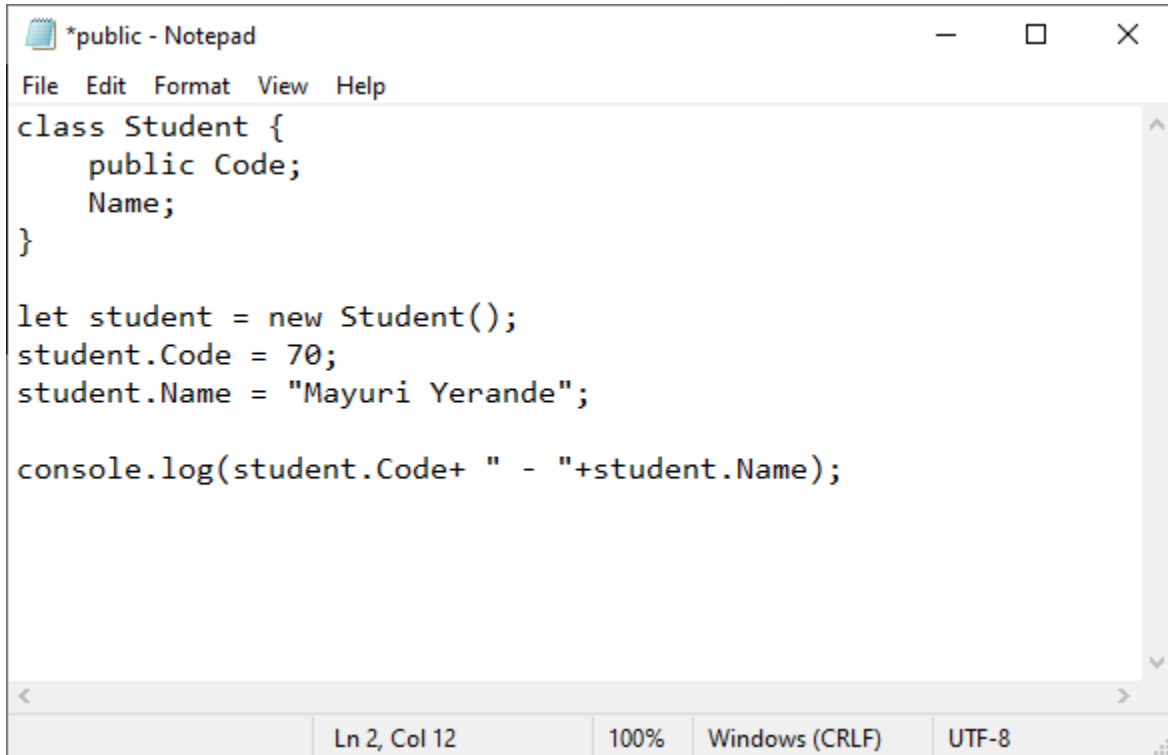
Output:-

```
C:\TypeScript>node multiple.ts
Manager Name: Mayuri Yerande
Manager Age: 20
Manager Gender: Female
Manager Number: 22512
Manager Salary: 85000
Manager Team Size: 8
Manager Project Name: Project Module-2
```

Part 3. Access Modifiers example using TypeScript

- Public example (by default)

Code:-



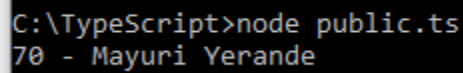
```
*public - Notepad
File Edit Format View Help
class Student {
    public Code;
    Name;
}

let student = new Student();
student.Code = 70;
student.Name = "Mayuri Yerande";

console.log(student.Code+ " - "+student.Name);

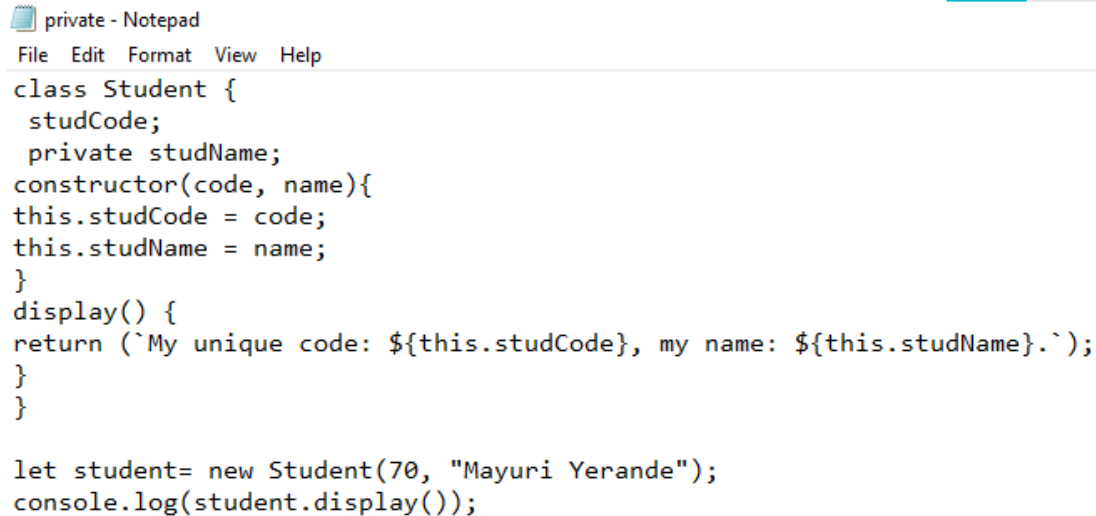
Ln 2, Col 12    100%    Windows (CRLF)    UTF-8
```

Output:-



```
C:\TypeScript>node public.ts
70 - Mayuri Yerande
```

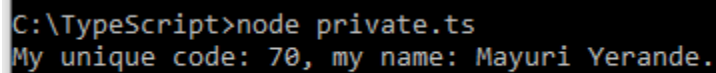
- **Private example**

Code:-

```
private - Notepad
File Edit Format View Help
class Student {
  studCode;
  private studName;
  constructor(code, name){
    this.studCode = code;
    this.studName = name;
  }
  display() {
    return `My unique code: ${this.studCode}, my name: ${this.studName}.`;
  }
}

let student= new Student(70, "Mayuri Yerande");
console.log(student.display());
```

|

Output:-

```
C:\TypeScript>node private.ts
My unique code: 70, my name: Mayuri Yerande.
```

- **Protected example**

Code:-

```
protected - Notepad
File Edit Format View Help
class Student {
  studCode;
  protected studName;
  constructor(code, name){
    this.studCode = code;
    this.studName = name;
  }
}
class Person extends Student {
  private department;

  constructor(code, name, department) {
    super(code, name);
    this.department = department;
  }

  getElevatorPitch() {
    return `My unique code: ${this.studCode}, my name: ${this.studName} and I am in ${this.department} Branch.`;
  }
}
let person = new Person(70, "Mayuri", "INFT");
console.log(person.getElevatorPitch());
|
```

Output:-

```
C:\TypeScript>node protected.ts
My unique code: 70, my name: Mayuri and I am in INFT Branch.
```

CONCLUSION: We implemented basic programs on typescript like hello world and calculator. We learnt about inheritance and access modifiers in typescript and implemented its programs.