Name: **Preeti Khamkar**

Class: **D15A**

Roll no.: **28**

Experiment – 2: TypeScript

1. **Aim:** To study Basic constructs in TypeScript.
2. **Problem Statement:**
    1. Implement a simple Calculator to demonstrate the usage of different data types (include any, never)
    2. Develop a TypeScript application for inheritance hierarchy. Create classes User, Admin (inherits from User), and DataManager. Implement access specifiers (public, private, protected) to regulate data visibility.

       User Class:
       username (private).
       email (protected).
       getUserInfo() (public).

       Admin Class (Inherits from User):
       adminName (private).
       adminEmail (protected).
       grantAccess(user: User) (public).

       DataManager Class:
       data (private).
       getData(admin: Admin) (protected).

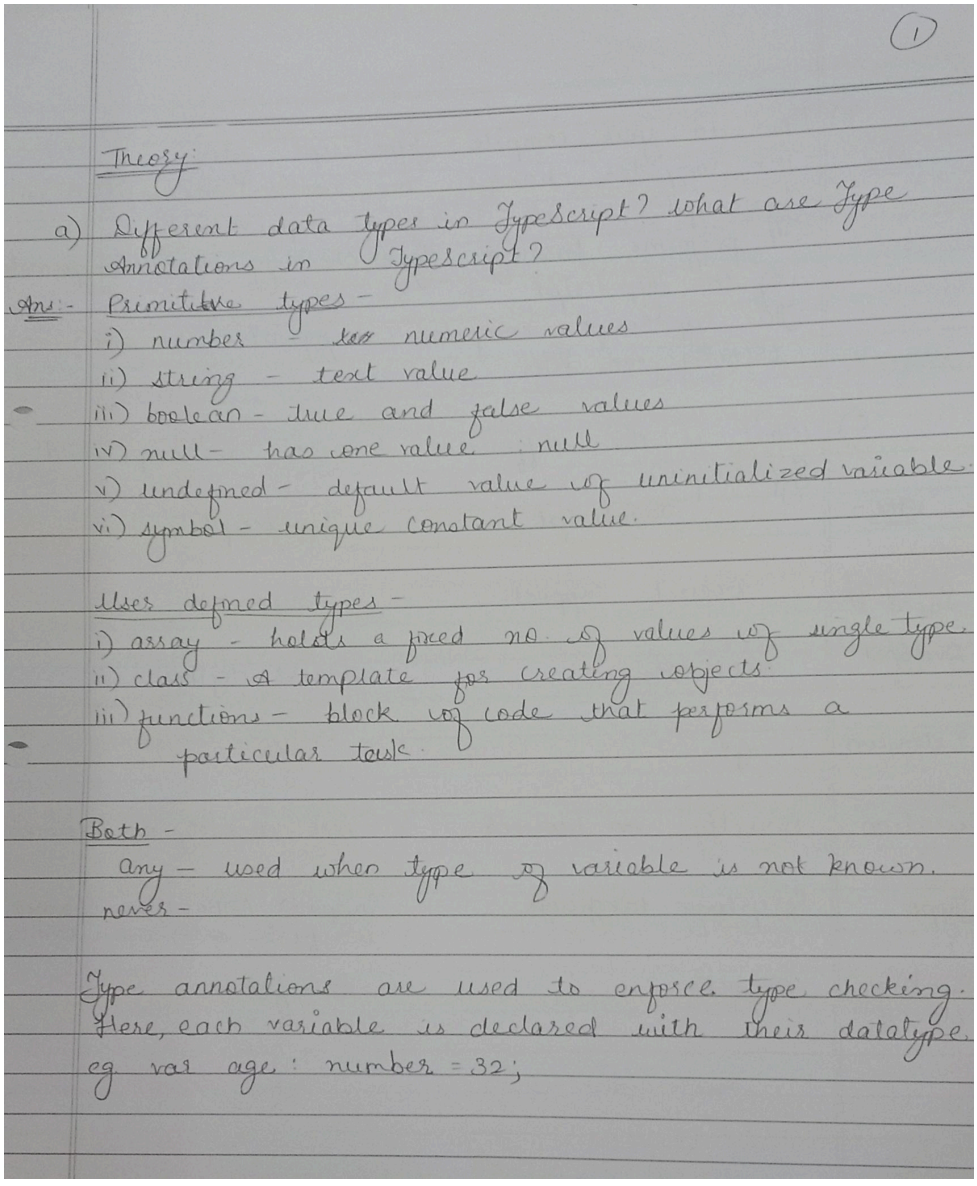
       Develop a system to demonstrate proper access control through inheritance.
    3. Create a TypeScript program for working with geometric shapes.

       Shape Interface:
       Define an interface Shape<T> with methods:

calculateArea(): T - Calculates and returns the area.
calculatePerimeter(): T - Calculates and returns the perimeter.

Shape Classes:
Implement classes for specific shapes (circle, rectangle, triangle) following the Shape<T> interface.
Use generics (T) to specify the numeric type for area and perimeter calculations.

3. Theory:



(1)

Theory:

a) Different data types in Typescript? what are Type Annotations in Typescript?

Ans:- Primitive types -
i) number - for numeric values
ii) string - text value
iii) boolean - true and false values
iv) null - has one value : null
v) undefined - default value of uninitialized variable.
vi) symbol - unique constant value.

User defined types -
i) array - holds a fixed no. of values of single type.
ii) class - a template for creating objects.
iii) functions - block of code that performs a particular task.

Both -
any - used when type of variable is not known.
never -

Type annotations are used to enforce type checking. Here, each variable is declared with their datatype.
eg var age : number = 32;

b) How do you compile TypeScript files?

Ans:- The compiler changes the instructions written in TypeScript to its Javascript equivalent.
- It performs passing, type-checking and transformation of the TypeScript to JS code.
- The typeScript compiler configuration is given in tsconfig.json file.

c) What is the difference between Javascript and TypeScript?

Ans:-

	Javascript	TypeScript
Strongly/ typed	Doesn't support	Supported
Developed by	Netscape	Anders Hejlberg
Extension	.js	.ts
Execution	Directly on browsers	Not directly on browsers.
Type	Scripting language	Supports OOPs like classes, interface, ~~inter~~ inheritance, etc
Number, String	Objects	Interface

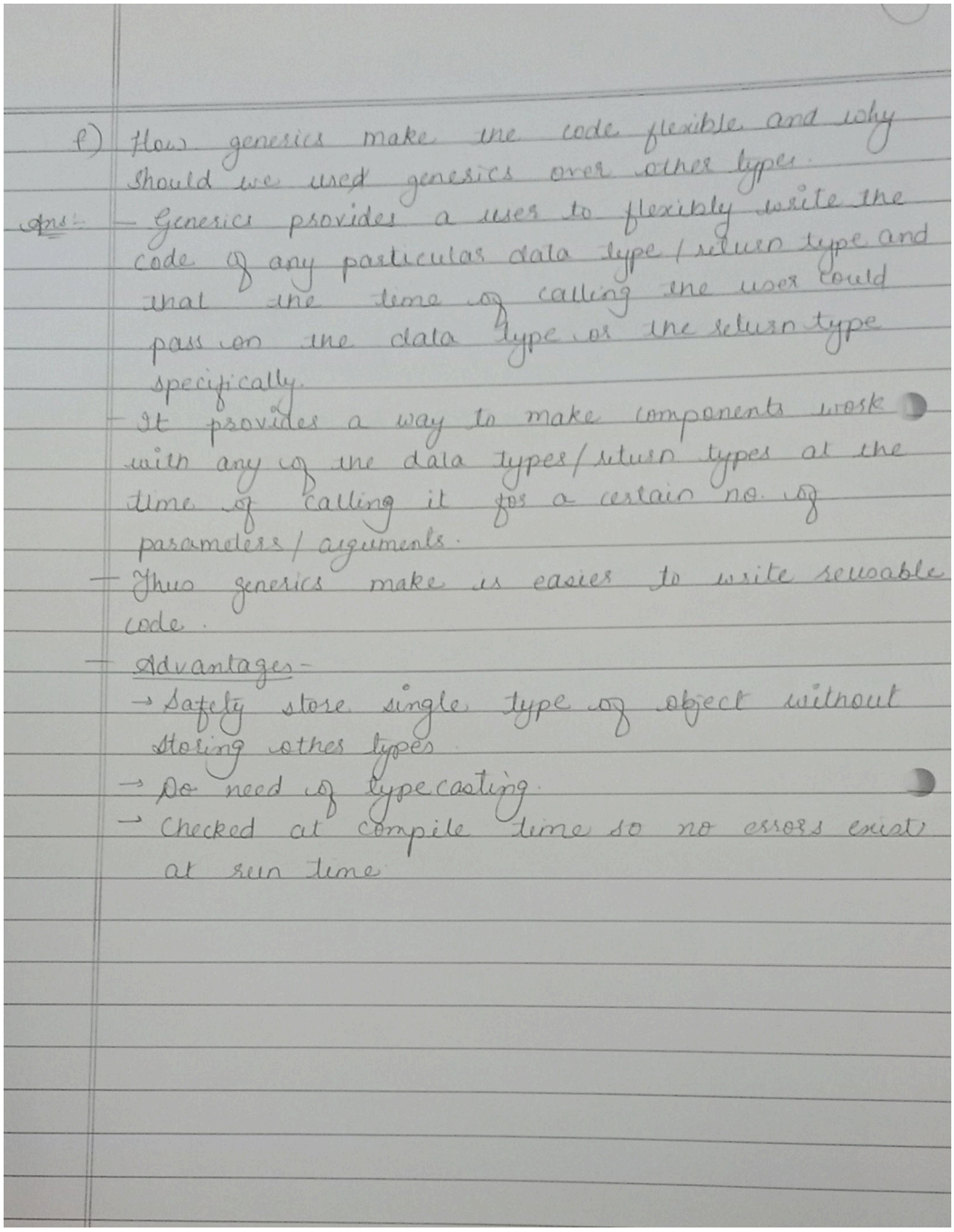
d) Compare how Javascript and Typescript implement Inheritance.

Ans: - Before ES6, JS used prototypal inheritances, using 'call', 'apply' to call base constructor.
- while, Typescript has been using 'Super' to call base constructor since before ES6.

e) what is the difference between Classes and Interfaces in Typescript? where are interfaces used?

Ans:

	Interface	Classes.
Keyword	interface	class
definition	Defines a structure which acts as a construct in the app contains only declaration.	Fundamental elements used to create reusable entities.
Usage	Create a structure for an entity.	Object creation, encapsulation for fields, methods.
mpile	Completely disappears	Cannot disappear.
ccess cifier	Public	Public, protected, private
structor	Cannot use	Can have.



4. **Output:**

   Code:

```
import * as readlineSync from 'readline-sync';

function calName():void {

    console.log('Welcome To The Calculator!')
```

```
}
function getUserInput(prompt: string): string {
  return readlineSync.question(prompt);
}
function throwError(errorMsg: string): never {
    console.error(errorMsg);
}
calName();
console.log(`OPTIONS-
1. Add
2. Subtract
3. Multiply
4. Divide
5. Exponent`)

const option : any = getUserInput('Enter your option:' );
let opt = parseInt(option);

const num1: string = getUserInput('Enter first number: ');

const num2: string = getUserInput('Enter second number: ');
    switch(opt) {
        case 1: {
        console.log(parseInt(num1) + parseInt(num2));
           break;
```

```
}
case 2: {
    console.log(parseInt(num1) - parseInt(num2));
   break;
}
case 3: {
    console.log(parseInt(num1) * parseInt(num2));
   break;
}
case 4: {
    if (parseInt(num2)==0){
        throwError('Denominator cannot be 0');
    }
    else{
        console.log(parseInt(num1) / parseInt(num2));
    }
   break;
}
case 5: {
    console.log(parseInt(num1) ** parseInt(num2));
   break;
}
default: {
   console.log('Option invalid')
   break;
```

```
    }
}
```

```
PS C:\Users\Preeti Khamkar\OneDrive\Desktop> ts-node cal.ts
Welcome To The Calculator!
OPTIONS-
1. Add
2. Subtract
3. Multiply
4. Divide
5. Exponent
Enter your option:1
Enter first number: 2
Enter second number: 3
5
```

```
Enter your option:2
Enter first number: 4
Enter second number: 5
-1
```

```
Enter your option:3
Enter first number: 4
Enter second number: 5
20
```

```
Enter your option:4
Enter first number: 6
Enter second number: 3
2
```

```
Enter your option:5
Enter first number: 3
Enter second number: 3
27
```

2)

```
class users{
    private username: string = 'abc';
    protected email: string = 'abc@gmail.com';

    public get getUserInfo(){
        return this.username, this.email;
    }
```

```
}
class Admins extends users{
    private adminName: string = 'admin1';
    protected adminEmail: string= 'admin@gmail.com';

    public grantAccess(user:users){
                            console.log(`AdminName:    ${this.adminName},    AdminEmail:
${this.adminEmail}`)
        return user
    }
}
class dataManage{
    private data:string= 'some private data'
    protected getData(admin:Admins){
        return admin
    }
    public gD(getData:any){
        console.log(`${this.adminName}`)
        return getData
    }
}
let u1 = new users();
let uN = u1.getUserInfo;
console.log(uN)

let u2 = new Admins();
```

```
let ad = u2.grantAccess(u1);
console.log(ad)

let u3 = new dataManage();
let dm = u3.gD(u2);
```

```
PS C:\Users\Preeti Khamkar\OneDrive\Desktop\typescript> npx ts-node 2.2.ts
abc@gmail.com
AdminName: admin1, AdminEmail: admin@gmail.com
users { username: 'abc', email: 'abc@gmail.com' }
```

3)

```
// Define an interface
interface Shape<T> {
    calculateArea(): T;
    calculatePerimeter(): T;
}
// circle
class Circle implements Shape<number> {
    constructor(private readonly radius: number) {}
    calculateArea(): number {
        return Math.PI * this.radius ** 2;
    }
    calculatePerimeter(): number {
        return 2 * Math.PI * this.radius;
```

```
    }
}
// rectangle
class Rectangle implements Shape<any> {
    constructor(private readonly width: any, private readonly height: any) {}
    calculateArea(): number {
        console.log("The may or may not be corect!")
        return this.width * this.height;
    }
    calculatePerimeter(): number {
        return 2 * (this.width + this.height);
    }
}
//  triangle
class Triangle implements Shape<number> {
       constructor(private readonly sideA: number, private readonly sideB: number, private readonly sideC: number) {}
    calculateArea(): number {
        const s = (this.sideA + this.sideB + this.sideC) / 2;
        return Math.sqrt(s * (s - this.sideA) * (s - this.sideB) * (s - this.sideC));
    }
    calculatePerimeter(): number {
        return this.sideA + this.sideB + this.sideC;
    }
}
// Example usage
```

```
const circle = new Circle(5);

console.log("Circle Area:", circle.calculateArea());

console.log("Circle Perimeter:", circle.calculatePerimeter());

const rectangle = new Rectangle("4", "6");

console.log("Rectangle Area:", rectangle.calculateArea());

console.log("Rectangle Perimeter:", rectangle.calculatePerimeter());

const triangle = new Triangle(3, 4, 5);

console.log("Triangle Area:", triangle.calculateArea());

console.log("Triangle Perimeter:", triangle.calculatePerimeter());
```

```
PS C:\Users\Preeti Khamkar\OneDrive\Desktop\typescript> npx ts-node 2.2.ts
Circle Area: 78.53981633974483
Circle Perimeter: 31.41592653589793
The may or may not be corect!
Rectangle Area: 24
Rectangle Perimeter: 92
Triangle Area: 6
Triangle Perimeter: 12
```