Object Oriented JS

Assignment

PW SKILLS

1. Create a BankAccount class in JavaScript using Object-Oriented Programming (OOP) principles. The class should have methods for depositing, withdrawing, and checking the account balance. Ensure that account balances cannot go below zero and that both deposit and withdrawal amounts must be positive.

```
const myAccount = new BankAccount('Mithun S', 1000);

myAccount.checkBalance(); // Account balance for Mithun S: Rs.1000

myAccount.deposit(500); // Deposited Rs.500. New balance: Rs.1500
myAccount.deposit(-50); // Invalid deposit amount. Please enter a positive amount.

myAccount.withdraw(200); // Withdrawn Rs.200. New balance: Rs.1300
myAccount.withdraw(1500); // Insufficient funds for withdrawal.
myAccount.withdraw(-500); // Invalid withdrawal amount. Please enter a positive amount.

myAccount.checkBalance(); // Account balance for Mithun S: Rs.1300
```

2. Create a student enrollment system using JavaScript classes. Implement two classes, Student and Admission, to manage student enrollments, course admission, and enrollment tracking. The Student class should have methods for enrolling in courses and displaying enrolled courses, while the Admission class should handle student enrollments, course admission, and the display of enrolled students. Your task is to implement these classes with clear and organized code, adhering to the specified requirements provided.

```
const admissionOffice = new Admission();

const student1 = new Student('Mithun', 'mithun@pw.live');
const student2 = new Student('Farman', 'farman@pw.live');

admissionOffice.enrollStudent(student1); // Mithun has been enrolled.
admissionOffice.enrollStudent(student2); // Farman has been enrolled.

admissionOffice.assignCourse(student1, 'Full Stack Web Development'); // Mithun has enrolled in Full Stack Web Development.
admissionOffice.assignCourse(student2, 'Data Science Masters'); // Farman has enrolled in Data Science Masters.

student1.showCourses(); // Mithun's enrolled courses: Full Stack Web Development
student2.showCourses(); // Farman's enrolled courses: Data Science Masters

admissionOffice.showEnrolledStudents();
// Enrolled students:
// - Mithun (mithun@pw.live)
// - Farman (farman@pw.live)
```

3. Create a Temperature class in JavaScript that manages both Celsius and Fahrenheit temperatures with getters and setters for each unit, allowing for automatic conversion between the two scales. The class should have an initial state of 0°C and 32°F, and it should validate input values for temperature updates, ensuring they are numeric. When setting the temperature in either Celsius or Fahrenheit, the class should correctly update the corresponding value in the other scale. For example, setting the Celsius temperature to 25°C should automatically update the Fahrenheit temperature to 77°F. Your challenge is to implement the Temperature class following the provided code structure and requirements while ensuring that it handles conversions and input validation accurately.

```
const temperature = new Temperature();

console.log(`Initial Celsius: ${temperature.getCelsius}°C`); // Initial Celsius: 0°C
console.log(`Initial Fahrenheit: ${temperature.getFahrenheit}°F`); // Initial Fahrenheit: 32°F

temperature.setCelsius = 25;
console.log(`Celsius: ${temperature.getCelsius}°C`); // Celsius: 25°C
console.log(`Fahrenheit: ${temperature.getFahrenheit}°F`); // Fahrenheit: 77°F

temperature.setFahrenheit = 68;
console.log(`Celsius: ${temperature.getCelsius}°C`); // Celsius: 20°C
console.log(`Fahrenheit: ${temperature.getFahrenheit}°F`); // Fahrenheit: 68°F
```

4. Develop a set of classes in JavaScript for calculating the area and perimeter of various shapes. Begin with a base class Shape that provides default methods for area and perimeter calculation and includes subclasses like Circle, Rectangle, and Triangle. Each subclass should inherit from Shape and override the area and perimeter calculation methods to provide accurate results for their respective shapes.

```
const circle = new Circle(5);
console.log(`Circle - Area: ${circle.calculateArea()}, Perimeter: ${circle.calculatePerimeter()}`);
// Circle - Area: 78.53981633974483, Perimeter: 31.41592653589793

const rectangle = new Rectangle(4, 6);
console.log(`Rectangle - Area: ${rectangle.calculateArea()}, Perimeter: ${rectangle.calculatePerimeter()}`);
// Rectangle - Area: 24, Perimeter: 20

const triangle = new Triangle(8, 6, 5, 7, 10);
console.log(`Triangle - Area: ${triangle.calculateArea()}, Perimeter: ${triangle.calculatePerimeter()}`);
// Triangle - Area: 24, Perimeter: 22
```

5. Create an Inventory system in JavaScript to manage products using prototypes. Implement a Product constructor function that defines the properties of a product such as name, category, price, and stock. Develop an Inventory constructor function that initializes an empty inventory array. Extend the Inventory prototype with a method addProduct that adds a valid Product object to the inventory and logs a confirmation message. Additionally, implement a method deleteProduct that removes a product from the inventory by name and logs whether the deletion was successful.

```
const inventory = new Inventory();

const product1 = new Product('Laptop', 'Electronics', 899, 10);
const product2 = new Product('Book', 'Books', 20, 50);

inventory.addProduct(product1); // Added Laptop to the inventory.
inventory.addProduct(product2); // Added Book to the inventory.

inventory.deleteProduct('Laptop'); // Deleted Laptop from the inventory.
```