Exercise 1: Library System

Problem Description:

Create a simple system for managing a library's collection of books. Each book has a title, author, and a boolean indicating availability.

Requirements:

1. Create a `Book` class with `title` (string), `author` (string), and `isAvailable` (boolean) properties.
2. Add a method `borrowBook()` that sets `isAvailable` to `false` if it's currently available. If not, print `"Book is currently unavailable."`
3. Add a method `returnBook()` that sets `isAvailable` back to `true`.
4. Create a `Library` class that stores an array of `Book` objects.
5. Add a method `addBook(book)` to add a book to the library.
6. Add a method `listAvailableBooks()` that prints the titles of all books that are currently available.

Input:

```
const library = new Library();

const book1 = new Book("Things fall apart", "Chinua Achebe", true);
const book2 = new Book("Atomic Habits", "Ryan Dall", true);
const book3 = new Book("Dream Count", "Chimamanda", false);

library.addBook(book1);
library.addBook(book2);
library.addBook(book3);

book1.borrowBook();
book3.borrowBook();

console.log("Available Books:");
library.listAvailableBooks();
```

Exercise 2: Employee and Payroll System

Problem Description:

Create a class system to manage employees and calculate their monthly salaries, including bonusess

Requirements:

1. Create a class `Employee` with `name` (string), `baseSalary` (number), and `bonus` (number, default to 0).
2. Add a method `getMonthlySalary()` that returns the total of `baseSalary` and `bonus`.
3. Create a class `Payroll` that holds an array of `Employee` objects.
4. Add a method `addEmployee(employee)` to add employees.

5. Add a method `generateReport()` that prints each employee's name and their total monthly salary.
6. Print the total payroll cost for all employees.

Input:

```
const payroll = new Payroll();

const emp1 = new Employee("John", 3000, 500);
const emp2 = new Employee("Jane", 4000);
const emp3 = new Employee("Alice", 3500, 700);

payroll.addEmployee(emp1);
payroll.addEmployee(emp2);
payroll.addEmployee(emp3);

payroll.generateReport();
```

Exercise 3: Task Manager with Due Dates

Problem Description:

design a task management system that tracks tasks, their completion status, and checks items that are overdue

Requirements:

1. Create a class `Task` with `title`, `dueDate` (Date object), and `completed` (default `false`).
2. Add a method `markCompleted()` to mark the task as completed.
3. Add a method `isOverdue()` that returns `true` if the task is not completed and the due date is in the past.
4. Create a class `TaskManager` that stores a list of tasks.
5. Add methods `addTask(task)`, `listOverdueTasks()`, and `listPendingTasks()`.

Input:

```
const manager = new TaskManager();

const task1 = new Task("Submit project", new Date("2025-05-01"));
const task2 = new Task("Pay bills", new Date("2025-05-15"));
const task3 = new Task("Read book", new Date("2025-04-30"));

task2.markCompleted();

manager.addTask(task1);
manager.addTask(task2);
manager.addTask(task3);

console.log("Overdue Tasks:");
```

```
manager.listOverdueTasks();

console.log("Pending Tasks:");
manager.listPendingTasks();
```

Exercise 4: Online Course Enrollment System

Problem Description:

Create a system to manage students and course enrollments, make sure students don't enroll twice in the same course.

Requirements:

1. Create a `Student` class with `name` and an array `courses`.
2. Add a method `enroll(courseName)` that adds the course only if not already enrolled.
3. Add a method `listCourses()` to display enrolled courses.
4. Create a `Course` class with `name` and `students` (array of student names).
5. Add a method `addStudent(student)` that adds the student name if not already enrolled.

Input:

```
const student = new Student("Joseph");
student.enroll("Math");
student.enroll("Science");
student.enroll("Math"); // Duplicate

const course = new Course("Math");
course.addStudent("Joseph");
course.addStudent("Joseph"); // Duplicate

console.log("Student Courses:");
student.listCourses();

console.log("Course Students:");
console.log(course.students);
```

Exercise 5: Simple Ecommerce Cart with Discounts

Problem Description:

Simulate a shopping cart that can apply discounts and compute final totals.

Requirements:

1. Create a `CartItem` class with `name`, `price`, `quantity`.
2. Add a method `getTotal()` returning `price * quantity`.

3. Create a `Cart` class with a list of `CartItem`s.
4. Add a method `addItem(item)` to add to the cart.
5. Add a method `applyDiscount(percent)` that reduces the total cart value by the percentage.
6. Add a method `getFinalTotal()` that returns total after discount.

Input:

```
const cart = new Cart();

const item1 = new CartItem("Phone", 600, 1);
const item2 = new CartItem("Charger", 25, 2);
const item3 = new CartItem("Case", 15, 3);

cart.addItem(item1);
cart.addItem(item2);
cart.addItem(item3);

cart.applyDiscount(10); // 10% discount

console.log("Final Total:", `$${cart.getFinalTotal().toFixed(2)}`);
```