```java
import java.time.DayOfWeek;
import java.time.LocalTime;
import java.util.EnumMap;
import java.util.Scanner;

public class ScheduleManager {
    public static class TimeInterval {
        private final LocalTime openingHour;
        private final LocalTime closingHour;

        public TimeInterval(LocalTime openingHour, LocalTime closingHour) {
            this.openingHour = openingHour;
            this.closingHour = closingHour;
        }

        public LocalTime getOpeningHour() {
            return openingHour;
        }

        public LocalTime getClosingHour() {
            return closingHour;
        }

        @Override
        public String toString() {
            return openingHour + " - " + closingHour;
        }
    }

    public enum TicketType {
        CHILD, STUDENT, ADULT, SENIOR
    }

    public static boolean isOpen(EnumMap<DayOfWeek, TimeInterval> schedule, DayOfWeek dayOfWeek,
                                 LocalTime time) {
        TimeInterval interval = schedule.get(dayOfWeek);
        return time.isAfter(interval.getOpeningHour()) && time.isBefore(interval.getClosingHour());
    }

    public static double calculatePrice(LocalTime time, TicketType ticketType) {
        double basePrice = 20.0;
        if (time.isBefore(LocalTime.NOON)) {
            basePrice -= 10.0; // Morning discount
        } else if (time.isAfter(LocalTime.of(17, 0))) {
            basePrice += 10.0;
        }

        double price;
        switch (ticketType) {
            case CHILD:
                price = 0.0;
                break;
            case STUDENT:
                price = basePrice * 0.25;
                break;
            case SENIOR:
                price = basePrice * 0.50;
                break;
            default:
                price = basePrice;
                break;
        }
        return price;
    }

    // Other methods and fields
    public static void main(String[] args) {
```

```java
EnumMap<DayOfWeek, TimeInterval> schedule = new EnumMap<>(DayOfWeek.class);
EnumMap<DayOfWeek, Double> revenue = new EnumMap<>(DayOfWeek.class);

// Set up opening and closing hours for each day
schedule.put(DayOfWeek.MONDAY, new TimeInterval(LocalTime.of(9, 0), LocalTime.of(18, 0)));
schedule.put(DayOfWeek.TUESDAY, new TimeInterval(LocalTime.of(11, 0), LocalTime.of(19, 0)));
schedule.put(DayOfWeek.WEDNESDAY, new TimeInterval(LocalTime.of(8, 0),
                                                   LocalTime.of(17, 0)));
schedule.put(DayOfWeek.THURSDAY, new TimeInterval(LocalTime.of(9, 0), LocalTime.of(20, 0)));
schedule.put(DayOfWeek.FRIDAY, new TimeInterval(LocalTime.of(10, 0), LocalTime.of(18, 0)));

// Initialize revenue map
for (DayOfWeek day : DayOfWeek.values()) {
    revenue.put(day, 0.0);
}

Scanner scanner = new Scanner(System.in);
while (true) {
    System.out.println("Enter a command (purchase, stats, exit): ");
    String command = scanner.nextLine();

    if (command.equalsIgnoreCase("exit")) {
        break;
    } else if (command.equalsIgnoreCase("purchase")) {
        System.out.println("Enter day of week (e.g., MONDAY): ");
        DayOfWeek day = DayOfWeek.valueOf(scanner.nextLine().toUpperCase());

        System.out.println("Enter time of arrival (HH:MM): ");
        LocalTime time = LocalTime.parse(scanner.nextLine());

        System.out.println("Enter ticket type (CHILD, STUDENT, ADULT, SENIOR): ");
        TicketType ticketType = TicketType.valueOf(scanner.nextLine().toUpperCase());

        // Check if open
        if (!isOpen(schedule, day, time)) {
            System.out.println("The purchase is not valid for the chosen period.");
            continue;
        }

        // Calculate price
        double price = calculatePrice(time, ticketType);

        // Update revenue
        revenue.put(day, revenue.get(day) + price);
        System.out.println("Purchase registered: " + ticketType + " ticket for " + day +
                           " at " + time + ". Price: $" + price);
    } else if (command.equalsIgnoreCase("stats")) {
        double totalRevenue = 0.0;
        DayOfWeek highestDay = null, lowestDay = null;
        double highestSales = Double.MIN_VALUE, lowestSales = Double.MAX_VALUE;

        for (DayOfWeek day : schedule.keySet()) {
            double dailyRevenue = revenue.get(day);
            totalRevenue += dailyRevenue;

            if (dailyRevenue > highestSales) {
                highestSales = dailyRevenue;
                highestDay = day;
            }

            if (dailyRevenue < lowestSales) {
                lowestSales = dailyRevenue;
                lowestDay = day;
            }
        }
```

```java
                double averageSales = totalRevenue / schedule.keySet().size();

                System.out.println("Highest sales: " + highestDay + " ($" + highestSales + ")");
                System.out.println("Lowest sales: " + lowestDay + " ($" + lowestSales + ")");
                System.out.println("Average sales per day: $" + averageSales);
                System.out.println("Total revenue for the week: $" + totalRevenue);
            } else {
                System.out.println("Unknown command. Please try again.");
            }
        }

        scanner.close();
    }
}
```