# TypeScript – Exam Regular – 15 June 2025

## Monthly Motel

Your task is **implement** **the** **class** **MonthlyMotel**, which allows for monthly booking and canceling of **Rooms** during the Winter and Summer months.

### Skeleton code

You are provided with a skeleton that comes with some files.

#### Contracts

In the **/contracts** folder of your skeleton you’ll be provided with some files, that the **MonthlyMotel** class relies on or is meant to work with. You **MUST NOT add/remove or change any files** in the /contracts folder - during evaluation the **/contracts** folder in your project will be swapped with the default one from the skeleton, so **any changes you made in the folder WILL be lost**.

* **util.ts** – contains enums for the Winter and Summer Months that the Motel should work with.
* **room.ts** – contains the interface for a Room, all rooms the motel works with should be compatible to the Room interface.
* **motel.ts** – contains the interface for a Motel, the MonthlyMotel class should implement this.
* **airconditionedRoom.ts** – a provided implementation of the Room interface, to help you understand the functionality of a Room.
* **partialMonthlyMotel.ts** – an **abstract** **class** **PartialMonthlyMotel** that comes with a some implementations, your **MonthlyMotel** class should inherit from this class

#### Additional Files

You are also provided with some additional files:

* **decorators.ts** - a decorators file that features the empty decorator functions linked to contracts classes.
* **monthlyMotel.ts** - the empty implementation of MonthlyMotel
* **index.ts** - a file that you can use for testing, see the Examples section bellow

You are **free to add or modify any files** **outside of the /contracts** folder as you see fit.

#### Skeleton explanation

Aside from the interfaces in the skeleton, you are given the following implementations:

**AirconditionedRoom -** an implementation of the **Room** **interface** that will help you understand the functionality of a Room, a Room:

1. Should have a **room** **number** which is one of 6 predefined string values as specified by the Room interface
2. Should expose a **totalPrice** property/accessor of type number – the total price for booking a room for a month.
3. Should expose a **cancellationPrice** property/accessor of type number – the amount of money returned on room cancellation for a given month.
4. Should have a constructor that takes in **price** and **room number** parameters and optionally any other parameters needed for the implementation

The implementation of **AirconditionedRoom** has **totalPrice** and **cancellationPrice** equal to the received **price** value, however other room implementations may take different parameters or calculate **totalPrice** and **cancellationPrice** in different ways.

### Tasks

You will be scored based on the following 3 tasks: Functionality, Structure and Decoration.

#### Functionality (25 points)

The **MonthlyMotel** class should:

* keep track of rooms added to the motel and their bookings. Each room will **always** **have** **an** **unique** **roomNumber** (one of the strings specified in the Room interface).
* keep track of which rooms are booked and for what month, preventing a room that wasn’t added to the motel from being booked or having a room being booked twice for the same month.
* Process room cancellation, by allowing only cancellations for existing rooms and only in months where they were booked, preventing a room from being cancelled multiple times for a month.
* Keep track of a **totalBudget** (initially 0)

Implement the following methods from the **Motel** interface:

1. **Method getTotalBudget()** – returns a string showing the motel name and totalBudget in the format

**"MotelName: <MotelName>**

**Total budget: $<totalBudget>"**

Where:

* **MotelName** is the **PartialMonthlyMotel**’s **MotelName** property
* **totalBudget** is the **sum of totalPrice** for all room bookings **minus the sum of all cancellationPayments** for all room cancellations, displayed to the 2nd decimal place.

1. **Method addRoom(room)** – **adds** the provided Room to the motel, the MonthlyMotel instance **should keep track of all added Rooms**.
   1. In case the passed value is not compatible with the Room interface returns the message **"Value was not a Room."**
   2. In case a Room with the same room number was already added return the message **"Room '<roomNumber>' already exists."**
   3. If a Room with that number does not exist in the Motel, add the room to the motel and then return the message **"Room '<roomNumber>' added."**
2. **Method bookRoom(roomNumber, bookedMonth)** – **books** the Room with the given **roomNumber** in the motel for the **provided month**, the MonthlyMotel instance should **keep track of all currently booked Rooms** and the **months they are booked** **for**.
   1. In case a Room with the provided **roomNumber** hasn’t been added to the Motel, show the message **"Room '<roomNumber>' does not exist."**
   2. In case the Room has been added, but is already booked for the provided month, return the message **"Room '<roomNumber>' is already booked for '<bookedMonth>'."**
   3. In case the Room has been added and is not booked for the provided month, book the Room for the provided month and return the message **"Room '<roomNumber>' booked for '<bookedMonth>'."**
3. **Method cancelBooking(roomNumber, bookedMonth)** – **cancels the booking** of the Room with the given **roomNumber** in the motel for the **provided month**.
   1. In case a Room with the provided **roomNumber** hasn’t been added to the Motel, show the message **"Room '<roomNumber>' does not exist."**
   2. In case the Room has been added, but has not been booked for the provided month, return the message **"Room '<roomNumber>' is not booked for '<bookedMonth>'."**
   3. In case the Room has been added and is booked for the provided month, cancel the booking of the Room for the provided month and return the message **"Booking cancelled for Room '<roomNumber>' for '<bookedMonth>'."**

#### Structure (50 Points)

Your **MonthlyMotel** implementation should match the following structure requirements:

**NOTE:** These tasks will only award points if the underlying functionality also exists and works as intended.

1. The **MonthlyMotel** should inherit the **PartialMonthlyMotel** abstract class and have method signatures compatible with the **Motel** **interface**. You are free to override any methods from **PartialMonthlyMotel** as you see fit.
2. The **MonthlyMotel** class should be a **generic** **class** accepting only **WinterMonth** or **SummerMonth** as its type. Motel instances instantiated with the **SummerMonth** type should only allow booking/cancelling rooms in the summer months and instances with **WinterMonth** type, should respectively only allow booking/cancelling rooms in the winter months.
3. Ensure that any code and implementations written by you, do not use type assertions to the **any** type
4. Implement a new **Apartment** class that implements the **Room** **interface** and uses the following functionality:
   1. Its constructor accepts the following 3 parameters:
      1. **price: number**
      2. **roomNumber** – one of 6 predefined string values as specified by the Room interface
      3. **numberOfGuests: number**
   2. The **totalPrice** of an apartment is equal to the **numberOfGuests multiplied by the price**
   3. The **cancellationPrice** of an apartment is equal to **80% of its totalPrice**
5. All **literal** and **advanced types** in the **MonthlyMotel** and implementations written by you should be replaced with equivalent **type aliases** for easier readability readability – **reuse interfaces where possible**. This task requires the **Apartment** class to be implemented otherwise it rewards reduced points.
6. **All access modifiers** of properties and methods **in your code**, **that are not part of the interfaces or the provided implementations**, should use either the **private or protected** access modifiers. . This task requires the **Apartment** class to be implemented otherwise it rewards reduced points.

#### Decoration (25 Points)

You are tasked to change the functionality of 2 existing classes:

* **AirconditionedRoom** - Your task is to change the **totalPrice** and **cancellationPrice** of an airconditioned room to be equal to **120% of its price** (the price parameter passed in the constructor. You are **NOT allowed** to directly modify the files inside the **/contracts** folder.
* **PartialMonthlyMotel –** Your task is to change the **MotelName** value, the new **MotelName** should be **"Monthly Motel"**. You are **NOT allowed** to directly modify the files inside the **/contracts** folder.

**IMPORTANT:** The modifications should change the class definitions themselves – check Example 5 to see that the expected modifications are directly available on the **AirconditionedRoom** and **PartialMonthlyMotel** class definitions.

* Modifying the functionality in a derived class like **MonthlyMotel**, while the functionality remains unchanged in **PartialMonthlyMotel will award NO points.**

### Examples

This is an example of how the **MonthlyMotel** class is **intended to be used**:

* Examples 1-3 use the non modified versions of the AirconditionedRoom (**100% price**) and MotelName (MotelName = **"Motel"**)
* Example 4 shows the new **Apartment** class with the the non modified versions of the AirconditionedRoom (**100% price**) and MotelName (MotelName = **"Motel"**)
* Examples 5-6 use the modified versions of the AirconditionedRoom (**120% price**) and MotelName (MotelName = **"Monthly Motel"**)

|  |
| --- |
| **Example 1** |
| let motel = new MonthlyMotel<SummerMonth>();  let roomA02: Room = new AirconditionedRoom(130, 'A02');  let roomB01: Room = new AirconditionedRoom(50, 'B01');  console.log(motel.addRoom(roomA02));  console.log(motel.addRoom(roomB01));  console.log(motel.getTotalBudget());  console.log(motel.bookRoom('A02', SummerMonth.August));  console.log(motel.bookRoom('B01', SummerMonth.July));  console.log(motel.bookRoom('B01', SummerMonth.September));  console.log(motel.getTotalBudget());  console.log(motel.cancelBooking('B01',SummerMonth.September));  console.log(motel.getTotalBudget()); |
| **Output 1** |
| **Room 'A02' added.**  **Room 'B01' added.**  **Motel: Motel**  **Total budget: $0.00**  **Room 'A02' booked for 'August'.**  **Room 'B01' booked for 'July'.**  **Room 'B01' booked for 'September'.**  **Motel: Motel**  **Total budget: $230.00**  **Booking cancelled for Room 'B01' for 'September'.**  **Motel: Motel**  **Total budget: $180.00** |

|  |
| --- |
| **Example 2** |
| let motel = new MonthlyMotel<WinterMonth>();  let roomA01: Room = {   roomNumber: 'A01',   get totalPrice() { return 120; },   get cancellationPrice() { return 120; }  };  let roomB01: Room = new AirconditionedRoom(200, 'B01');  let roomNumberB02: 'A01' | 'A02' | 'A03' | 'B01' | 'B02' | 'B03' = 'B02';  let roomB02 = {      roomNumber: roomNumberB02,      flatDiscount: 20,      price: 320,      get totalPrice() { return this.price  - this.flatDiscount },      get cancellationPrice() { return this.totalPrice \* 0.5; }  };  console.log(motel.addRoom(roomA01));  console.log(motel.addRoom(roomA01));  console.log(motel.addRoom(roomB01));  console.log(motel.getTotalBudget());  console.log(motel.bookRoom('B02', WinterMonth.December));  console.log(motel.addRoom(roomB02));  console.log(motel.bookRoom('B02', WinterMonth.December));  console.log(motel.getTotalBudget());  console.log(motel.bookRoom('A01', WinterMonth.December));  console.log(motel.bookRoom('A01', WinterMonth.December));  console.log(motel.bookRoom('B01', WinterMonth.February));  console.log(motel.getTotalBudget());  console.log(motel.cancelBooking('B02', WinterMonth.January));  console.log(motel.cancelBooking('B02', WinterMonth.December));  console.log(motel.cancelBooking('B01', WinterMonth.February));  console.log(motel.getTotalBudget()); |
| **Output 2** |
| **Room 'A01' added.**  **Room 'A01' already exists.**  **Room 'B01' added.**  **Motel: Motel**  **Total budget: $0.00**  **Room 'B02' does not exist.**  **Room 'B02' added.**  **Room 'B02' booked for 'December'.**  **Motel: Motel**  **Total budget: $300.00**  **Room 'A01' booked for 'December'.**  **Room 'A01' is already booked for 'December'**  **Room 'B01' booked for 'February'.**  **Motel: Motel**  **Total budget: $620.00**  **Room 'B02' is not booked for 'January'.**  **Booking cancelled for Room 'B02' for 'December'.**  **Booking cancelled for Room 'B01' for 'February'.**  **Motel: Motel**  **Total budget: $270.00** |

**Note:** This example is meant to check correct typization and constraints, the comments in the output are not the result of runtime execution, but errors that TS should flag on the red text in the input.

|  |
| --- |
| **Example 3** |
| let motel = new MonthlyMotel<SummerMonth>();  console.log(motel.bookRoom('A02', WinterMonth.December));  console.log(motel.cancelBooking('A02', WinterMonth.December));  console.log(motel.bookRoom('A04', SummerMonth.August)); |
| **Output 3** |
| **//TypeScript Error: "Argument of type 'WinterMonth.December' is not assignable to parameter of type 'SummerMonth'**  **//TypeScript Error: "Argument of type 'WinterMonth.December' is not assignable to parameter of type 'SummerMonth'**  **//TypeScript Error: Argument of type '"A04"' is not assignable to parameter of type '"A01" | "A02" | "A03" | "B01" | "B02" | "B03"'** |

|  |
| --- |
| **Example 4** |
| let motel = new MonthlyMotel<WinterMonth>();  let roomA01: Room = new Apartment(110, 'A01', 4);  let roomA02: Room = new Apartment(70, 'A02', 3);  console.log(motel.addRoom(roomA01));  console.log(motel.addRoom(roomA02));  console.log(motel.bookRoom('A01', WinterMonth.March));  console.log(motel.getTotalBudget());  console.log(motel.cancelBooking('A01',WinterMonth.March));  console.log(motel.getTotalBudget());  console.log(motel.bookRoom('A02', WinterMonth.February));  console.log(motel.getTotalBudget()); |
| **Output 4** |
| **Room 'A01' added.**  **Room 'A02' added.**  **Room 'A01' booked for 'March'.**  **Motel: Motel**  **Total budget: $440.00**  **Booking cancelled for Room 'A01' for 'March'.**  **Motel: Motel**  **Total budget: $88.00**  **Room 'A02' booked for 'February'.**  **Motel: Motel**  **Total budget: $298.00** |

|  |
| --- |
| **Example 5** |
| let roomA01: Room = new AirconditionedRoom(100, 'A01');  console.log(roomA01.totalPrice);  console.log(roomA01.cancellationPrice);  console.log(PartialMonthlyMotel.MotelName); |
| **Output 5** |
| **120**  **120**  **Monthly Motel** |

|  |
| --- |
| **Example 6** |
| let motel = new MonthlyMotel<SummerMonth>();  let roomA01: Room = new AirconditionedRoom(100, 'A01');  let roomA02: Room = new AirconditionedRoom(80, 'A02');  console.log(motel.addRoom(roomA01));  console.log(motel.addRoom(roomA02));  console.log(motel.bookRoom('A01', SummerMonth.August));  console.log(motel.bookRoom('A02', SummerMonth.June));  console.log(motel.getTotalBudget());  console.log(motel.cancelBooking('A01', SummerMonth.August));  console.log(motel.getTotalBudget()); |
| **Output 6** |
| **Room 'A01' added.**  **Room 'A02' added.**  **Room 'A01' booked for 'August'.**  **Room 'A02' booked for 'June'.**  **Motel: Monthly Motel**  **Total budget: $216.00**  **Booking cancelled for Room 'A01' for 'August'.**  **Motel: Monthly Motel**  **Total budget: $96.00** |