**Design Rationale**

**Introduction**

In the design of a booking system, object-oriented programming (OOP) principles, particularly inheritance and polymorphism, play crucial roles. The application of these principles is clearly delineated in the UML diagram provided, which shows how the classes are related and interact within the system.

**Object-Oriented Design Considerations**

**Inheritance**

Definition and Application: Inheritance is a cornerstone of OOP that allows new classes to derive from existing ones. In our booking system, GroupBooking and IndividualBooking are subclasses that inherit from the superclass Booking.

**Benefits:**

Code Reusability: Shared attributes and methods like booking dates, customer details, and cancelation policies are defined once in the Booking class and reused in the subclasses, reducing redundancy.

Simplified Modifications: Changes made to the Booking class propagate automatically to GroupBooking and IndividualBooking, easing maintenance and updates.

**Polymorphism**

Definition and Application: Polymorphism permits classes to have methods with the same name but different implementations. In the system, the ToString() method is overridden in both GroupBooking and IndividualBooking to provide customized string representations.

**Benefits:**

Flexibility: Enables methods to perform differently based on the subclass instance, increasing the system's flexibility.

Scalability: New booking types can implement their own ToString() method, allowing the system to expand seamlessly without altering existing code.

Advantages of Object-Oriented Design

Utilizing OOP principles such as inheritance and polymorphism in the booking system provides several advantages:

Code Reusability and Extensibility: It's easier to add new booking types by extending the Booking class, promoting rapid development and extension.

Maintainability: Centralized updates in the base class improve consistency and decrease error rates across subclasses.

Flexibility and Scalability: Polymorphic methods allow the system to adapt more dynamically to future requirements, such as integrating new types of bookings or modifying business logic without disrupting existing functionalities.

**Conclusion**

The structured use of OOP principles enhances the robustness, maintainability, and scalability of the booking system. The UML diagram illustrates these relationships and operational dynamics, confirming the system’s capability to manage various booking types efficiently while remaining adaptable for future enhancements.

**UML Diagram**

**A diagram of a computer

Description automatically generated with medium confidence**