GROUP PROJECT

GROUP 6

GROUP MEMBERS

- KISA EMMANUEL
- ARAPTA HILDA
- MUCUNGUZI GODFREY
- ABSOLOM ORIANGA
- TENDO CALVIN

INTRODUCTION

In this project, we have developed a Hotel Reservation System designed using Object-Oriented Programming (OOP) principles. The primary objective of this project is to demonstrate the implementation of the four key pillars of OOP: Encapsulation, Abstraction, Inheritance, and Polymorphism, in Python programming. The system manages hotel rooms, guest reservations, availability checks, and payment processing in an intuitive and efficient manner, ensuring ease of use for the hotel management team.

OBJECTIVES

1 Our main goal is to implement all four pillars of OOP in the design and development of a functional hotel reservation system.

2 To develop a functioning system.

BACKGROUND

In the past, hotel reservations were largely done manually — either through phone calls, face-to-face interactions, or via written records. While these traditional methods have served their purpose, they come with several drawbacks. Our Hotel Reservation System, will offers a modern, efficient, and error-free alternative that significantly improves the process of managing hotel bookings.

METHODOLOGY

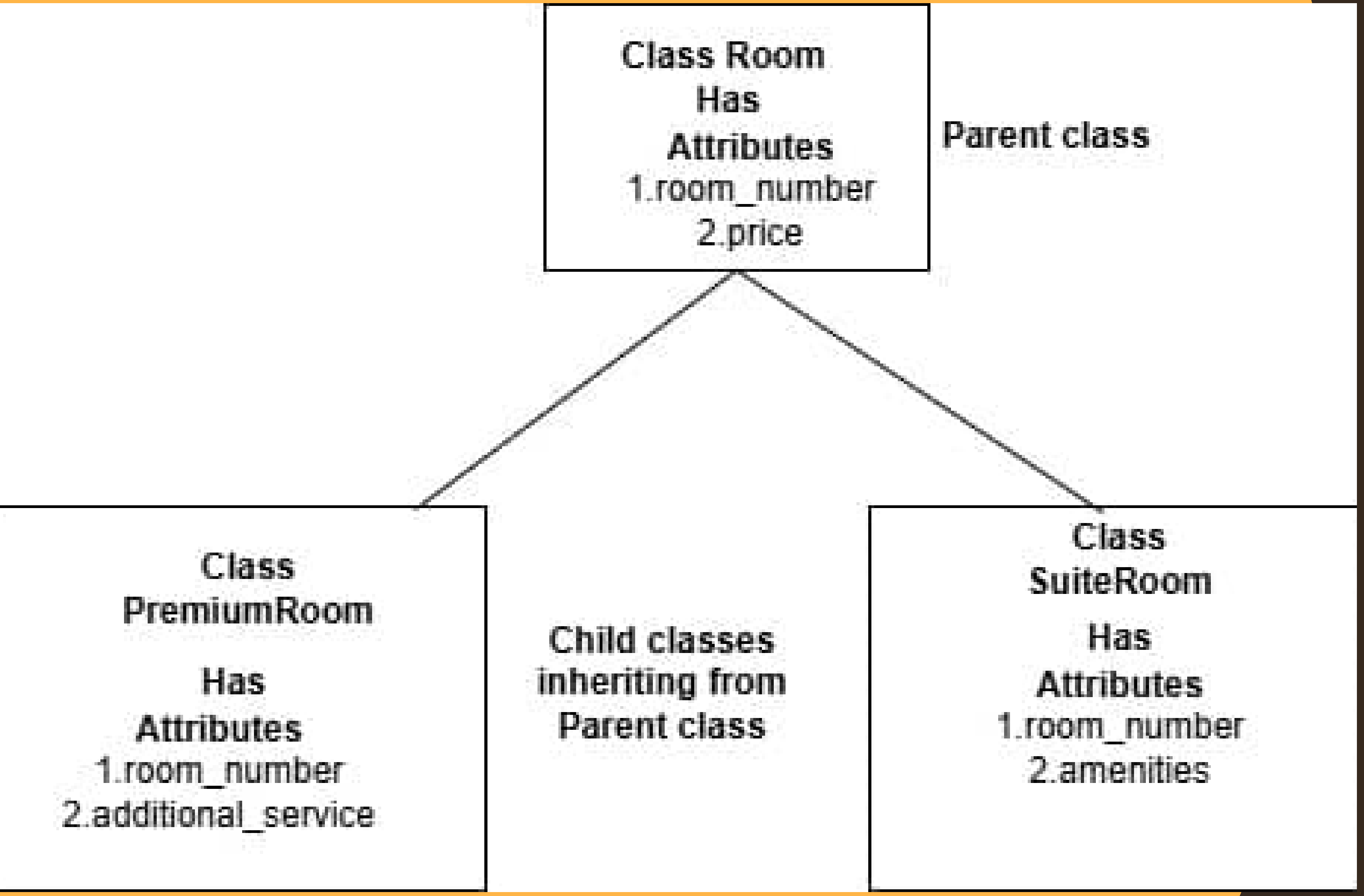
1

The first step in the methodology involves gathering all the fuctional and non-functional requirements for the hotel reservation system. This includes defining system features like user authentication (login for system access), room management (adding and removing roooms), reservation system (checking room availability, and reservations), guest management (storing guest information) and payment processing

2

The second involves the design of the system based on the gathered requirements, ensuring that the system is scalable, maintainable, and efficient. In this phase, we primarily focus on defining classes e.g Guest, Room, Hotel and Reservation and their relationships using the **principles of Object-Oriented Programming**

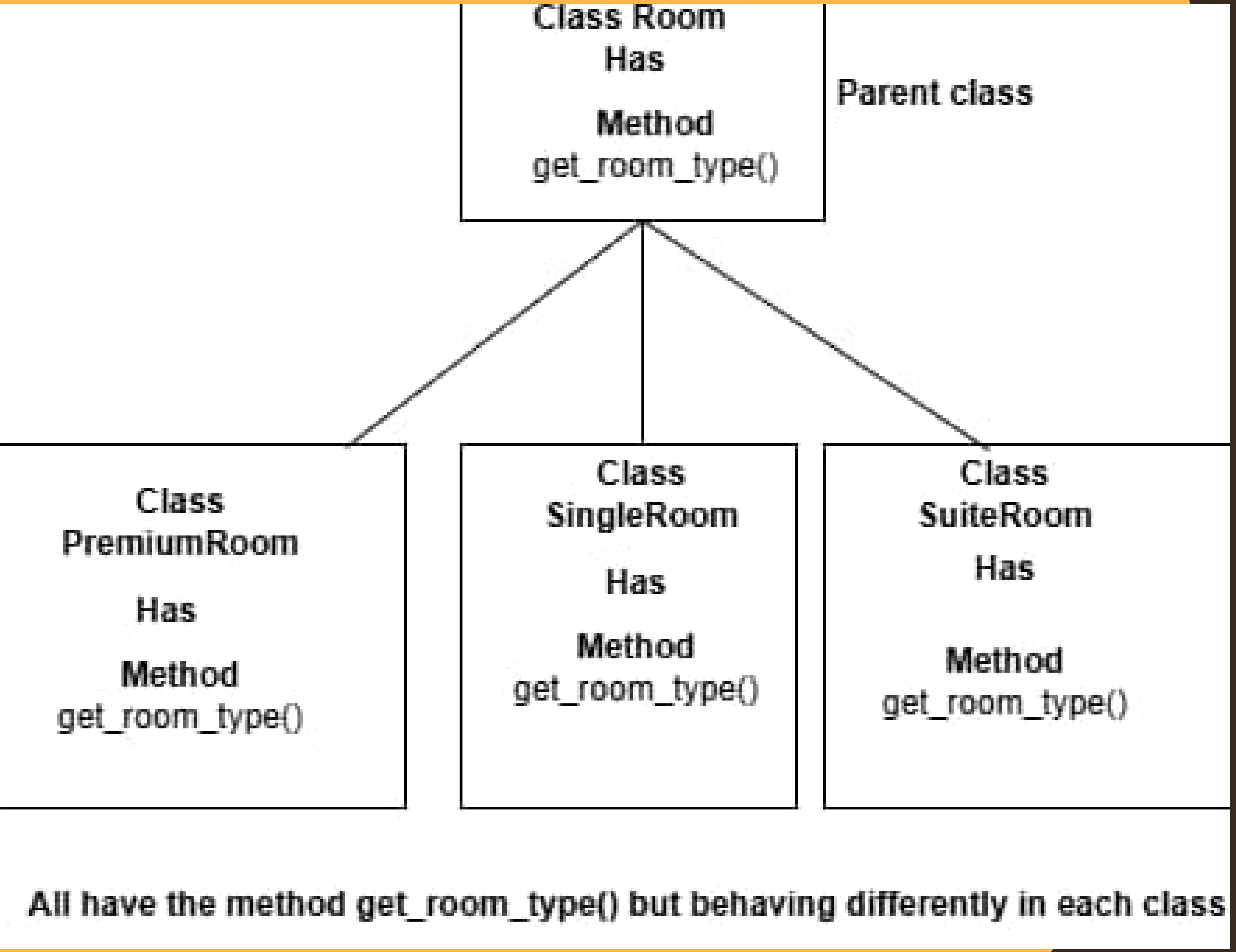
INHERITANCE



Inheritance allows classes to inherit methods and properties from a parent class. In the the project, it was used to create a hierarchy of different types of rooms and payment methods.

- Room Class is the base (parent) class, which defines common attributes and methods for all room types, such as room_number, price, and methods like check_availability, book_room, release_room, etc.
- PremiumRoom, SuiteRoom, and SingleRoom are subclasses (children of the Room class). Each subclass inherits from Room and can access its methods and attributes but also adds specific behavior or attributes.
    - PremiumRoom adds additional_service.
    - SuiteRoom adds amenities.
    - SingleRoom is simpler and does not add any extra features.

POLYMORPHISM

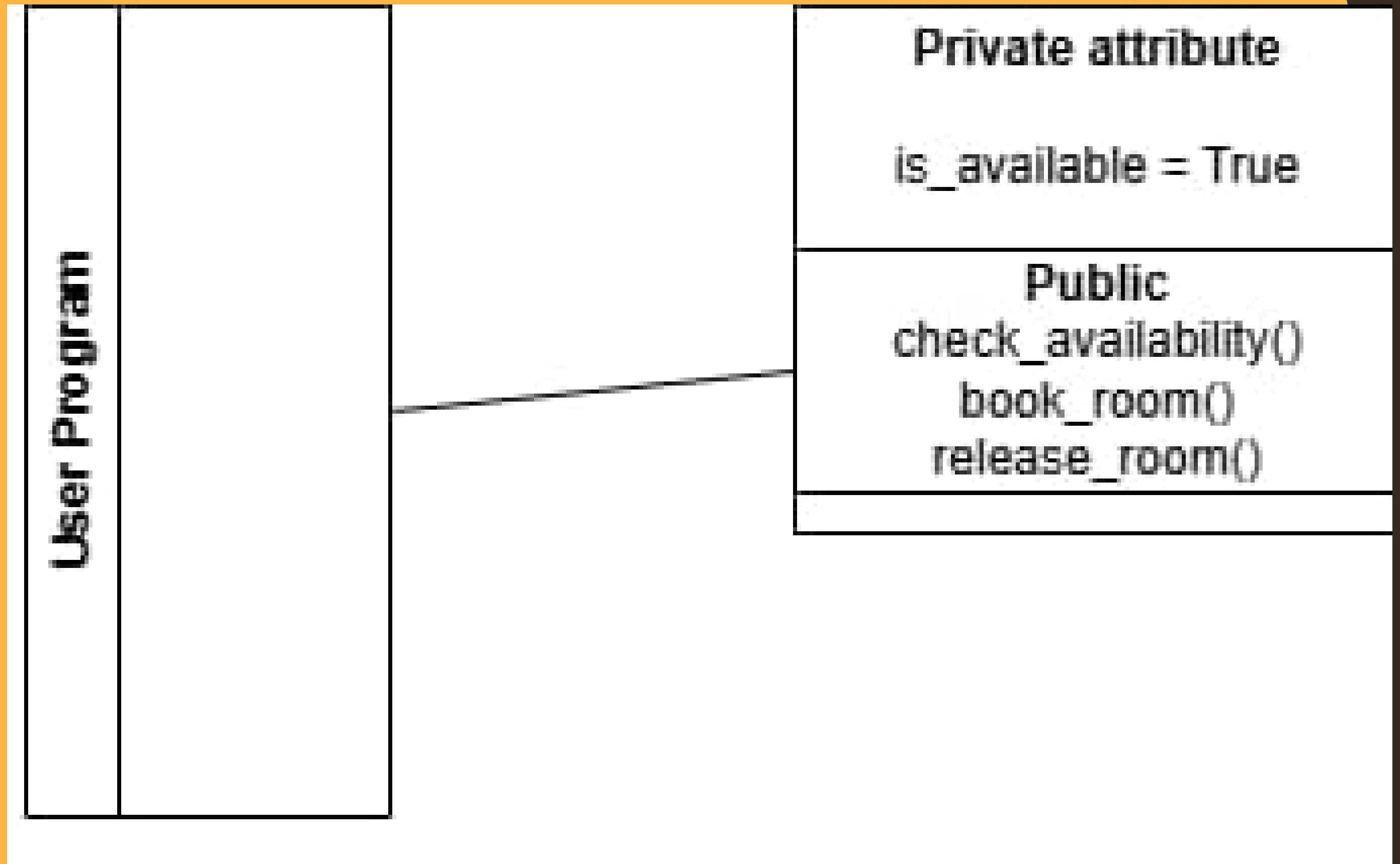


Polymorphism refers to the ability of objects to take on different forms. It allows methods to behave differently depending on the object calling them.

This concept has used;

- The get_room_type() method in the Room class. Although this method is defined in the parent Room class, it is overridden in each subclass (PremiumRoom, SuiteRoom, SingleRoom) to return a specific type of room:
- The pay() method in the PaymentMethod class hierarchy. Each subclass of PaymentMethod implements its own version of the pay method, demonstrating polymorphism. When the user selects a payment method, the corresponding pay() method for that type of payment is called.

ABSTRACTION

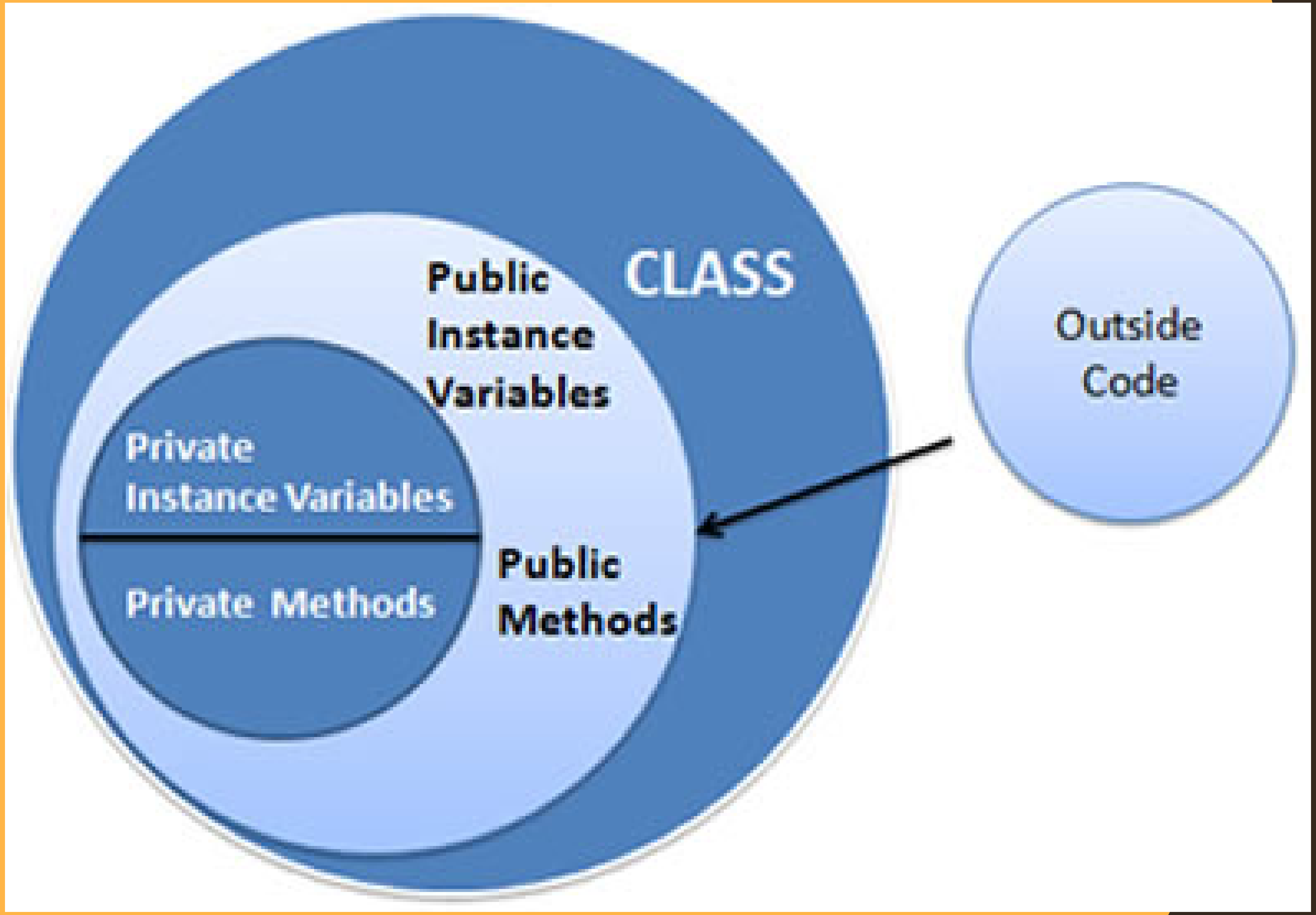


The user wiil be able to interract with the public methods such as book_room() without necessary knowing how the rooms are managed internally.

Abstraction is the concept of hiding complex implementation details and showing only the necessary functionality. It simplifies the interface by focusing on essential aspects and hiding unnecessary ones.

- For example we see this concept in use in the Room class. It defines the essential methods like check_availability(), book_room(), and release_room(), but it hides the details of how the rooms are managed internally (like the __is_available attribute). This allows other parts of the system to interact with rooms without needing to understand how they are implemented.
- The also PaymentMethod class is another example of abstraction. It defines the pay() method but does not provide its implementation. The specific payment methods (Cash, Credit Card, Mobile Money) provide their own implementations of the pay() method. Thus, clients interacting with payment systems only need to know about the pay() method and not the details of how each payment method works.

ENCAPSULATION



Encapsulation is the concept of bundling data (attributes) and methods (functions) that operate on the data into a single unit, usually a class, and restricting direct access to some of the object’s components.

Private attributes such as __room_number, __price, and __is_available in the Room class meant that they cannot be accessed directly outside of this class. For instance, if you try to access room.__price from outside the class, Python will raise an error because these attributes are not accessible directly. And to be enable to access the above attributes, a modification was made to these private attributes but using public methods (e.g., get__price, get__room_number) to access this information.

CONCLUSION

This project has provided valuable insights into the practical application of the four key pillars of Object-Oriented Programming (OOP). Through the development of the Hotel Reservation System, we have demonstrated how Encapsulation, Abstraction, Inheritance, and Polymorphism can be effectively implemented in Python to create a robust, efficient, and user-friendly system. By managing hotel rooms, guest reservations, availability checks, and payment processing, this project highlights how OOP principles can simplify complex tasks and ensure maintainability and scalability in real-world applications.

THANK YOU ALL FOR LISTENING

GROUP 6