**Project Summary**

## **Project Purpose**

* In this simulation project, we aimed to replicate the operations of a hotel, focusing on the interactions between guests, front desk employees, and bellhops. The project required the implementation of a multithreaded environment, where each thread represented a guest, front desk employee, or bellhop. The objective was to create a realistic and coordinated environment where guests check into the hotel, obtain room assignments, drop off bags if needed, and eventually retire for the evening. This project allowes us to explore multithreading, coordination, and synchronization using semaphores.

## **Project Implementation**

* The project was implemented using Java and the Java Collections Framework. It involved the creation of multiple types to threads, including guest threads, front desk employee threads, and bellhop threads. Semaphores were used to coordinate and synchronize their activities. Each guest had specific tasks, such as checking in, receiving a room, and interacting with bellhops. The front desk employees assigned rooms to guests, while bellhops handled guests’ luggage. The simulation was designed to maintain mutual exclusion, munimize busy waiting, and ensure each thread printed its own activities.
* Throughout the project, I faced several challenges:
  + Coordinating the activities of multiple threads to achieve a realistic and synchronized sequence of events was challenging. I overcame this by using Semaphores to manage access to shared resources and control the order of operations.
  + One significant challenges was ensuring that all threads, including the front desk employees and bellhops, terminated gracefully once all guests had completed their stay. I relied on Semaphores and thread joining to address this issue.
  + Ensuring that the simulation’s output matched the specified requirements and included accurate information was essential.

## **Personal Experience**

* Developing this project was a valuable learning experience. It provided insights into multithreading, semaphores, and thread coordination. I faced several challenges during development, such as ensuring that threads terminated correctly, coordinating the orderly termination of threads, and handling concurrency issues. Debugging the program was particularly challenging due to the nature of multithreading. Through this project, I gained a deeper understanding of how to create and coordinate multiple threads in Java, as well as how to use semaphores for synchronization. It reinforced the importance of minimizing mutual exclusion to maximize concurrency and the significance of proper coordination to prevent race conditions. Additionally, I learned about the limitations of using a for loop to join threads when they have different execution times.