# Ski Rental Shop Application

## Introduction

This guide provides step-by-step instructions for application programmers on how to implement and use the classes designed for the ski rental shop application.

# Class Overview

### Customer Class

- \*\*Purpose\*\*: The purpose for the customer is to efficiently browse and rent ski and snowboard equipment from the shop. They can view the available inventory of skis and snowboards, choose the rental duration that suits their needs (hourly, daily, weekly), and receive an estimate of the rental cost before finalizing the transaction. Customers can also take advantage of promotions like family rentals for groups and discounts with valid coupons. By providing their name and ID number, they streamline the rental process and ensure accurate billing. Ultimately, the goal for the customer is to enjoy a convenient and hassle-free experience while obtaining the necessary equipment for their skiing or snowboarding activities.

- \*\*Attributes\*\*: strName, intID, intSkisRented, intSnowboardsRented, intTotalItemsRented, intRentalHours, intRentalDays, intRentalWeeks, dblSkisHourly, dblSkisDaily, dblSkisWeekly, dblSnowboardsHourly, dblSnowboardsDaily, dblSnowboardsWeekly, dblRentalPrice, strCouponCode, dblDiscount

Here's a step-by-step guide on how to use the CustomerClass and its properties:

1. Import the CustomerClass: If the CustomerClass is defined in a separate file, import it into your current Python script or environment using from filename import CustomerClass.

2. Instantiate a Customer object: Create a new instance of the CustomerClass by providing values for the required parameters (strName, intID, intRentalTime, strRentalBasis). Optionally, you can also provide values for other parameters such as intSkisRented, intSnowboardsRented, strCouponCode, and dtmRentalStart.

3. Accessing and modifying object properties:

# To access the properties of the customer object, you can use dot notation (customer1.strName, customer1.intID, etc.).

# You can modify the properties using the setter methods. For example, to change the name of the customer:

4. Handling exceptions:

If you try to set a property with an invalid value, such as a negative ID or a non-integer value for intSkisRented, an exception will be raised. Make sure to handle these exceptions appropriately in your code.

5. Printing the customer object:

You can print the customer object directly or convert it to a string using the \_\_str\_\_ or \_\_repr\_\_ methods.

6. Coupon code validation:

The strCouponCode property ensures that the coupon code contains only letters. If you try to set it with a value containing digits or special characters, it will raise an exception.

### Shop Class

- \*\*Purpose\*\*: The purpose for the ski rental shop is to efficiently manage its inventory and facilitate rental transactions with customers. The shop maintains an up-to-date display of available skis and snowboards, ensuring transparency and accessibility for customers. It calculates rental estimates and exact bills based on the duration of the rental, providing the best price options according to the rental patterns and any applicable promotions. The shop also tracks daily rentals and revenue collected, allowing for informed decision-making and business optimization. Overall, the purpose of the ski rental shop is to deliver high-quality rental services while maximizing revenue and customer satisfaction.

- \*\*Attributes\*\*: intInitialSkisInventory, intInitialSnowboardsInventory, intSkiAvailable, intSnowboardAvailable, dblEstimateRentalPrice, dblActualRentalPrice, intDailySkis, intDailySnowboards, dblDailyRevenue

Here's a step-by-step guide on how to utilize the methods in the ShopClass:

1. Instantiate the ShopClass object: Create an instance of the ShopClass by calling its constructor.

2. Setting Initial Inventory: Set the initial inventory of skis and snowboards using the setter methods.

3. Starting an Estimate Request:

# Call the start\_Request() method to begin estimating a rental based on a customer's request.

# Provide the required parameters: intTimeRequested, strRentalBasis, intSkisRequested, intSnowboardsRequested, strCouponCode.

4. Returning Rental:

# Call the return\_Rental() method to start the return process based on a customer's existing rental.

# Provide the customer's ID as a parameter.

5. Accessing Shop Information:

Access information about the shop, such as initial inventory, using getter methods or by directly accessing properties.

6. Handling Exceptions:

Ensure that you handle exceptions raised by the methods appropriately in your code. For example, if you try to set initial inventory with a negative value, an exception will be raised.

7. Additional Notes:

Ensure that all required parameters are provided when calling the methods, and handle any validation or calculation errors as needed.

## Conclusion

In conclusion, the ski rental shop application is designed to streamline the process of renting skis and snowboards, ensuring a seamless experience for both customers and the shop’s staff. Customers can easily view available equipment, provide their details, and choose a rental plan that suits their needs, taking advantage of hourly, daily, or weekly rates, as well as family discounts and promotional offers. The shop can manage its inventory, issue accurate estimates and bills, and keep track of daily rentals and revenue. By following the step-by-step guide provided, application programmers can implement the system efficiently, ensuring that all functionalities work together harmoniously to provide a reliable and user-friendly service. This system not only simplifies the rental process but also helps the shop maintain a clear record of transactions, ultimately leading to better customer satisfaction and business management.