Question 1: Extending a Shape Area Calculator

Add support for a `Triangle` class without modifying the existing `Shape`, `Rectangle`, or `Circle` classes.

```
from math import pi

class Shape:
    def area(self):
        pass

class Rectangle(Shape):
    def __init__(self, width, height):
        self.width = width
        self.height = height

    def area(self):
        return self.width * self.height

class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return pi * self.radius ** 2

# Adding a new Triangle class without modifying existing code
class Triangle(Shape):
    def __init__(self, base, height):
        self.base = base
        self.height = height

    def area(self):
        return 0.5 * self.base * self.height

# Testing
shapes = [Rectangle(5, 10), Circle(7), Triangle(6, 8)]
for shape in shapes:
    print(f"{shape.__class__.__name__} Area: {shape.area()}")
```

```
Rectangle Area: 50
Circle Area: 153.93804002589985
Triangle Area: 24.0
```

Question 2: Extending a Discount Calculator

Add support for a `GoldCustomer` class without modifying the existing `DiscountCalculator` or `Customer` classes.

```
class DiscountCalculator:
    def calculate_discount(self, customer):
        return customer.get_discount()

class Customer:
    def __init__(self, total):
        self.total = total

    def get_discount(self):
        return 0  # Default discount for unknown customers

class RegularCustomer(Customer):
    def get_discount(self):
        return self.total * 0.1

class PremiumCustomer(Customer):
    def get_discount(self):
        return self.total * 0.2

# Adding a new GoldCustomer class without modifying existing code
class GoldCustomer(Customer):
    def get_discount(self):
        return self.total * 0.3

# Testing
customers = [RegularCustomer(1000), PremiumCustomer(1000), GoldCustomer(1000)]
```

```
discount_calculator = DiscountCalculator()

for customer in customers:
    print(f"{customer.__class__.__name__} Discount: {discount_calculator.calculate_discount(customer)}")
```

```
RegularCustomer Discount: 100.0
PremiumCustomer Discount: 200.0
GoldCustomer Discount: 300.0
```

Question 3: Extending a Notification System

Add support for a `PushNotification` class without modifying the existing `Notification`, `EmailNotification`, or `SMSNotification` classes.

```
class Notification:
    def send(self):
        pass

class EmailNotification(Notification):
    def __init__(self, recipient, message):
        self.recipient = recipient
        self.message = message

    def send(self):
        print(f"Email sent to {self.recipient}: {self.message}")

class SMSNotification(Notification):
    def __init__(self, phone_number, message):
        self.phone_number = phone_number
        self.message = message

    def send(self):
        print(f"SMS sent to {self.phone_number}: {self.message}")

# Adding a new PushNotification class without modifying existing code
class PushNotification(Notification):
    def __init__(self, user_id, message):
        self.user_id = user_id
        self.message = message

    def send(self):
        print(f"Push Notification sent to User {self.user_id}: {self.message}")

# Testing
notifications = [
    EmailNotification("google.com", "Hello Email!"),
    SMSNotification("1234567890", "Hello SMS!"),
    PushNotification("user123", "Hello Push Notification!")
]

for notification in notifications:
    notification.send()
```

```
Email sent to google.com: Hello Email!
SMS sent to 1234567890: Hello SMS!
Push Notification sent to User user123: Hello Push Notification!
```