Public safety notification system

Python Code:

```
class NotificationRecord:
    def __init__(self, alert_id, message):
        self.alert_id = alert_id
        self.message = message

    def update_message(self, new_message):
        self.message = new_message

    def __str__(self):
        return f"Alert ID: {self.alert_id}, Message: {self.message}"



class SubscriberList:
    def __init__(self):
        self.subscribers = []

    def add_subscriber(self, email):
        if email not in self.subscribers:
            self.subscribers.append(email)

    def remove_subscriber(self, email):
        if email in self.subscribers:
            self.subscribers.remove(email)

    def get_subscribers(self):
        return self.subscribers



class PublicSafetyNotificationSystem:
    def __init__(self):
        self.notifications = {}
```

```
        self.subscriber_lists = {}

    def create_notification(self, alert_id, message):
        self.notifications[alert_id] = NotificationRecord(alert_id, message)

    def read_notification(self, alert_id):
        return self.notifications.get(alert_id)

    def update_notification(self, alert_id, new_message):
        if alert_id in self.notifications:
            self.notifications[alert_id].update_message(new_message)

    def delete_notification(self, alert_id):
        if alert_id in self.notifications:
            del self.notifications[alert_id]

    def send_public_alerts(self, alert_id):
        if alert_id not in self.notifications:
            return []
        message = self.notifications[alert_id].message
        alerts = []
        for sub_list in self.subscriber_lists.values():
            for subscriber in sub_list.get_subscribers():
                alerts.append(f"Sent to {subscriber}: {message}")
        return alerts

    def manage_subscriber_list(self, list_id):
        if list_id not in self.subscriber_lists:
            self.subscriber_lists[list_id] = SubscriberList()
        return self.subscriber_lists[list_id]


# Unit Testing
```

```
import unittest

class TestPublicSafetyNotificationSystem(unittest.TestCase):
    def setUp(self):
        self.psns = PublicSafetyNotificationSystem()
        self.psns.create_notification(1, "High flood risk in your area.")
        self.list_id = 1
        self.sub_list = self.psns.manage_subscriber_list(self.list_id)
        self.sub_list.add_subscriber("email@example.com")

    def test_notification_creation_and_retrieval(self):
        self.assertEqual(str(self.psns.read_notification(1)), "Alert ID: 1, Message: High flood risk in your area.")

    def test_update_notification(self):
        self.psns.update_notification(1, "Updated flood alert!")
        self.assertEqual(str(self.psns.read_notification(1)), "Alert ID: 1, Message: Updated flood alert!")

    def test_delete_notification(self):
        self.psns.delete_notification(1)
        self.assertIsNone(self.psns.read_notification(1))

    def test_send_alerts_to_subscribers(self):
        alerts = self.psns.send_public_alerts(1)
        self.assertEqual(len(alerts), 1)
        self.assertIn("Sent to email@example.com: High flood risk in your area.", alerts)

    def test_manage_subscribers(self):
        emails = self.sub_list.get_subscribers()
        self.assertIn("email@example.com", emails)
        self.sub_list.remove_subscriber("email@example.com")
        self.assertNotIn("email@example.com", self.sub_list.get_subscribers())
```

```
if __name__ == "__main__":
    unittest.main()
```