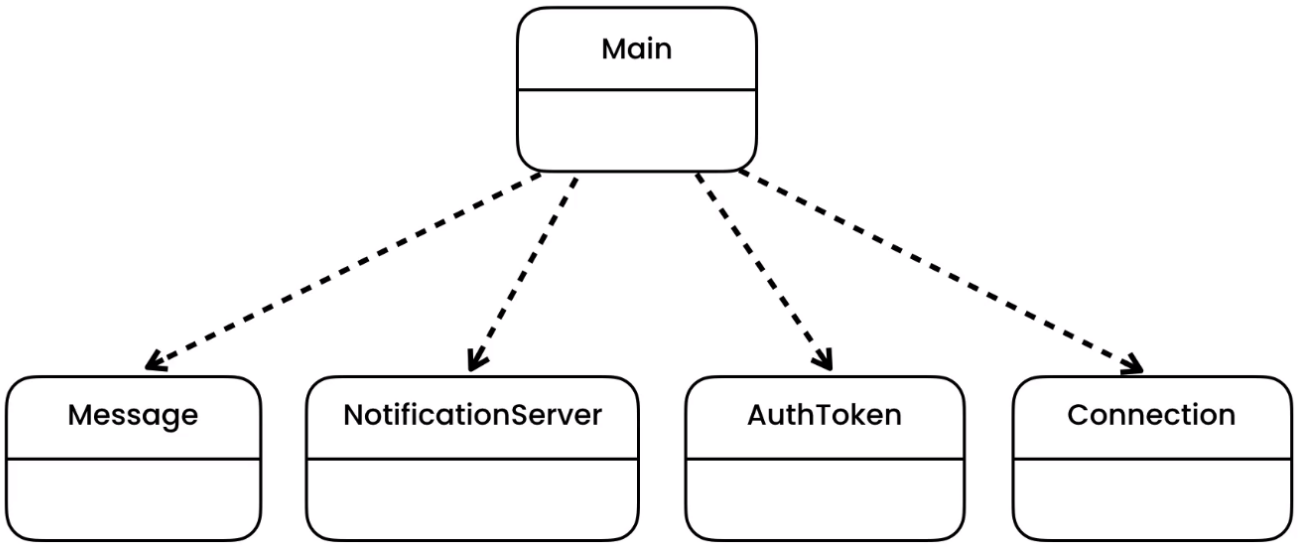
# Facade Pattern

Helps to proviode a simple interface to a complex system

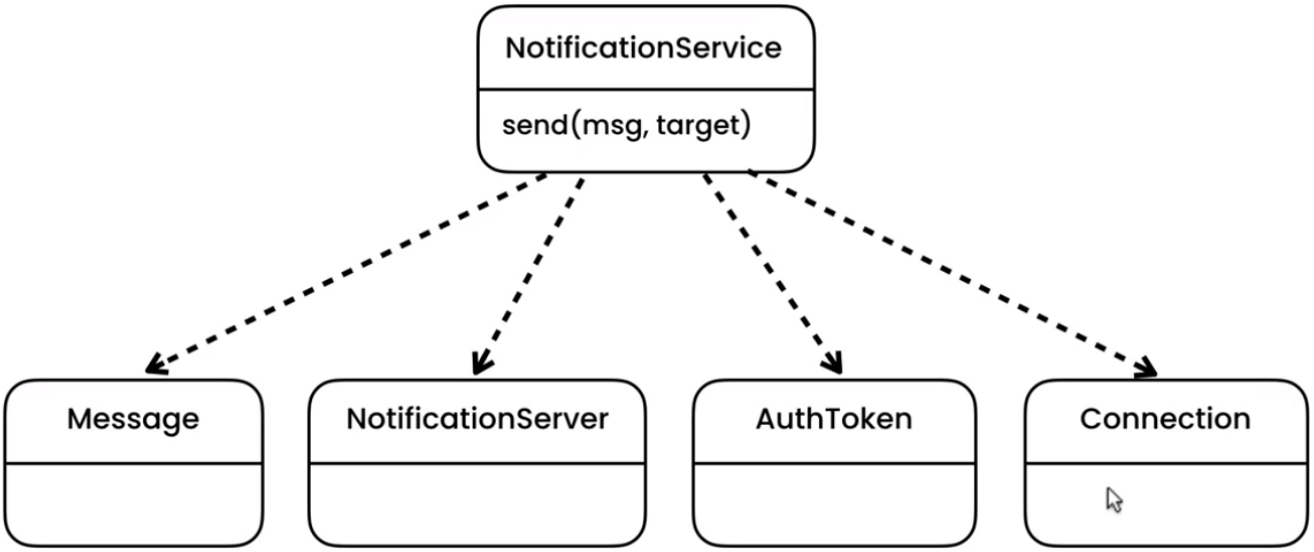
*Example Probem* - You are Building a mobile app with ability to send push notifications to Users. Below is an example of all of the steps taken to carry out this task, the facde pattern can be used to reduce the complexity form below and reduce the numbr of steps taken.

```
blic class Main {
  public static void main(String[] args) {
      var server = new NotificationServer();
      var connection = server.connect( ipAddress: "ip");
      var authToken = server.authenticate( appID: "appID",  key: "key
      var message = new Message( content: "Hello World");
      server.send(authToken, message,  target: "target");
      connection.disconnect();
  }
```

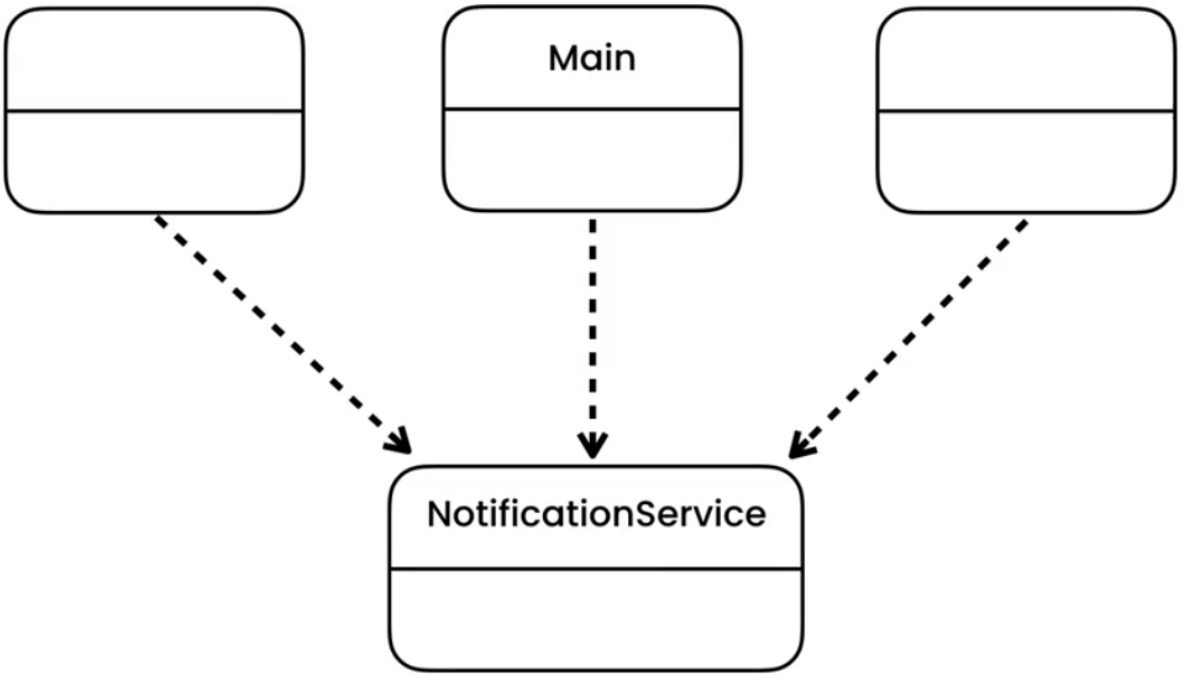
Current Structure :




main class is coupled with 4 other classes. If we have 10 other classes in our app from which we send a push notification, this will add more coupling. thefacade pattern aims to reduce the complexity and coupling highlighted above.We implement a notification service that implements the above steps and then use it in all of the classes which send a push notification.




Reducing complexity and coupling by using a facade which forwards the requests to all of the necessary classes :




| | | | |
|---|---|---|---|
| | | | |

```
public class NotificationServer {
  // connect() -> Connection
 // authenticate(appID, key) -> AuthToken
 // send(authToken, message, target)
 // conn.disconnect()

 public Connection connect(String
ipAddress) {
    return new Connection();
 }

  public AuthToken authenticate(String
appID, String key) {
    return new AuthToken();
 }

  public void send(AuthToken authToken, M
essage message, String target) {
    System.out.println("Sending a
message");
 }
}
```

```
public class Message {
  private String content;

 public Message(String
content) {
    this.content =
content;
 }
}
```

```
public class C
onnection {
  public void
disconnect() {

  }
}
```

```
public class
AuthToken {
}
```

```
public class NotificationService
{
  public void send(String
message, String target) {
    var server = new Notificatio
nServer();
 var connection = server.connect
("ip");
 var authToken = server.
authenticate("appID", "key");
 server.send(authToken, new Mess
age(message), target);
 connection.disconnect();
 }
}
```

```
public class Main{


public statice void send
(String[] args) {

var service = new notificationService();

service.send("Hello World," "target");

}

}
```