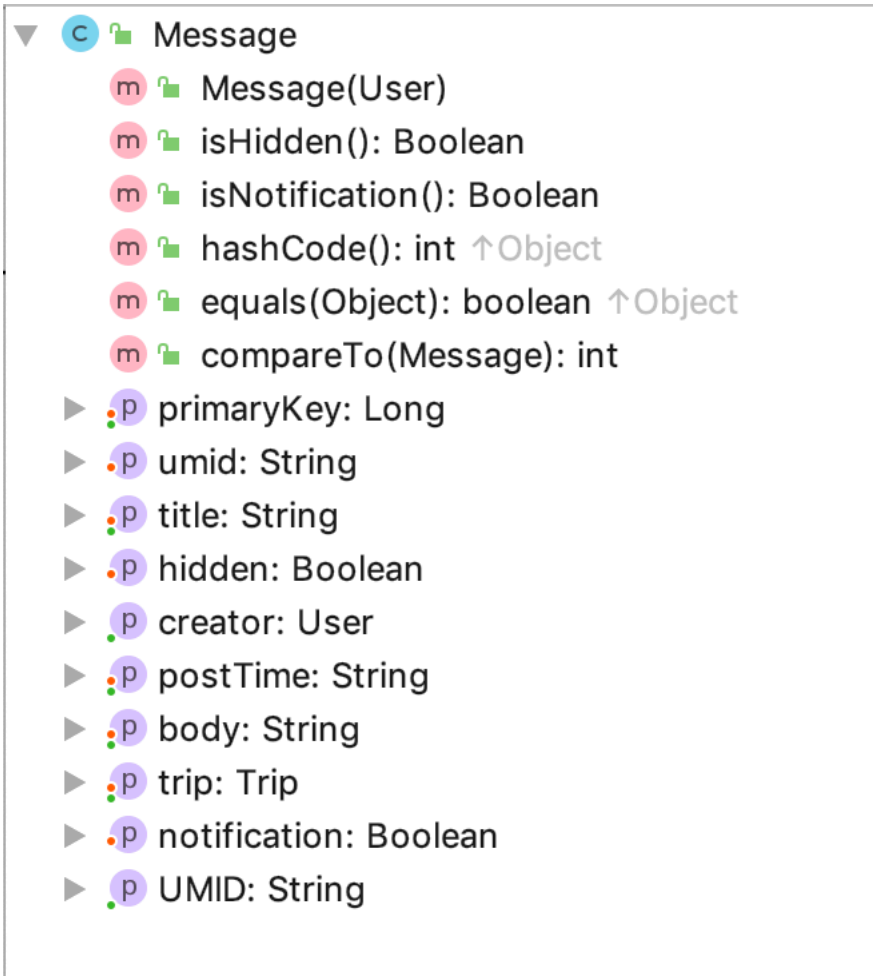
Composite



We use composite to construct object with a few other objects inside.

Template

We use template to define gateway object and then extends with different implementation to execute the operation to database.

```
public abstract class Gateway<T> {
    protected Connection connection;

    public Gateway() {
    }

    public abstract Gateway<T> save(Object object);

    public abstract Gateway<T> delete(Object object);

    protected abstract Map<Long, T> load();
```

```
    public abstract Map<Long, T> getLoaded();

    public abstract T find(Long id);
}
```

```
public class MessageGateway extends Gateway<Message> {
    private static Map<Long, Message> messages;

    public MessageGateway(Connection con) { connection = con; }

    @Override
    public Gateway<Message> save(Object object) {...}

    @Override
    public Gateway<Message> delete(Object object) {...}

    @Override
    protected Map<Long, Message> load() {...}

    @Override
    public Map<Long, Message> getLoaded() {...}

    @Override
    public Message find(Long id) {...}
}
```

Singleton:
We use one singleton to access the same connector to the database. And make sure only one database is used.

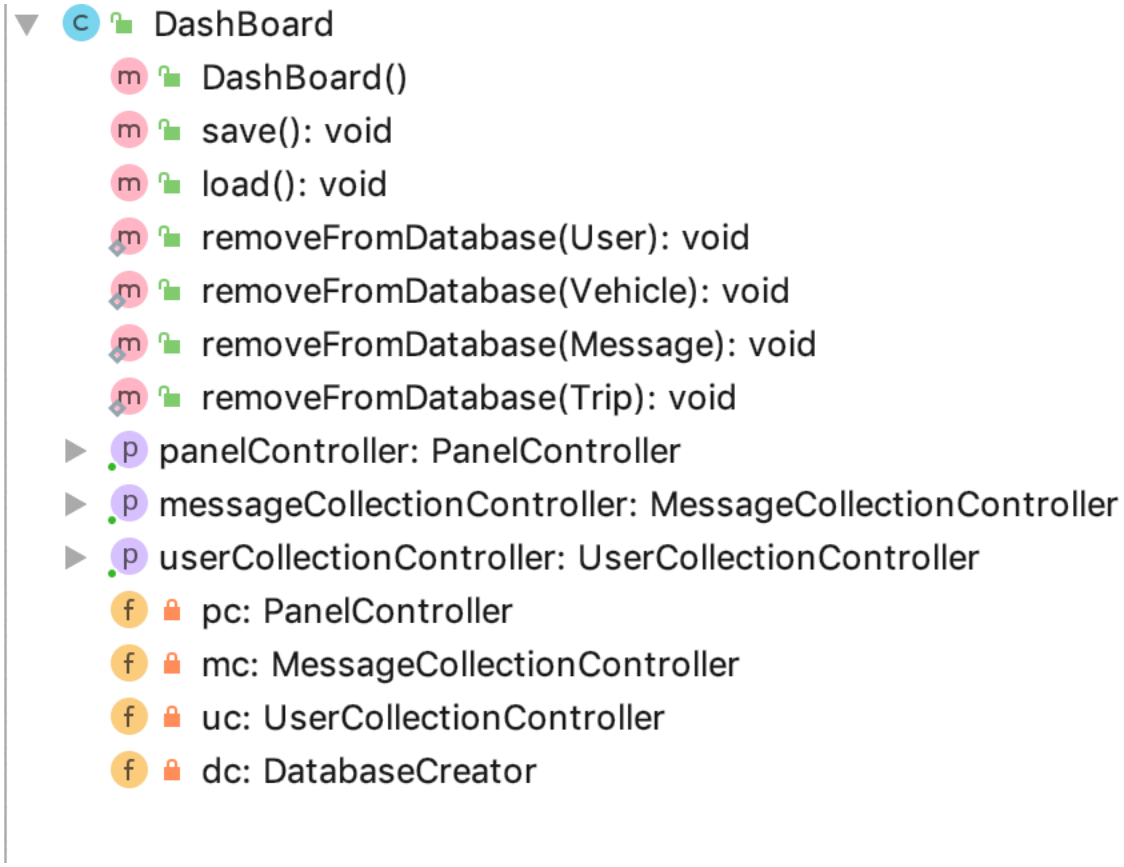
```
public class DatabaseCreator {
    private static DatabaseCreator singleton = null;
    private Connection connection = null;

    private DatabaseCreator() {...}

    public static DatabaseCreator getInstance() {...}

    public void freeInstance() {...}

    public Connection getConnection() { return connection; }
}
```



Facade:
Our controller system is complex, so we implemented a facade(Dash Board)

DashBoard cover the complexity of the subsystem and is the entry point of the system.

Iterator:
Iterator is builtin in java and is used to traverse the collection object.

```
@Test
void pushReminders() {
    m.setTrip(new Trip(user, new Date(1000), 1));
    messageCollection.insert(m);
    messageCollection.pushReminders(new Date(1000));
    Iterator<Message> itr = user.getNotifications().iterator();
    Message me = itr.next();
    assertTrue(me.isNotification());
}
```

Observer: Almost every listener in Swing, builtin in java

```
backButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        backButtonActionPerformed(evt);
    }
});
```

Builder: the add function of Frame use builder pattern, instead of passing all info about other Panel or component, it serves as a builder to simplify the process.

```
public void changeFrame(JPanel newFrame) {
    this.stack.push(newFrame);

    frame.getContentPane().removeAll();
    frame.add(newFrame);
    frame.revalidate();
    frame.pack();
    frame.repaint();
}
```