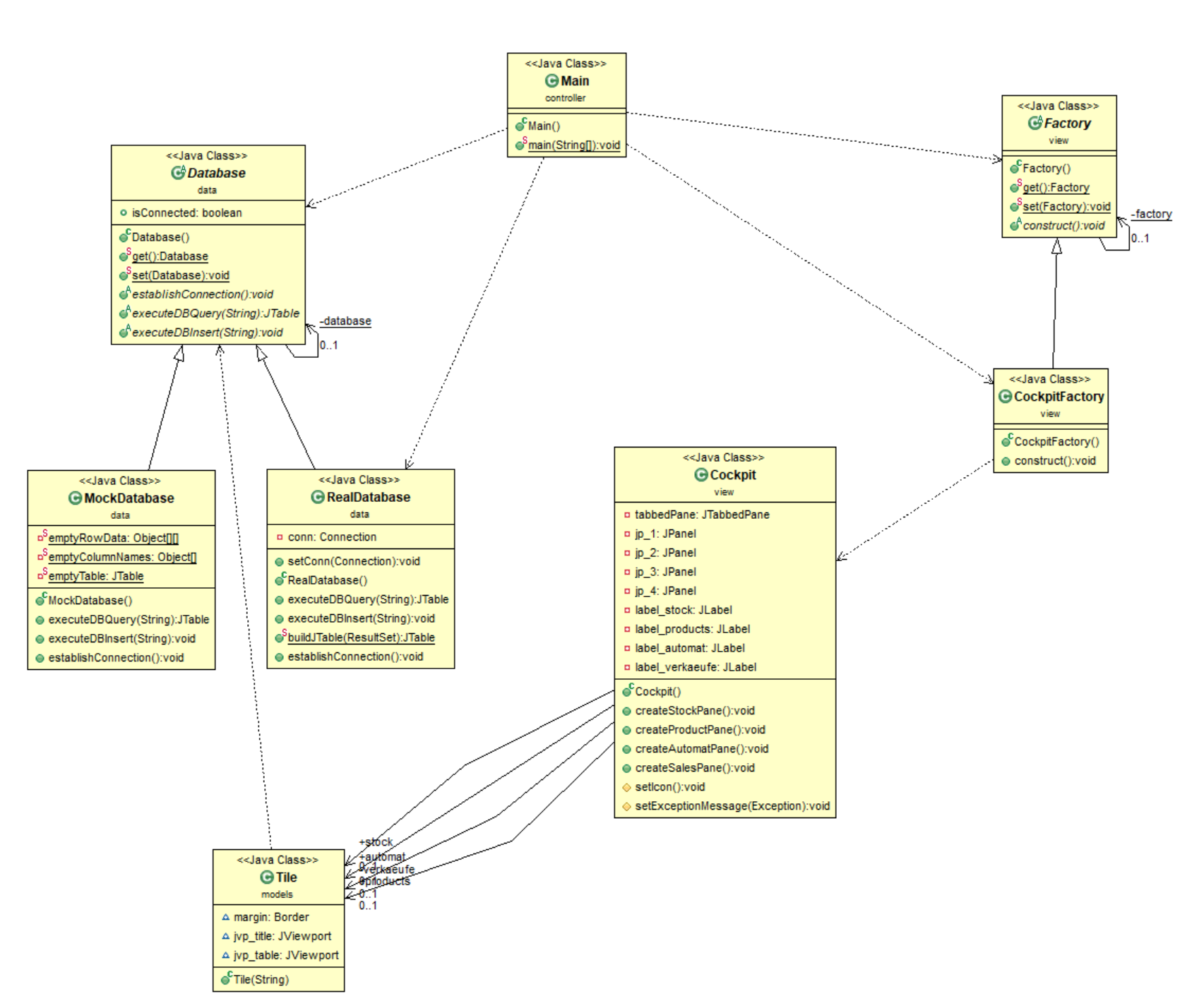
# Erzeugung des Cockpits mit Tiles



Das ist gut:

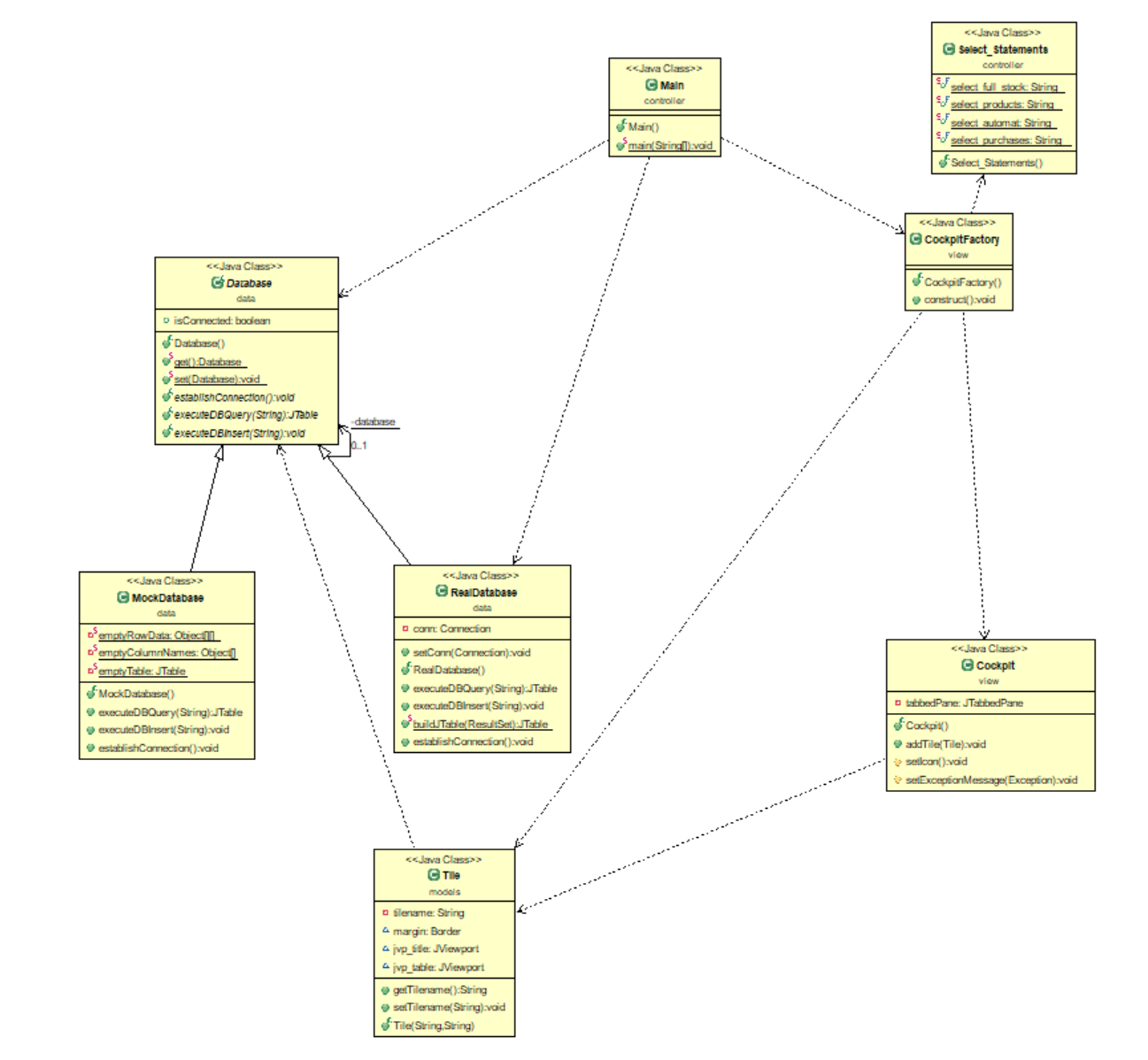
* Automat wird von einer Factory erstellt -> die kann für ihn alles steuern

Probleme:

* Anzahl der Tiles ist fest
* feste Knüpfung an die Datenbank
* Tiles sollten die DB nicht kennen müssen bzw nicht mit ihr interagieren!

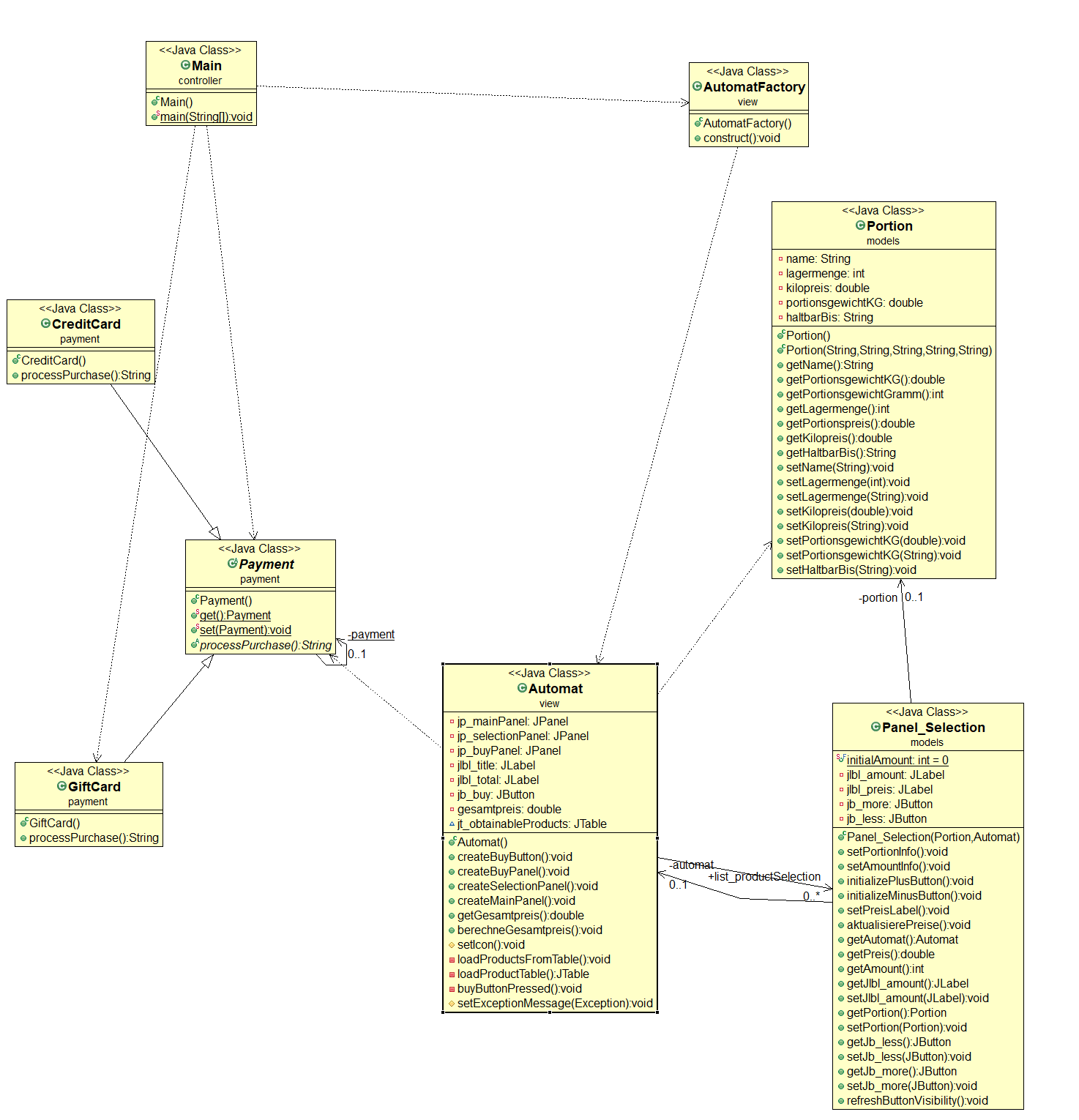
Lösung:

* Wir führen eine Methode addTile() im Cockpit ein
* die Factory erzeugt die Tiles für das Cockpit

Jetzt muss das Cockpit nicht mehr selbst die Tiles erzeugen, das macht die Cockpit-Factory.

Die Anzahl der Tiles und der Inhalt kann in der Factory bestimmt werden.

# Erzeugung des Automats



Das ist gut:

* Automat hat die Panels

Das ist nicht gut:

* Die Panels kennen den Automaten! Ganz mies

Was will ich umsetzen?

1. Die Panel-Selection sollte nicht den Automaten kennen müssen.
2. ~~der Automat soll die Tabellen gegeben bekommen (Factory)~~
3. Benachrichtigen des Automaten mithilfe eines Observers?

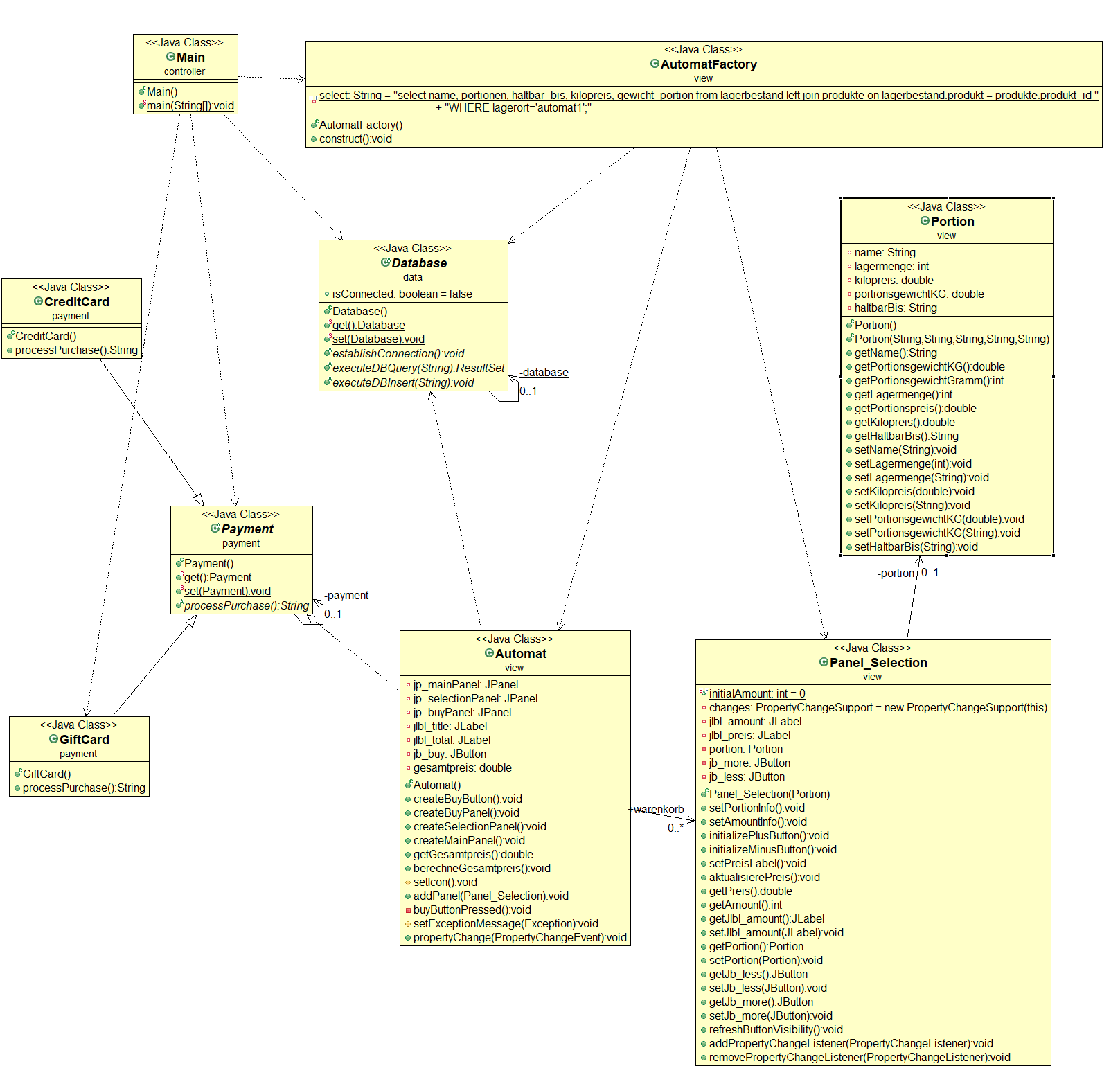
Gestaltung der Benachrichtigung:

Das Panel\_Selection muss seinen passenden Automaten benachrichtigen, falls die ausgewählte Menge verändert wurde.

Was jetzt geändert wurde:

Automat ist der Property Listener, Panel fired das Event falls die Ausgewählte Menge verändert wird.

Die Automat Factory erzeugt die Selection Panels, nicht der Automat selber.



Tiles lesen im Moment noch selber aus der DB.

# Payment-Klassen und Exceptions

Im Paket payment sind die abstrakte Klasse Payment sowie die zwei Klassen CreditCardPayment und GiftCardPayment zu finden, die von Payment erben.

Welche der beiden Zahlungsmethoden für den Einkauf beim Automaten verwendet werden, wird zentral in der Main festgelegt.   
Die processPurchase(double gesamtpreis)-Methode der Payment Klasse wirft eine PaymentButcherException, die processPurchase-Methoden der beiden erbenden Klassen CreditCardPayment und GiftCardPayment werfen aber jeweilige Sonderformen dieser Exception, die CreditCardException sowie GiftCardException, die beide von PaymentButcherException erben.

CreditCardException und GiftCardException wiederum sollen auftreten, wenn im eigentlichen Zahlungsablauf etwas nicht funktioniert, d.h. beispielsweise ein falscher Kreditkarten-Pincode eingegeben wurde oder das Guthaben der Geschenkgutscheinkarte nicht mehr ausreicht.   
Unser Programmablauf besitzt keine derartigen Funktionalitäten (Kartenlesegerät), bei denen diese Exceptions tatsächlich auftreten könnten – für den reellen Anwendungsfall haben wir die entsprechenden Klassen dennoch schon einmal implementiert.