

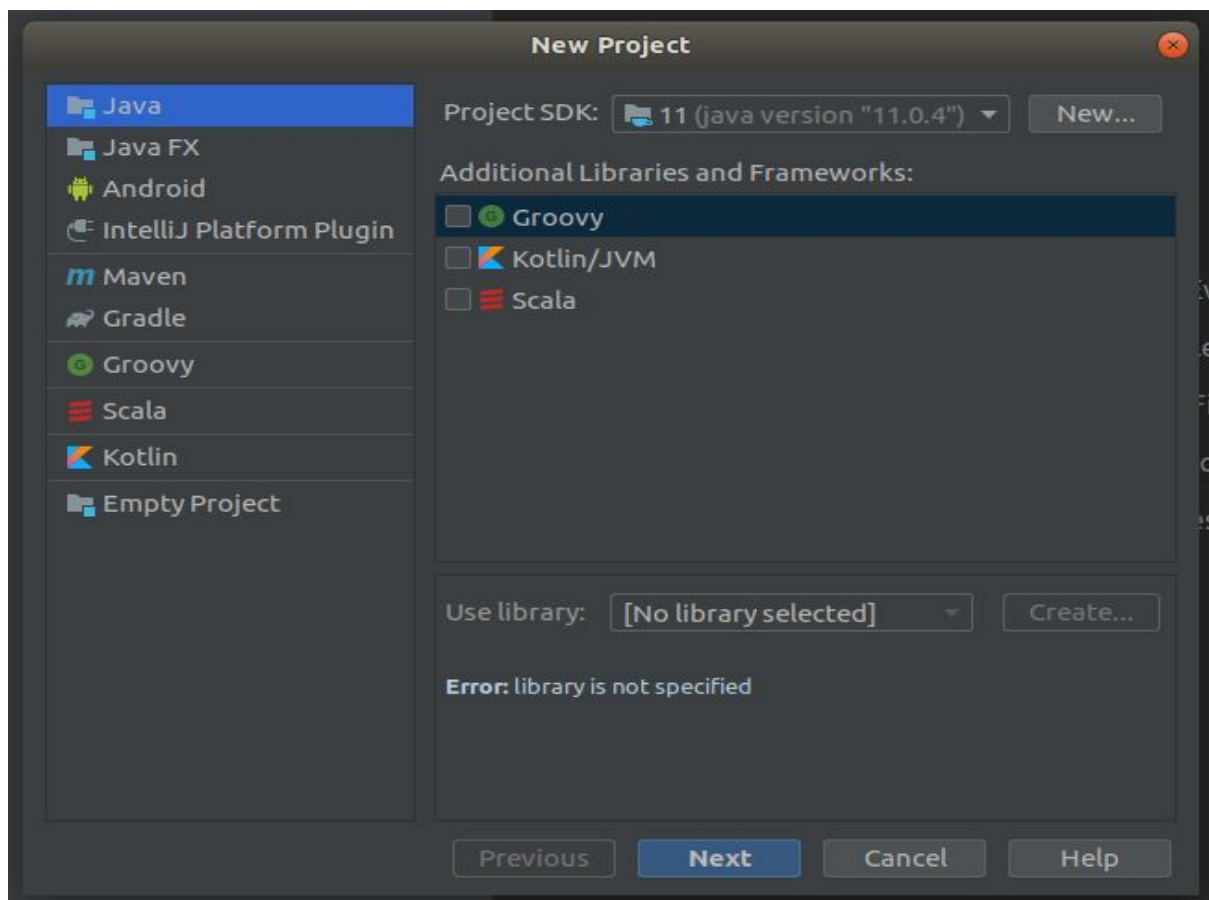
# Java praktijk - supermarkt

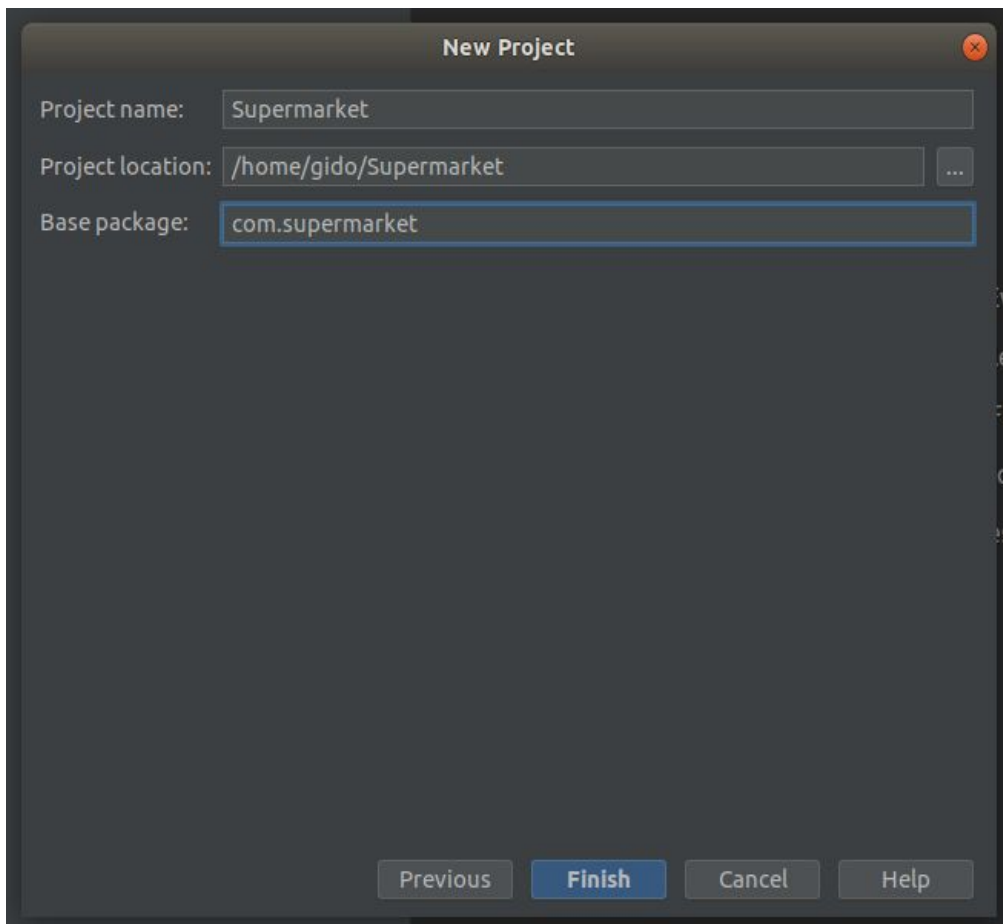
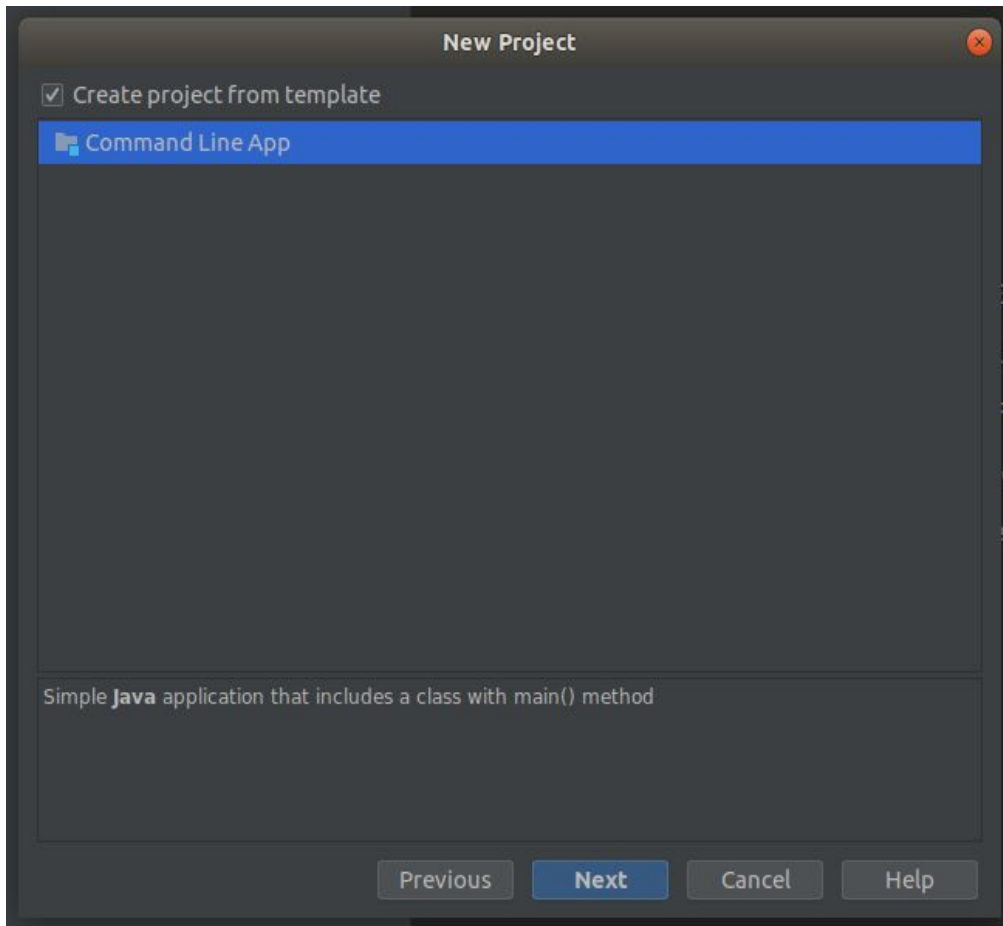
## deel 1

We gaan een supermarkt situatie namaken. We hebben daarbij om te beginnen een Java project in een IDE (Integrated Development Environment) als IntelliJ nodig.


## Opdracht 1

Maak in IntelliJ een nieuw Java project aan door de onderstaande stappen te volgen.



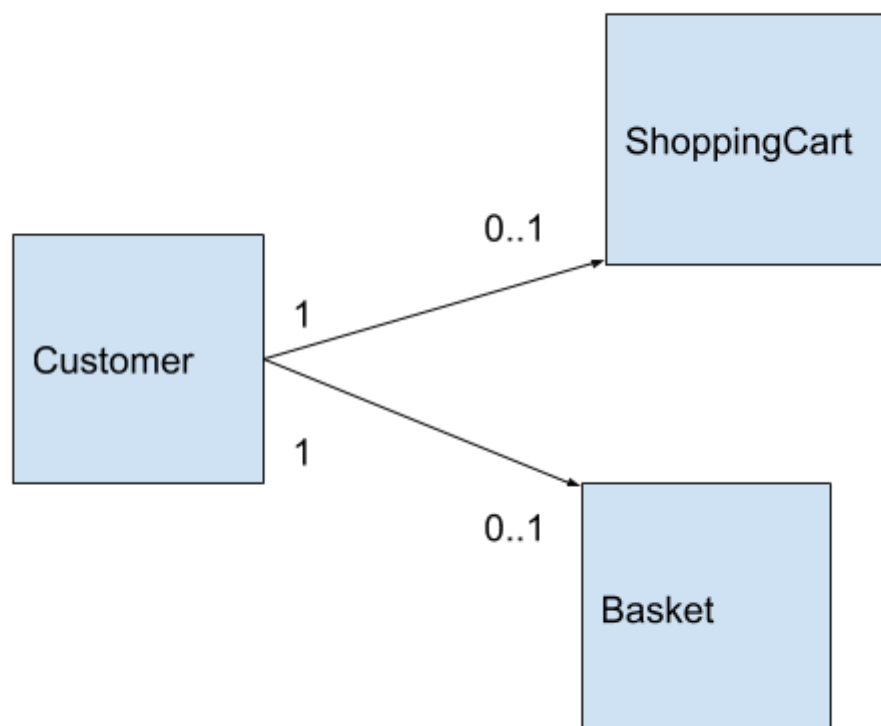


Je hebt nu een Java 11 project in IntelliJ genaamd 'Supermarket'. Het bevat het root package `com.supermarket`, met daarin als entry point voor een applicatie de class `Main`.

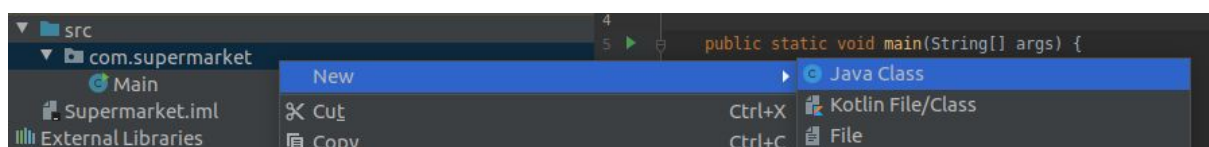
Je kunt je applicatie starten met de  knop. Op dit moment doet het feitelijk nog niets, maar tijdens en na het uitvoeren van elke volgende opdracht kun je op deze manier het effect van je wijzigingen zien. Zo kun je doorlopend testen of het werkt zoals verwacht en vroegtijdig fouten opsporen. Ga dan pas verder wanneer het goed werkt.

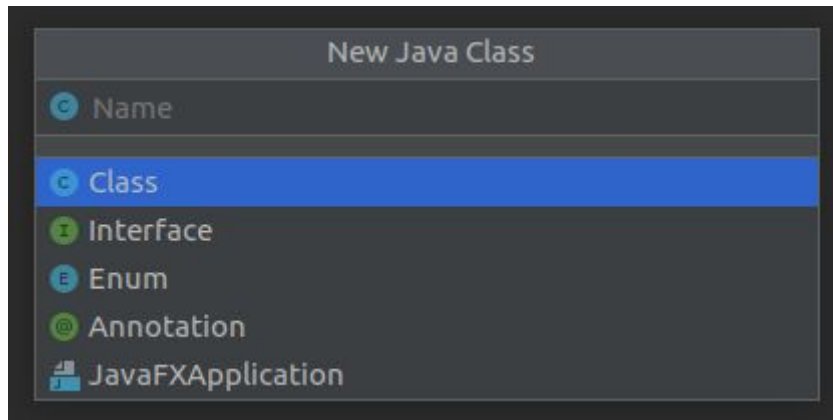
## Opdracht 2

Voor de supermarkt beperken we ons op dit moment tot het modelleren van klanten, winkelwagens en winkelmandjes. Zie hieronder het domein model.



Maak binnen het root package voor elke entiteit een class, door dit package te selecteren, hiervoor het context menu op te roepen en de afgebeelde stappen uit te voeren.





Geef elke class dezelfde naam als in het diagram (CamelCase). Je hebt hierna lege classes, met een default constructor.

Maak in de main methode een declaratie van een (lokale) variabele

- met `Customer` als type en
- een zelf te bepalen naam.

Initialiseer vervolgens deze variabele met een nieuw `Customer` object, waarvoor de (default) constructor moet worden aangeroepen.

Zorg er hierna voor dat de waarde van deze variabele wordt uitgeprint vanuit de main methode.

## Opdracht 3

De pijlen tussen de entiteiten in het domein model vertegenwoordigen 'has a' relaties, waarbij de richting van zo een pijl aangeeft wat in dat geval de parent is en wat de child: een `Customer` instantie (parent) heeft zowel een `ShoppingCart` instantie (child) als een `Basket` instantie (child). Ook wordt er bij de relatie aangegeven hoeveel van die objecten er aan beide kanten zijn. Zo is er één `Customer` en geldt voor beide children dat er geen of één object van beschikbaar is voor die parent.

Zo een relatie wordt geïmplementeerd door in de *parent* een *field* of *member variabele* (een variabele op class niveau) te declareren. Als zo een variabele *geen* waarde heeft, oftewel `null`, is er *geen* instantie aan die relatie gekoppeld; in het andere geval uiteraard wel. Beide scenario's worden door deze structuur ondersteund: door de tijd heen kan onze `Customer` op het ene moment bijvoorbeeld een `Basket` object hebben en later niet meer.


Breng de uitgewerkte relaties aan door variabelen te declareren in de nog lege `Customer` class. Laat de declaraties voorafgaan door het keyword 'public', wat ervoor zorgt dat zo een variabele ook van buiten dat object toegankelijk is. Initialiseer deze variabelen overigens nog niet.


Ken vervolgens, vanuit de main methode, nieuw geconstrueerde instanties van `ShoppingCart` en `Basket` aan deze variabelen van je lokale `Customer` toe.


Print hierna binnen de main methode voor beide variabelen de waarde, door de betreffende variabele te benaderen via je `Customer` object en deze referentie mee te geven als parameter aan de print methode.

## Opdracht 4

We geven `Customer` ook een naam. Declareer een *publiek toegankelijke* member variabele van het type `String` met de naam 'name'.

Elke `Customer` heeft een naam en bovendien vanaf het begin. Daarom gaan we deze variabele initialiseren via de *constructor*. Maak eerst een no-arguments constructor binnen `Customer`. Werkt je applicatie nu nog? Waarom 

Definieer voor deze constructor een `String` parameter genaamd 'name'. Werkt je applicatie nu nog steeds? Waarom 

Geef nu bij de aanroep van de `Customer` constructor een naam naar keuze mee. Werkt je applicatie nu? Waarom 

Initialiseer vervolgens de member variabele `name` met de ontvangen waarde van die constructor parameter. Die member variabele wordt echter overschaduwd door deze parameter, dus verzin een manier om dat te omzeilen.

Print tot slot de naam van je `Customer` in de main methode door deze daaraan op te vragen. Is dit de meegegeven naam 