Consider a supermarket, which sells products and produces invoices. We are going to model the supermarket using Maps and Sets.

1. Download **PL2_Generics_Initial** from moodle.

The Supermarket class only has one data structure:

```
public class Supermarket {
    Map <Invoice, Set<Product>> m;
```

Nevertheless, this data structure implies that we can perform the operations of **equality checking**, **comparison** and **hash code extraction** on the Invoice and Product classes. To do this we should override the ***equals*** and ***hashCode*** methods, as well as implement the ***Comparable*** interface.

2. Implement the methods in **Invoice** and **Product**. In the Invoice class, equality, comparison and hash code refers to its **reference** field. In product, all the methods refer to its **identification** field. This implementation has the InvoiceTest and ProductTest suites to pass.

Now that the Invoice and Product classes are finished we can start to develop the **Supermarket** class. Initially we must be able to create a supermarket, so the first methods to be developed are *getInvoices* and *numberOfProductsPerInvoice*.

The *getInvoices* method receives a list of strings with the following format:

- there are *n* list elements to each Invoice
- the first element always starts with an I, followed by the reference and the date.
- the next *n-1* elements are its products. Each line starts with a P, followed by its identification, quantity and price

This is an example containing 2 invoices. Each line is a string element in the list:

```
I,INV001,2016/09/10
P,EGG,12,200
P,APPLE,2,140
P,BUTTER,1,100
I,INV002,2016/09/11
P,PEAR,3,230
P,CHIPS,3,320
```

3. Implement the *getInvoices* method. The test for this method will appear next.

The *numberOfProductsPerInvoice* method returns the count of the products in each invoice. The result is a Map of key Invoice and integer data, in which each data element is the number of products of the corresponding Invoice.

4. Implement the *numberOfProductsPerInvoice* method. One supermarket test must pass.

Now we implement the *betweenDates(d1,d2)* method. Its objective is to return a Set in which there are only invoices which date is greater that d1 and smaller than d2.

5. Implement the *betweenDates* method. Two supermarket tests must pass.

6. Implement the *totalOfProduct(id)* method which sums all the price*quantity of product *id* in all the invoices. Three supermarket tests must pass.

Finally a conversion in needed. The original Map of invoices to products will be transformed in another from product ids to invoices. The resulting map will have a set of invoices for each product id. An invoice is a part of the set if, in the original map, the product is mapped to the invoice.

7. Implement the *convertInvoices* method. All test must now pass.