# Form Factory Method**

Subclasses implement a method similarly,
except for the way they create a type of object

*On each subclass, extract the instantiation code to a Factory Method, then declare an abstract version of the Factory Method on the superclass*

```
abstract class Query…
   public abstract void doQuery() throws QueryException;

class QuerySD51 extends Query …
   public void doQuery() throws QueryException {
      if (sdQuery != null) sdQuery.clearResultSet();
      sdQuery = sdSession.createQuery(SDQuery.OPEN_FOR_QUERY);
      executeQuery(sdQuery);
}

class QuerySD52  extends Query …
   public void doQuery() throws QueryException {
      if (sdQuery != null) sdQuery.clearResultSet();
      sdQuery = sdLoginSession.createQuery(SDQuery.OPEN_FOR_QUERY);
      executeQuery(sdQuery);
}
```

```
abstract class Query …
   protected abstract SDQuery createQuery();
   public abstract void doQuery() throws QueryException;

class QuerySD51 extends Query …
   protected SDQuery createQuery() {
      return sdSession.createQuery(SDQuery.OPEN_FOR_QUERY);
   }
   public void doQuery() throws QueryException {
      if (sdQuery != null) sdQuery.clearResultSet();
      sdQuery = createQuery();
      executeQuery(sdQuery);
   }

class QuerySD52  extends Query …
   protected SDQuery createQuery() {
      return sdLoginSession.createQuery(SDQuery.OPEN_FOR_QUERY);
   }
   public void doQuery() throws QueryException {
      if (sdQuery != null) sdQuery.clearResultSet();
      sdQuery = createQuery();
      executeQuery(sdQuery);
   }
```

### Motivation

This refactoring prepares you to *Form Template Method (345)* [Fowler].
[More Coming Soon]