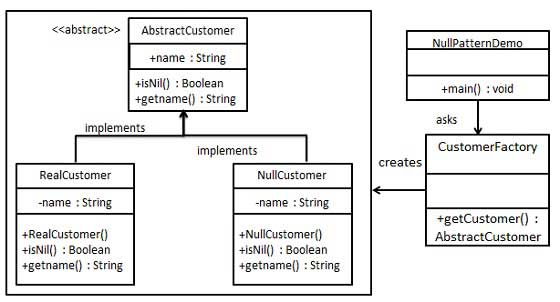
In Null Object pattern, a null object replaces check of NULL object instance. Instead of putting if check for a null value, Null Object reflects a do nothing relationship. Such Null object can also be used to provide default behaviour in case data is not available.

In Null Object pattern, we create an abstract class specifying various operations to be done, concrete classes extending this class and a null object class providing do nothing implemention of this class and will be used seemlessly where we need to check null value.

## Implementation

We are going to create a *AbstractCustomer* abstract class defining opearations. Here the name of the customer and concrete classes extending the *AbstractCustomer* class. A factory class *CustomerFactory* is created to return either *RealCustomer* or *NullCustomer*objects based on the name of customer passed to it.

*NullPatternDemo*, our demo class, will use *CustomerFactory* to demonstrate the use of Null Object pattern.



/\*

\* Step 1

\* create an abstract class

\* AbstractCustomer.java

\*/

abstract class AbstractCustomer{

protected String name;

public abstract boolean isNil();

public abstract String getName();

}

/\*

\* step 2

\* create concrete classes extending the above class

\* RealCustomer.java - Non null customer object

\*/

class RealCustomer extends AbstractCustomer{

public RealCustomer(String name) {

this.name = name;

}

public String getName() {

return name;

}

public boolean isNil() {

return false;

}

}

/\*

\* NullCustomer.java

\*/

class NullCustomer extends AbstractCustomer{

@Override

public String getName() {

return "Not availabe in customer databse";

}

@Override

public boolean isNil() {

return true;

}

}

/\*

\* step 3

\* create customer factory class

\* CustomerFactory.java

\*/

class CustomerFactory {

public static final String[] names = {"Rob", "Joe", "Julie"};

public static AbstractCustomer getCustomer(String customerName) {

for(String name : names) {

if(name.equals(customerName)) {

return new RealCustomer(name);

}

}

return new NullCustomer();

}

}

/\*

\* Step 4

\* Use the CustomerFactory to get either RealCustomer or NullCustomer objects on the name of customer passed to it.

\*/

public class NullObjectPatternDemo {

public static void main(String[] args) {

AbstractCustomer customer1 = CustomerFactory.getCustomer("Rob");

AbstractCustomer customer2 = CustomerFactory.getCustomer("Bob");

AbstractCustomer customer3 = CustomerFactory.getCustomer("Julie");

AbstractCustomer customer4 = CustomerFactory.getCustomer("Laura");

System.out.println("customers");

System.out.println(customer1.getName());

System.out.println(customer2.getName());

System.out.println(customer3.getName());

System.out.println(customer4.getName());

}

}