

Exercise 5	Constructors Chaining and Method Overloading
<p>Step 1 - Create the classes we will use in this project</p> <p>At the completion of this step you have:</p> <ol style="list-style-type: none"> 1) defined an Order class with a default constructor that does not do anything 2) defined a BatchOrder class that inherits the members of Order and has a default constructor that does not do anything 3) defined an OverSeasOrder class that inherits the members of BatchOrder which inherits the members of Order and has a default constructor that does not do anything 4) defined a TestOrder class with a public static void main method that can be executed as a program. Inside the main method you instantiated an object of type Order, type BatchOrder, and type OverSeasOrder <p>Why might we have a BatchOrder that inherits the characteristics of Order? A Batch is when someone takes a group of items and processes them all at once, for instance, maybe their machines are very busy during the day and they group items in a batch and run them at night when the machines are idle. Our company may have a rule that says anyone that purchases an item and does not pick express shipping has their order processed in batch. The Order class itself does not contain the logic to group the orders to satisfy these rules but we still want to use aspects of the Order class and modify them in BatchOrder</p> <p>Why an OverSeasOrder class extending BatchOrder? Our company has a rule most times called a business rule that all overseas orders are processed in batch. We extend BatchOrder because the logic for an over seas order is different than a basic batched order.</p>	<p>Create 4 classes which means 4 individual source files like the following:</p> <pre> public class Order { public Order() { } } public class BatchOrder extends Order{ public BatchOrder() { } } public class OverSeasOrder extends BatchOrder{ public OverSeasOrder() { } } public class TestOrder { public TestOrder() { } public static void main(String[] args) { Order order1 = new Order(); BatchOrder batch1 = new BatchOrder(); OverSeasOrder seal = new OverSeasOrder(); } } </pre>

Step 2 – add a println inside the various default constructors we created in Step 1

We add these so we can see how many times and in what order the constructors are invoked.

Remember all classes inherit the members of the super class directly above them in the chain of classes. Everything starts from the Object() class. The super() reference is invoked every time we new-up or instantiate an object. In the default constructor of the new class there is an implicit super() reference.

So, when we write the code new Order() to the right it runs the Order() default constructor which implicitly runs the Object() constructor with a super() reference that does not have to actually be coded, hence implicit. The super is run first, than any code in the constructor.

Moreover, the new BatchOrder() runs the BatchOrder default constructor which runs the Order() default constructor which runs the Object() default constructor.

Even more, the new OverSeasOrder() runs the BatchOrder() default constructor runs the Order() default constructor runs the Object() default constructor.

```
public class Order {
    public Order() {
        System.out.println("I am in the Order
        Constructor");
    }
}

public class BatchOrder extends Order{
    public BatchOrder() {
        System.out.println("I am in the BatchOrder
        Constructor");
    }
}

public class OverSeasOrder extends
BatchOrder{
    public OverSeasOrder() {
        System.out.println("I am in the
        OverSeasOrder Constructor");
    }
}

public class TestOrder {
    public TestOrder() {
        System.out.println("I am in the TestOrder
        Constructor");
    }
}

    }
    public static void main(String[] args) {
        Order order1 = new Order();
        BatchOrder batch1 = new BatchOrder();
        OverSeasOrder seal = new OverSeasOrder();
    }
}
```

Step 3 – Save all, fix any typos you may have made, run TestOrder

This line of code:
Order order1 = new Order();

Causes this line of output:
I am in the Order Constructor

This line of code:
BatchOrder batch1 = new
BatchOrder();

Causes these lines of output:
I am in the Order Constructor
I am in the BatchOrder Constructor

This line of code:
OverSeasOrder seal = new
OverSeasOrder();

Causes these lines of output:
I am in the Order Constructor
I am in the BatchOrder Constructor
I am in the OverSeasOrder
Constructor

The code that runs inside the main method is this:
Order order1 = new Order();
BatchOrder batch1 = new BatchOrder();
OverSeasOrder seal = new OverSeasOrder();

The output from the System.out.println's is as follows:

```
I am in the Order Constructor  
I am in the Order Constructor  
I am in the BatchOrder Constructor  
I am in the Order Constructor  
I am in the BatchOrder Constructor  
I am in the OverSeasOrder Constructor
```

Step 4 – Create an order identifier in the Order class by creating a createOrder() method and a second constructor that takes a String parameter

We made a method to create a random ID. I thought about using the Random class, decided to use the current time because I have seen this done many times in Java code. Looked up Time in google, Time under the Java 5 API tells us it is deprecated in favor of Calendar. Looked up Calendar in google for Java 5 and found the getTimeInMillis() method.

Converted the return value long to a string and concatenated it onto the buyers lastname.

A random-ish order ID will be returned

```
import java.util.GregorianCalendar;  
public class Order {  
    private String id = null;  
    private GregorianCalendar currentTime = new  
    GregorianCalendar();  
  
    public Order() {  
        System.out.println("I am in the Order  
        Constructor");  
    }  
  
    public Order(String lastName) {  
        System.out.println("I am in the Order  
        Constructor that takes a string as a  
        parameter");  
  
        id = createOrderID(lastName);  
        System.out.println("Your Order ID is: " +  
        id);  
    }  
  
    public String createOrderID(String  
    lastName) {  
        return lastName +  
        currentTime.getTimeInMillis();  
    }  
}
```

Step 5 – update the TestOrder class to use the new constructor in the Order Class

Added a Scanner to take input from the user. Added two Strings to take in two unique last names for processing IDs.

Created an object order2 of type Order using our new constructor that takes in a String.

Created an object batch2 of type BatchOrder using its default constructor which runs the Order default constructor. Can not use the name we entered with BatchOrder because it does not have a constructor that takes in a string.

```
import java.util.Scanner;
public class TestOrder {

    public TestOrder() {
        System.out.println("I am in the TestOrder
        Constructor");
    }

    public static void main(String[] args) {
        String lastName = null;
        String lastName2 = null;
        Scanner input = new Scanner(System.in);

        Order order1 = new Order();
        BatchOrder batch1 = new BatchOrder();
        OverSeasOrder sea1 = new OverSeasOrder();

        System.out.println("Please Enter last
        name");
        lastName = input.next();
        Order order2 = new Order(lastName);

        System.out.println("Please Enter another
        last name");
        lastName2 = input.next();
        BatchOrder batch2 = new BatchOrder();
    }
}
```

Step 6 – update the BatchOrder class to create a new constructor that runs the Order’s constructor that takes in a String and creates an order ID

```
public class BatchOrder extends Order{
    public BatchOrder() {
        System.out.println("I am in the BatchOrder
        Constructor");
    }

    public BatchOrder(String surname) {
        super(surname);
        System.out.println("I am in the BatchOrder
        Constructor");
    }
}
```

<p>Step 7 – update the TestOrder class to use the new constructor in the BatchOrder Class</p> <p>Update the object batch2 of type BatchOrder using its new constructor which runs the Order constructor using the super reference.</p>	<pre>import java.util.Scanner; public class TestOrder { public TestOrder() { System.out.println("I am in the TestOrder Constructor"); } public static void main(String[] args) { String lastName = null; String lastName2 = null; Scanner input = new Scanner(System.in); Order order1 = new Order(); BatchOrder batch1 = new BatchOrder(); OverSeasOrder seal = new OverSeasOrder(); System.out.println("Please Enter last name"); lastName = input.next(); Order order2 = new Order(lastName); System.out.println("Please Enter another last name"); lastName2 = input.next(); BatchOrder batch2 = new BatchOrder(lastName2); } }</pre>
<p>Step 8 – Save all, fix any typos you may have made, run TestOrder</p> <p>Some of the new code is this:</p> <pre>System.out.println("Please Enter last name"); lastName = input.next(); Order order2 = new Order(lastName); System.out.println("Please Enter another last name"); lastName2 = input.next(); BatchOrder batch2 = new BatchOrder(lastName2);</pre> <p>We have seen code like this before, notice these constructors take in a String</p>	<p>The output from the System.out.println's is as follows:</p> <pre>I am in the Order Constructor I am in the Order Constructor I am in the BatchOrder Constructor I am in the Order Constructor I am in the BatchOrder Constructor I am in the OverSeasOrder Constructor Please Enter last name everett I am in the Order Constructor that takes a string as a parameter Your Order ID is: everett1203398018790 Please Enter another last name minich I am in the Order Constructor that takes a string as a parameter Your Order ID is: minich1203398040070 I am in the BatchOrder Constructor that takes a string as a parameter</pre>

<p>Step 9 – update the OverseasOrder class to create a new constructor that runs the BatchOrder’s constructor which runs the Order’s that takes in a String and creates an order ID</p>	<pre> public class OverseasOrder extends BatchOrder{ public OverseasOrder() { System.out.println("I am in the OverSeasOrder Constructor"); } public OverseasOrder(String namelast) { super(namelast); System.out.println("I am in the OverSeasOrder Constructor that takes in a String"); } } </pre>
<p>Step 10 – update the TestOrder class to use the new constructor in the OverseasOrder Class</p> <p>Create the object sea2 of type OverseasOrder using its new constructor which runs the BatchOrder constructor using the super reference which runs the Order constructor to create an Order ID.</p>	<pre> import java.util.Scanner; public class TestOrder { public TestOrder() { System.out.println("I am in the TestOrder Constructor"); } public static void main(String[] args) { String lastName = null; String lastName2 = null; String lastName3 = null; Scanner input = new Scanner(System.in); Order order1 = new Order(); BatchOrder batch1 = new BatchOrder(); OverseasOrder sea1 = new OverseasOrder(); System.out.println("Please Enter last name"); lastName = input.next(); Order order2 = new Order(lastName); System.out.println("Please Enter another last name"); lastName2 = input.next(); BatchOrder batch2 = new BatchOrder(lastName2); System.out.println("Please Enter a third last name"); lastName3 = input.next(); OverseasOrder sea2 = new OverseasOrder(lastName3); } } </pre>

Step 11 – Save all, fix any typos you may have made, run TestOrder

Some of the new code is as follows:

```
System.out.println("Please Enter a  
third last name");  
lastName3 = input.next();  
OverSeasOrder sea2 = new  
OverSeasOrder(lastName3);
```

```
I am in the Order Constructor  
I am in the Order Constructor  
I am in the BatchOrder Constructor  
I am in the Order Constructor  
I am in the BatchOrder Constructor  
I am in the OverSeasOrder Constructor  
Please Enter last name  
everett  
I am in the Order Constructor that takes a  
string as a parameter  
Your Order ID is: everett1203398018790  
Please Enter another last name  
minich  
I am in the Order Constructor that takes a  
string as a parameter  
Your Order ID is: minich1203398040070  
I am in the BatchOrder Constructor that  
takes a string as a parameter  
Please Enter a third last name  
jones  
I am in the Order Constructor that takes a  
string as a parameter  
Your Order ID is: jones1203398050796  
I am in the BatchOrder Constructor that  
takes a string as a parameter  
I am in the OverSeasOrder Constructor that  
takes in a String
```

Step 12 add a processOrder() method to the Order class

At this point we are not going to do anything in the processOrder() method other than print a message.

Notice that this new method lives in the Order Class

Object

- Order (method lives here)
- BatchOrder
- OverSeasOrder

```
import java.util.GregorianCalendar;
public class Order {

    private String id = null;
    private GregorianCalendar currentTime = new
    GregorianCalendar();

    public Order() {
        System.out.println("I am in the Order
        Constructor");
    }

    public Order(String lastName) {
        System.out.println("I am in the Order
        Constructor that takes a string as a
        parameter");
        id = createOrderID(lastName);
        System.out.println("Your Order ID is: " +
        id);
    }

    public String createOrderID(String
    lastName){
        return lastName +
        currentTime.getTimeInMillis();
    }

    public void processOrder(){
        System.out.println("I am in Order running
        the processOrder() method.");
    }
}
```

Step 13 use all of the instantiated objects that have the processOrder() method to invoke the processOrder() method

```
import java.util.Scanner;
public class TestOrder {
    public TestOrder() {
        System.out.println("I am in the TestOrder
        Constructor");
    }

    public static void main(String[] args) {
        String lastName = null;
        String lastName2 = null;
        String lastName3 = null;
        Scanner input = new Scanner(System.in);

        Order order1 = new Order();
        BatchOrder batch1 = new BatchOrder();
        OverSeasOrder sea1 = new OverSeasOrder();

        System.out.println("Please Enter last
        name");
        lastName = input.next();
        Order order2 = new Order(lastName);

        System.out.println("Please Enter another
        last name");
        lastName2 = input.next();
        BatchOrder batch2 = new
        BatchOrder(lastName2);

        System.out.println("Please Enter a third
        last name");
        lastName3 = input.next();
        OverSeasOrder sea2 = new
        OverSeasOrder(lastName3);

        order1.processOrder();
        order2.processOrder();
        batch1.processOrder();
        batch2.processOrder();
        sea1.processOrder();
        sea2.processOrder();
    }
}
```

Step 14 overload the processOrder() method, create another processOrder(String myString) method that takes a String parameter and uses it

```
import java.util.GregorianCalendar;
public class Order {
    private String id = "default";
    private GregorianCalendar currentTime =
        new GregorianCalendar();

    public Order() {
        System.out.println("I am in the Order
        Constructor");
        id = "default";
    }
    public Order(String lastName) {
        System.out.println("I am in the Order
        Constructor that takes a string as a
        parameter");
        id = createOrderID(lastName);
        System.out.println("Your Order ID is: " +
        id);
    }

    public String createOrderID(String
    lastName){
        return lastName +
        currentTime.getTimeInMillis();
    }

    public void processOrder(){
        System.out.println("I am in Order running
        the processOrder() method.");
    }

    public void processOrder(String orderid){
        String newID = null;
        if (id.equals("default")) {
            newID = createOrderID(id);
            System.out.println("processing order in
            Order using id: " + newID);
        } else {
            System.out.println("processing order in
            Order using id: " + orderid);
        }
    }
}
```

Step 15 – update the TestOrder class to use the new processOrder method that takes a String

```
import java.util.Scanner;
public class TestOrder {
    public TestOrder() {
        System.out.println("I am in the TestOrder
        Constructor");
    }

    public static void main(String[] args) {
        String lastName = null;
        String lastName2 = null;
        String lastName3 = null;
        Scanner input = new Scanner(System.in);

        Order order1 = new Order();
        BatchOrder batch1 = new BatchOrder();
        OverSeasOrder sea1 = new OverSeasOrder();

        System.out.println("Please Enter last
        name");
        lastName = input.next();
        Order order2 = new Order(lastName);

        System.out.println("Please Enter another
        last name");
        lastName2 = input.next();
        BatchOrder batch2 = new
        BatchOrder(lastName2);

        System.out.println("Please Enter a third
        last name");
        lastName3 = input.next();
        OverSeasOrder sea2 = new
        OverSeasOrder(lastName3);

        order1.processOrder();
        order2.processOrder();
        batch1.processOrder();
        batch2.processOrder();
        sea1.processOrder();
        sea2.processOrder();
        sea2.processOrder(lastName2);

        OverSeasOrder sea3 = new OverSeasOrder();
        sea3.processOrder(lastName3);
    }
}
```

```
I am in the Order Constructor
I am in the Order Constructor
I am in the BatchOrder Constructor
I am in the Order Constructor
I am in the BatchOrder Constructor
I am in the OverSeasOrder Constructor
Please Enter last name
e
I am in the Order Constructor that takes a
string as a parameter
Your Order ID is: e1203555252309
Please Enter another last name
f
I am in the Order Constructor that takes a
string as a parameter
Your Order ID is: f1203555255444
I am in the BatchOrder Constructor that
takes a string as a parameter
Please Enter a third last name
g
I am in the Order Constructor that takes a
string as a parameter
Your Order ID is: g1203555259049
I am in the BatchOrder Constructor that
takes a string as a parameter
I am in the OverSeasOrder Constructor that
takes in a String
I am in Order running the processOrder()
method.
I am in Order running the processOrder()
method.
I am in Order running the processOrder()
method.
I am in Order running the processOrder()
method.
I am in Order running the processOrder()
method.
I am in Order running the processOrder()
method.
I am in Order running the processOrder()
method.
processing order in Order using id: f
I am in the Order Constructor
I am in the BatchOrder Constructor
I am in the OverSeasOrder Constructor
processing order in Order using id:
default1203555259049
```