



PROMINEO TECH

Final Project Work

Points possible: 100

Category	Criteria	% of Grade
Functionality	Does the code work?	25
Organization	Is the code clean and organized? Proper use of white space, syntax, and consistency are utilized. Names and comments are concise and clear.	25
Creativity	Student solved the problems presented in the assignment using creativity and out of the box thinking.	25
Completeness	All requirements of the assignment are complete that were committed to for the given week.	25

Instructions: In Eclipse, or an IDE of your choice, write the code that accomplishes the objectives listed below. Ensure that the code compiles and runs as directed. Take screenshots of the code and of the running program (make sure to get screenshots of all required functionality) and paste them in this document where instructed below. Create a new repository on GitHub for this week's assignments and push this document, with your Java project code, to the repository. Add the URL for this week's repository to this document where instructed and submit this document to your instructor when complete.

Coding Steps:

Continue contributing to your final project.

URL to GitHub Repository: <https://github.com/danielleyokley/finalApi>

Screenshots of Code: Finished up all classes in my remaining packages (util, service and controller)

Screenshots below :



PROMINEO TECH

```
finalProjectApi  MembershipLevel  OrderService.java  SustainabilityP  »»
1 package com.promineotech.finalProjectApi.util;
2
3 6 references
4 public enum SustainabilityProceeds {
5     DONATIONFIVEPERCENT(.05),
6     DONATIONTENPERCENT(.10),
7     DONATIONFIFTEENPERCENT(.15),
8     DONATIONTWENTYPERCENT(.20);
9
10 2 references
11 private double proceed;
12
13 4 references
14 SustainabilityProceeds(double proceed) {
15     this.proceed = proceed;
16 }
17
18 1 reference
19 public double getProceed() {
20     return proceed;
21 }
22 }
```

```
SustainabilityP  OrderStatus.java  AddressRepository  CustomerService  »» 11
1 package com.promineotech.finalProjectApi.service;
2
3 import org.apache.logging.log4j.LogManager;
4
11
12 @Service
13 4 references
14 public class CustomerService {
15
16     3 references
17     private static final Logger logger = LogManager.getLogger(CustomerService.class);
18
19     @Autowired
20     6 references
21     private CustomerRepository repo;
22
23     1 reference
24     public Customer getById(Long id) throws Exception {
25         try {
26             return repo.findOne(id);
27         } catch (Exception e) {
28             logger.error("Exception occurred while trying to retrieve customer: " + id, e);
29             throw e;
30         }
31     }
32
33     1 reference
34     public Iterable<Customer> getCustomers() {
35         return repo.findAll();
36     }
37
38     1 reference
39     public Customer createCustomer(Customer customer) {
40         return repo.save(customer);
41     }
42
43     1 reference
44     public Customer updateCustomer(Customer customer, Long id) throws Exception {
45         try {
46             Customer oldCustomer = repo.findOne(id);
47             oldCustomer.setAddress(customer.getAddress());
48             oldCustomer.setFirstName(customer.getFirstName());
49             oldCustomer.setLastName(customer.getLastName());
50             oldCustomer.setReviews(customer.getReviews());
51             return repo.save(oldCustomer);
52         } catch (Exception e) {
53             logger.error("Exception occurred while trying to update customer: " + id, e);
54             throw new Exception("Unable to update customer.");
55         }
56     }
57
58     1 reference
59     public void deleteCustomer(Long id) throws Exception {
60         try {
61             repo.delete(id);
62         } catch (Exception e) {
63             logger.error("Exception occurred while trying to delete customer: " + id, e);
64             throw new Exception("Unable to delete customer.");
65         }
66     }
67 }
```



PROMINEO TECH

```
OrderStatus.jav  AddressReposito  CustomerService  OrderService.ja  » 12

50      Order order = repo.findOne(orderId);
51      order.setStatus(OrderStatus.CANCELED);
52      return repo.save(order);
53  } catch (Exception e) {
54      logger.error("Exception occurred while trying to cancel order: " + orderId, e);
55      throw new Exception("Unable to update order.");
56  }
57  }
58
59  1 reference
60  public Order completeOrder(Long orderId) throws Exception {
61      try {
62          Order order = repo.findOne(orderId);
63          order.setStatus(OrderStatus.DELIVERED);
64          return repo.save(order);
65      } catch (Exception e) {
66          logger.error("Exception occurred while trying to complete order : " + orderId, e);
67          throw new Exception("Unable to update order.");
68      }
69  }
70  1 reference
71  private Order initializeNewOrder(Set<Long> productIds, Customer customer) {
72      Order order = new Order();
73      order.setProducts(convertToProductSet(productRepo.findAll(productIds)));
74      order.setOrdered(LocalDate.now());
75      order.setEstimatedDelivery(LocalDate.now().plusDays(DELIVERY_DAYS));
76      order.setCustomer(customer);
77      order.setInvoiceAmount(calculateOrderTotal(order.getProducts(), customer.getProceeds()));
78      order.setStatus(OrderStatus.ORDERED);
79      addOrderToProducts(order);
80      return order;
81  }
82  1 reference
83  private void addOrderToProducts(Order order) {
84      Set<Product> products = order.getProducts();
85      for (Product product : products) {
86          product.getOrders().add(order);
87      }
88  }
89  1 reference
90  private Set<Product> convertToProductSet(Iterable<Product> iterable) {
91      Set<Product> set = new HashSet<Product>();
92      for (Product product : iterable) {
93          set.add(product);
94      }
95      return set;
96  }
97  1 reference
98  private double calculateOrderTotal(Set<Product> products, SustainabilityProceeds proceeds) {
99      double total = 0;
100      for (Product product : products) {
101          total += product.getPrice();
102      }
103      return total + total * proceeds.getProceed();
104  }
```



PROMINEO TECH

```
AddressRepository CustomerService OrderService.java ProductService.java 13
1 package com.promineotech.finalProjectApi.service;
2
3+ import org.apache.logging.log4j.LogManager;
11
12 @Service
13     4 references
14     public class ProductService {
15
16         2 references
17         private static final Logger logger = LogManager.getLogger(ProductService.class);
18
19         @Autowired
20         5 references
21         private ProductRepository repo;
22
23         1 reference
24         public Iterable<Product> getProducts() {
25             return repo.findAll();
26         }
27
28         1 reference
29         public Product createProduct(Product product) {
30             return repo.save(product);
31         }
32
33         1 reference
34         public Product updateProduct(Product product, Long id) throws Exception {
35             try {
36                 Product oldProduct = repo.findOne(id);
37                 oldProduct.setDescription(product.getDescription());
38                 oldProduct.setName(product.getName());
39                 return repo.save(oldProduct);
40             } catch (Exception e) {
41                 logger.error("Exception occurred while trying to delete product: " + id, e);
42                 throw new Exception("Unable to update product.");
43             }
44         }
45
46         1 reference
47         public void removeProduct(Long id) throws Exception {
48             try {
49                 repo.delete(id);
50             } catch (Exception e) {
51                 logger.error("Exception occurred while trying to delete product: " + id, e);
52                 throw new Exception("Unable to update product.");
53             }
54         }
55     }
56 }
```



PROMINEO TECH

```
CustomerService  OrderService.java  ProductService.  CustomerControl  »14
1 package com.promineotech.finalProjectApi.controller;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4
5 @RestController
6 @RequestMapping("/customers")
7 public class CustomerController {
8
9
10
11     @Autowired
12     5 references
13     private CustomerService service;
14
15     @RequestMapping(value="/{id}", method=RequestMethod.GET)
16     public ResponseEntity<Object> getCustomer(@PathVariable Long id) {
17         try {
18             return new ResponseEntity<Object>(service.getCustomerById(id), HttpStatus.OK);
19         } catch (Exception e) {
20             return new ResponseEntity<Object>(e.getMessage(), HttpStatus.NOT_FOUND);
21         }
22     }
23
24     @RequestMapping(method=RequestMethod.GET)
25     public ResponseEntity<Object> getCustomers() {
26         return new ResponseEntity<Object>(service.getCustomers(), HttpStatus.OK);
27     }
28
29     @RequestMapping(method=RequestMethod.POST)
30     public ResponseEntity<Object> createCustomer(@RequestBody Customer customer) {
31         return new ResponseEntity<Object>(service.createCustomer(customer), HttpStatus.CREATED);
32     }
33
34     @RequestMapping(value="/{id}", method=RequestMethod.PUT)
35     public ResponseEntity<Object> updateCustomer(@RequestBody Customer customer, @PathVariable Long id) {
36         try {
37             return new ResponseEntity<Object>(service.updateCustomer(customer, id), HttpStatus.OK);
38         } catch (Exception e) {
39             return new ResponseEntity<Object>(e.getMessage(), HttpStatus.NOT_FOUND);
40         }
41     }
42
43     @RequestMapping(value="/{id}", method=RequestMethod.DELETE)
44     public ResponseEntity<Object> deleteCustomer(@PathVariable Long id) {
45         try {
46             service.deleteCustomer(id);
47             return new ResponseEntity<Object>("Successfully deleted customer with id: " + id, HttpStatus.OK);
48         } catch (Exception e) {
49             return new ResponseEntity<Object>(e.getMessage(), HttpStatus.NOT_FOUND);
50         }
51     }
52 }
53
54
55
56
57
58
59
60
```



PROMINEO TECH

```
CustomerService | ProductService. | CustomerControl | OrderController 15
1 package com.promineotech.finalProjectApi.controller;
2
3 import java.util.Set;
17
18 @RestController
19 @RequestMapping("customers/{id}/orders")
20 public class OrderController {
21
22     @Autowired
23     3 references
24     private OrderService service;
25
26     @RequestMapping(method=RequestMethod.POST)
27     public ResponseEntity<Object> createCustomer(@RequestBody Set<Long> productIds, @PathVariable Long id) {
28         try {
29             return new ResponseEntity<Object>(service.submitNewOrder(productIds, id), HttpStatus.CREATED);
30         } catch (Exception e) {
31             return new ResponseEntity<Object>(e, HttpStatus.BAD_REQUEST);
32         }
33
34     @RequestMapping(value="/{orderId}", method=RequestMethod.PUT)
35     public ResponseEntity<Object> updateOrder(@RequestBody Order order, @PathVariable Long orderId) {
36         try {
37             if (order.getStatus().equals(OrderStatus.CANCELED)) {
38                 return new ResponseEntity<Object>(service.cancelOrder(orderId), HttpStatus.OK);
39             } else if (order.getStatus().equals(OrderStatus.DELIVERED)) {
40                 return new ResponseEntity<Object>(service.completeOrder(orderId), HttpStatus.OK);
41             }
42             return new ResponseEntity<Object>("Invalid update request.", HttpStatus.BAD_REQUEST);
43         } catch (Exception e) {
44             return new ResponseEntity<Object>(e.getMessage(), HttpStatus.NOT_FOUND);
45         }
46     }
47 }
48 }
```




PROMINEO TECH

```
CustomerService CustomerControl OrderController ProductControll 16
1 package com.promineotech.finalProjectApi.controller;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4
14 @RestController
15 @RequestMapping("/products")
16 public class ProductController {
17
18     @Autowired
19     4 references
20     private ProductService service;
21
22     @RequestMapping(method=RequestMethod.GET)
23     public ResponseEntity<Object> getProducts() {
24         return new ResponseEntity<Object> (service.getProducts(), HttpStatus.OK);
25     }
26
27     @RequestMapping(method=RequestMethod.POST)
28     public ResponseEntity<Object> createProduct(@RequestBody Product product) {
29         return new ResponseEntity<Object> (service.createProduct(product), HttpStatus.CREATED);
30     }
31     @RequestMapping(value="/{id}", method=RequestMethod.PUT)
32     public ResponseEntity<Object> updateProduct(@RequestBody Product product, @PathVariable Long id) {
33         try {
34             return new ResponseEntity<Object>(service.updateProduct(product, id), HttpStatus.OK);
35         } catch (Exception e) {
36             return new ResponseEntity<Object>("Unable to update product.", HttpStatus.BAD_REQUEST);
37         }
38     }
39
40     @RequestMapping(value="/{id}", method=RequestMethod.DELETE)
41     public ResponseEntity<Object> deleteProduct(@PathVariable Long id) {
42         try {
43             service.removeProduct(id);
44             return new ResponseEntity<Object>("Successfully deleted product with id: " + id, HttpStatus.OK);
45         } catch (Exception e) {
46             return new ResponseEntity<Object>("Unable to delete product.", HttpStatus.BAD_REQUEST);
47         }
48     }
49 }
50
51
```



PROMINEO TECH

```
OrderStatus.java AddressRepository CustomerService OrderService.java 12
1 kage com.promineotech.finalProjectApi.service;
2
3 import java.time.LocalDate;
21
22 service
23 references
24
25 3 references
26 private static final Logger logger = LogManager.getLogger(OrderService.class);
27 1 reference
28 private final int DELIVERY_DAYS = 7;
29
30 @Autowired
31 5 references
32 private OrderRepository repo;
33
34 @Autowired
35 1 reference
36 private CustomerRepository customerRepo;
37
38 @Autowired
39 1 reference
40 private ProductRepository productRepo;
41
42 1 reference
43 public Order submitNewOrder(Set<Long> productIds, Long customerId) throws Exception {
44     try {
45         Customer customer = customerRepo.findOne(customerId);
46         Order order = initializeNewOrder(productIds, customer);
47         return repo.save(order);
48     } catch (Exception e) {
49         logger.error("Exception occurred while trying to create a new order for customer : " + customerId, e);
50         throw e;
51     }
52 }
53
54 1 reference
55 public Order cancelOrder(Long orderId) throws Exception {
56     try {
57         Order order = repo.findOne(orderId);
58         order.setStatus(OrderStatus.CANCELED);
59         return repo.save(order);
60     } catch (Exception e) {
61         logger.error("Exception occurred while trying to cancel order: " + orderId, e);
62         throw new Exception("Unable to update order.");
63     }
64 }
65
66 1 reference
67 public Order completeOrder(Long orderId) throws Exception {
68     try {
69         Order order = repo.findOne(orderId);
70         order.setStatus(OrderStatus.DELIVERED);
71         return repo.save(order);
72     } catch (Exception e) {
73         logger.error("Exception occurred while trying to complete order : " + orderId, e);
74         throw new Exception("Unable to update order.");
75     }
76 }
```