|  |  |  |  |
| --- | --- | --- | --- |
| 1 | Introductions - explain about the technical work done in your recent project. | **NDR(non directional )DR pages**  CounterPartyHistory  created Apis get Non directional ->(classes : NondirectonalCustomers vs directonalCustomers)  Retrieve counterparty details from customer service  ->counterparty id and counterparty name,past debts,Addons such as liquidity addon,basis addons from customer service  ->passing counter partyid,business date to get Trade details(trade count,trade id,worst case scenarios)  ->wrote logic to find out risexposure by using measurement,creditAmount  ->written validation service to validate counterparty details(counterparty id,business date,compare businate)  ->writen global exeception handler to handle customernotfound,tradeDetailsNotfound  ->Fiegn client to connect customer service and trade service  ->introduced caching mechnism    **improved customer history pages(Class :customerHisotry):**  ->optimized query by changing below as,  -> Avoid subqueries and use joins,avoid unnessary joins  ->rewrite to customer logic(introduce caching mechanism to avoid unnessary db hits)  ->combining tables such cust\_crossAddons and cust\_riskAddons(update service,entity,)  ->unnecessary usage of code  **calender tooltip(Classes :StressData) :** if you select particular date, we may know if t+1 or t2 are available  ->wrote api to fetch t1 table or t2 tables found or not by passing counterpary id and businessdate?  findout number of worstcaseScenario where counterpartymembers fall into that?  Api ->SensitivityDetails->passing sensitivityId and BusinessDate to fetch the Sensitivitycustomers data  ->For each customerid,find out worstcaseScenario  customerList.stream().collect(Collectors.groupingBy(Customer::getCname,Collectors.mapping(Customer::getWorstCaseName, Collectors.toSet())))  .forEach((k,v)->System.out.println("customer Name :: " +k +" List of worst cases : " +v));  GroupBy used:  ----------  Map<Object, Set<stressdata>> counterpartyByWorstCaseScenario = counterpartyByWorstCaseScenario.stream().collect(Collectors.groupingBy(cw->cw.getscenarioId(),Collectors.toset()));  countOfworstCase::  List<Customer> customerList = Arrays.asList(  new Customer("trade1", "worstcase4", random.get()),  new Customer("trade4", "worstcase2", random.get()),  new Customer("trade3", "worstcase1", random.get()),  new Customer("trade2", "worstcase4", random.get()),  new Customer("trade5", "worstcase1", random.get()),  new Customer("trade6", "worstcase1", random.get()),  new Customer("trade7", "worstcase2", random.get()),  new Customer("trade8", "worstcase1", random.get())  );  customerList.stream().collect(Collectors.groupingBy(Customer::getWorstCaseName,Collectors.counting()))  .forEach((k,v)->System.out.println("worstcase Name :: " +k +" total of worst cases : " +v));  customerList.stream().collect(Collectors.groupingBy(Customer::getCname,Collectors.mapping(Customer::getWorstCaseName, Collectors.toSet())))  .forEach((k,v)->System.out.println("customer Name :: " +k +" List of worst cases : " +v));  Output:  worstcase Name :: worstcase2 total of worst cases : 2  worstcase Name :: worstcase1 total of worst cases : 4  worstcase Name :: worstcase4 total of worst cases : 2  customer Name :: trade4 List of worst cases : [worstcase2, worstcase1]  customer Name :: trade2 List of worst cases : [worstcase1, worstcase4]  customer Name :: trade3 List of worst cases : [worstcase1]  customer Name :: trade1 List of worst cases : [worstcase2, worstcase4]  Predicate:  ------------  Predicate<Customer> pred = customer->customer.getCreditRiskAmt()>10000;  if(pred.test(customer))  {  }  Function:  ---------  default dollar. converting to euro.  Function<Double,Double> covertedRate = (input) -> input \* dollar\_euro\_rate;  rate = covertedRate.apply(input);  Supplier<Integer> rds = () -> new Random().nextInt(10);    ->  Junit:  --------------  import org.junit.Before;  import org.junit.Test;  import org.mockito.InOrder;  import static org.mockito.Mockito.\*;  import org.mockito.Mock;  public class NDRCustomerTest {  //private ProductService productService;  private NDRCustomerDao NDRCustomerDao;  private Product product;  private int purchaseQuantity = 15;  List<WorstCases> ListWorstcase;  List<Exchange> ListExchanges;  @Before  public void setupMock() {  productService = new ProductService();  product = mock(Product.class);  productDao = mock(ProductDao.class);  productService.setProductDao(productDao);  ListExchanges = Arrays.asList(new Exchange(exname,rate,date));  ListWorstCases = Arrays.asList(new WorstCases(exname,rate,date));    }  @Test  public void testGetNDRustomerData() throws InsufficientNDRCustomerException {    System.out.println("Stubbing getAvailableProducts(product) to return " + availableQuantity);  when(productDao.getAvailableExchangesRates(transactionId,businessdate)).thenReturn(ListExchanges);  when(productDao.getWorscastcases(transactionId,businessdate)).thenReturn(ListWorstcase);  List data = NDRCustomerDao.GetNDRustomerData(transactionId, businessdate);  AssertTrue(data.size()>0)  }  @Test(expected = InsufficientCustomerCreditException.class)  public void purchaseWithInsufficientAvailableQuantity() throws InsufficientProductsException {  int availableQuantity = 3;  System.out.println("Stubbing getAvailableProducts(product) to return " + availableQuantity);  when(productDao.getAvailableProducts(product)).thenReturn(availableQuantity);  try {  System.out.println("productService.buy(product" + purchaseQuantity + ") should throw InsufficientProductsException");  productService.buy(product, purchaseQuantity);  } catch (InsufficientCustomerCreditException e) {  System.out.println("InsufficientProductsException has been thrown");  verify(productDao, times(0)).orderProduct(product, purchaseQuantity);  System.out.println("Verified orderProduct(product, " + purchaseQuantity + ") is not called");  throw e;  }  }  }  @Test  void whenValidInput\_thenReturns200() throws Exception {  UserResource user = new UserResource("Zaphod", "zaphod@galaxy.net");    mockMvc.perform(post("/forums/{forumId}/register", 42L)  .contentType("application/json")  .param("sendWelcomeMail", "true")  .content(objectMapper.writeValueAsString(user)))  .andExpect(status().isOk());  } |  |
| 2 | Explain the Java11 features used. | 1.Running Java Files Directly(No need to compile the single file and directly run it.)  2.Improved var Handling(more readable and writable, we can use it in lambda expression)  // Pre-Java 11 Function<String, Integer> preJava11Func = (String s) -> Integer.parseInt(s) + 10; // Java 11 Function<String, Integer> java11Func = (var s) -> Integer.parseInt(s) + 10; 3.HTTP Client API ([JEP 321](https://openjdk.org/jeps/321)) Java 11 introduces a new HTTP Client API to replace the older HttpURLConnection and provide a more modern, convenient way to access HTTP resources. The new API supports HTTP/1.1, HTTP/2, and WebSocket communication and can handle request and response bodies using the reactive-streams API.  Here’s an example illustrating the use of the new HTTP Client API:  import java.net.URI; import java.net.http.HttpClient; import java.net.http.HttpRequest; import java.net.http.HttpResponse;  public class HttpClientDemo {  public static void main(String[] args) throws Exception {  HttpClient httpClient = HttpClient.newHttpClient();   HttpRequest request = HttpRequest.newBuilder()  .uri(URI.create("https://api.example.com/data"))  .GET()  .build();   HttpResponse<String> response = httpClient.send(request, HttpResponse.BodyHandlers.ofString());   System.out.println(response.statusCode());  System.out.println(response.body());  } }  4.New Utility Methods for String  isBlank(): Checks if a string is empty or contains only white space characters.  String nonBlank = "Hello, world!";  boolean isNonBlank = nonBlank.isBlank(); // false  String blank = " \t ";  boolean isBlank = blank.isBlank(); // true  2. strip(), stripLeading(), and stripTrailing: Strips leading and/or trailing white spaces from a string, including Unicode white space characters beyond the ASCII space character.  String text = " Java 11 ";  String stripped = text.strip(); // "Java 11"  String leadingStripped = text.stripLeading(); // "Java 11 "  String trailingStripped = text.stripTrailing(); // " Java 11"  3. lines(): Returns a stream of lines extracted from the given string, separated by line terminators.  String multilineText = "Line 1\nLine 2\nLine 3";  multilineText.lines().forEach(System.out::println);  4. repeat(int): Repeats the string the specified number of times.  String repeatedText = "ABC".repeat(3); // "ABCABCABC"  Java Flight Recorder (JFR) Event Streaming (JEP 328)  Java 11 adds streaming capabilities to the Java Flight Recorder (JFR), a profiling and event collection framework. This feature allows developers to access data during application execution, enabling more robust monitoring and profiling of Java applications. |  |
| 3 | How are these Java11 features useful as compared to Java8? |  |  |
| 4 | How were different features implemented before Java11 and what is the benfit we get out of using Java11? |  |  |
| 5 | Explain the recent flow that you have coded, mention the classes, annotations used. | NDRCustomer{GetcNDRCustomerDetails(exchangeRate,worstcase),DirectionalCustomer page creation. CalendercomponentsAPIs (isStressAPIDataavailable,ISsensityAvailable)  ,Sensitivity,CusotmerHistoryServices, TaxServices Apis |  |
| 6 | Exlain the response received | 200 -succesful response(NDR,calender,sensitivity)  public class ApiErrorResponse {  private final String transcationId;  private final String businessDate;  private final String message;  private final Integer statusCode;  private final String statusName;  private final String CounterpartyId;  private final String method;  private final LocalDateTime timestamp; } |  |
| 7 | What is the version of springboot used? | 2.7 |  |
| 8 | Multithreading |  |  |
| 9 | Functional interfaces |  |  |
| 10 | Reactive programming |  |  |
| 11 | Springboot concepts |  |  |
| 12 | BDD |  |  |
| 13 | examples of where junit is used - positive & negative |  |  |
| 14 | Define UT cases for the flow you have coded |  |  |
| 15 | Setup of Kafka and how used, were there any recommendations made by you? |  |  |