





**Lab Guide for Business Tier using POJOs**

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

CONFIDENTIAL



CONFIDENTIAL

Asreet-Tech.com Technologies Limited Docum ent Revision History

**Document Rev ision History**

**Version Date Author(s) Reviewer(s) Comments**

1.0 Jun-2009



ER/CORP/CRS/ED112/007 CONFIDENTIAL Version No: 1.0 i

Asreet-Tech.com Limited Table of Contents

**Contents**

**COPY RIGHT NOTICE .................................................................................................. II DOCUM ENT R EVISION HISTORY ......................................................................................I CONTENTS ............................................................................................................. II CONTEX T ............................................................................................................... 1**



**DAY 1 ASSIGNMENTS ................................................................................................. 1**

ASSI GNMENT 1: UNDERSTANDING COLLECTIONS FRAMEWORK- LI ST...................................................... 1

ASSI GNMENT 2: UNDERSTANDING COLLECTIONS FRAMEWORK- SET ...................................................... 2

ASSI GNMENT 3: UNDERSTANDING COLLECTIONS FRAMEWORK- MAP...................................................... 4

ASSI GNMENT 4: COLLECTIONS AND GENERICS ........................................................................... 7

ASSI GNMENT 5: DEBUG THE ASSI GNMENT ..............................................................................15

ASSI GNMENT 6: MULTI THREADED PROGRAMMI NG I N JAVA...............................................................16

ASSI GNMENT 7: TO LEARN SYNCHRONI ZATI ON OF JAVA METHODS........................................................26

ASSI GNMENT 8: LEARNING HOW TO DEBUG A MULTI THREADED JAVA PROGRAM ...........................................27

ASSI GNMENT 9: EX ERCI SES FOR SELF REVIEW...........................................................................32



ER/CORP/CRS/ED112/007 CONFIDENTIAL Version No. 1.0 ii

Asreet-Tech.com Limited Lab Guide for <Co urse Nam e>

**Context**

This document contains assignments to be completed as part of the hands on for the subject

Business Tier with POJO (Course code: <course code>).

**Note:** In order to complete the course, assignments in this document must be completed in the sequence mentioned.

**Day 1 Assignments**

**Assignment 1: Understanding Collections Framework- List**

**Objective:** To understand how to use List

**Problem Description:** Create a class that stores all the customers’ name s of a bank and display it whenever required

**Estimated time: 20 Mins**

**Step 1:** Create a package in the working project and name it as bank

**Step 2:** Create a class AddCustomer as given below

**/\***

**\* This java file is a class that explains the use of List**

**\*/**





**package com.seedBank.customer;**

**import java.util.List;**

**import java.util.ArrayList;**

**import java.util.Iterator;**

**/\*\***

**\* <h3>Description :</h3> This class add the customer names to a List**

**\* and contains a method that will display the contents of the list**

**\*** [**@author**](mailto:@author) **E&R Dept.,Asreet-Tech.com Ltd.**

**\*/**

**public class AddCustomer{**

**/\*\***

**\* <h3>Description :</h3> This method is used to print the list**

**\* of customer names**

**\*** [**@param**](mailto:@param) **List<String> arrayList the collection of customer names**

**\*/**

ER/CORP/CRS/ED112/007 CONFIDENTIAL Version No. 1.0 1 of 32

Asreet-Tech.com Limited Lab Guide for <Co urse Nam e>

**public void printCustomers(List<String> arrayList){**

**//Create the iterator to iterate over the List Iterator<String> iterator=arrayList.iterator(); System.out.println("Customers are : "); while(iterator.hasNext()){**

**System.out.println(iterator.next());**

**}**

**}**

**/\*\***

**\* <h3>Description :</h3> The starter method for the application**

**\*** [**@param**](mailto:@param) **Sting [] args The command line arguments**

**\*/**

**public static void main(String args[]){**

**//Create reference of List and object of ArrayList List<String> list=new ArrayList<String>(); list.add("Jack");**



**list.add("Harry"); list.add("Tabrez"); list.add("Leezu");**

**AddCustomer addCustomer=new AddCustomer();**

**//Call printCustomers()to print the contents of list addCustomer.printCustomers(list);**

**}**

**}**

**Summary of this assignment:**

 How to create List and addi ng elements to it

 How to iterate over the list to print its contents

**Note:** In the same way you can also create LinkedList and use the extra methods of LinkedList like addFirst() and addLast()





**Assignment 2: Understanding Collections Framework- Set**

**Objective:** To understand how to use Set

ER/CORP/CRS/ED112/007 CONFIDENTIAL Version No. 1.0 2 of 32

Asreet-Tech.com Limited Lab Guide for <Co urse Nam e>

**Problem Description:** Create a class that stores some records to a Set and then prints the records if the Set is not empty.

**Estimated time: 20 Mins**

**Step 1:** Create a class EntrySet as below

**Step 2:** Create HashSet and add records to it and print the records





**/\***

**\* This java file is a class that explains the use of Set**

**\*/**

**package com.seedBank.records;**

**import java.util.Set;**

**import java.util.HashSet;**

**/\*\***

**\* <h3>Description :</h3>This class add records to Set and contains a**

**\* method that will display the records of the Set**

**\*** [**@author**](mailto:@author) **E&R Dept.,Asreet-Tech.com Ltd.**

**\*/**

**public class EntrySet{**

**/\*\***

**\* <h3>Description :</h3>This method will take Set as input and**

**\* if it is empty set prints a message and then print the records**

**\* in the set**

**\*** [**@param**](mailto:@param) **ser set of records**

**\*/**

**public void printSetRecords(Set<String> set){**

**if(set.isEmpty()){**

**System.out.println("Set is an empty set");**

**}else {**

**System.out.println("Set elements are :"+ set);**

**}**

**}**

**/\*\***

**\* <h3>Description :</h3> The starter method for the application**

**\*** [**@param**](mailto:@param) **Sting [] args The command line arguments**

**\*/**

**public static void main(String args[]){**

**//Create HashSet and add some entries to it**

**Set<String> set=new HashSet<String>();**

**set.add("First Entry");**

ER/CORP/CRS/ED112/007 CONFIDENTIAL Version No. 1.0 3 of 32

Asreet-Tech.com Limited Lab Guide for <Co urse Nam e>

**set.add("Second Entry");**

**set.add("Third Entry");**

**//Try to enter the same entry again set.add("First Entry");**

**EntrySet setDemo=new EntrySet ();**

**//It will print records of the set setDemo.printSetRecords(set);**

**//Create a new empty HashSet**

**Set<String> emptySet=new HashSet<String>();**

**//It will say that the set is an empty set setDemo.printSetRecords(emptySet);**

**}**

**}**

**Summary of this assignment:**

 How to create Set and adding eleme nts to it

 How to print the records of a Set

 How to check whether the Set is empty or not











**Note:** One can also use size() method to check whether the Set is empty or not and can also display the number or records in a Set

**Note:** Set records can also be printed usi ng Iterator in the same way as that of List

**Note:** In the same way LinkedHashSet and TreeSet can also be created

LinkedHashSet : It will insert the records by insertion order

TreeSet : It will insert the records into the Set in sorted order

**Assignment 3: Understanding Collections Framework- Map**

**Objective:** To understand how to use Map

**Problem Description:** Create a class that stores userid and corresponding password and prints password for a given userid

ER/CORP/CRS/ED112/007 CONFIDENTIAL Version No. 1.0 4 of 32

Asreet-Tech.com Limited Lab Guide for <Co urse Nam e>

**Estimated time: 25 Mins**

**Step 1:** Create a class LoginDetai ls under **com.seedBank.login** package as given below

**Step 2:** Create a Map with <Integer,String> and store userid and password in the map

**/\***

**\* This file contains a class that explains how to Map with generic**

**\*/**

**package com.seedBank.login;**

**import java.util.Map;**

**import java.util.LinkedHashMap;**

**import java.util.Set;**

**import java.util.Iterator;**

**/\*\***

**\* <h3>Description :</h3>This class add userid and corresponding**

**\* password to Map and contains a method that will iterate over the**

**\* Set of keys of Map**

**\*** [**@author**](mailto:@author) **E&R Dept.,Asreet-Tech.com Ltd.**

**\*/**

**public class LoginDetails{**

**/\*\***

**\* <h3>Description :</h3> This method will take**

**\* Map<Integer, String> as input and will check for a particular**

**\* key in the Map. If it is present, then print "Key is already**

**\* present in map" and replace it with new Key, Value pair**





**\*** [**@param**](mailto:@param) **map The map to which the key is to be added**

**\*/**

**public void addSameKey(Map<Integer,String> map){**

**if(map.containsKey(1003)){**

**System.out.println("Key is already present in map");**

**}**

**/\*\***

**}**

**map.put(1003,"John");**

**\* <h3>Description :</h3> This method will take**

**\* Map<Integer,String> as input and create a Set of**

**\* keys(userid) and then print all the userid**

**\*** [**@param**](mailto:@param) **map The map containing the details of userid and**

**\* the password**

**\*/**

**public void printLoginDetails(Map<Integer,String> map){**

**//Creating a set of keys of map**

**Set<Integer> set=map.keySet();**

**Iterator<Integer> iterator=set.iterator();**

ER/CORP/CRS/ED112/007 CONFIDENTIAL Version No. 1.0 5 of 32

Asreet-Tech.com Limited Lab Guide for <Co urse Nam e>

**while(iterator.hasNext()){ System.out.println(iterator.next());**

**}**

**}**

**/\*\***

**\* <h3>Description :</h3> The starter method for the application**

**\*** [**@param**](mailto:@param) **Sting [] args the command line arguments**

**\*/**

**public static void main(String args[]){ Map<Integer,String> map=new**

**LinkedHashMap<Integer,String>();**

**map.put(1000,"Sam"); map.put(1001,"Tom"); map.put(1002,"Zen"); map.put(1003,"Zen");**

**System.out.println("Map before adding the same key 1003"+**

**map);**



**LoginDetails loginDetails=new LoginDetails();**

**loginDetails.addSameKey(map);**

**System.out.println("Value for 1003, after addSameKey() : "+map.get(1003));**

**System.out.println("Map after adding the same key 1003"+**

**map);**

**loginDetails.printLoginDetails(map);**

**}**

**}**

**Summary of this assignment:**

 How to create Map with generics and putting key, values pair to it

 How to find a particular key in the Map

 How to get Set of keys from Map

**Note:** Map is not a Collection so one cannot iterate over the Map. To iterate over it, convert it into Set and then use iterator





**Note:** The value can be repeated in map but keys cannot be repeated, if you try to add same key with new value, it will replace the previous key, value pair with new one.



ER/CORP/CRS/ED112/007 CONFIDENTIAL Version No. 1.0 6 of 32

Asreet-Tech.com Limited Lab Guide for <Co urse Nam e>

**Assignment 4: Collections and Generics**

**Objective:** To learn how to use the collections and ge neric features.

**Problem Description:** Automating the add customer feature in SeedBank by usi ng the collections framework and generics.

Classes required:

**Customer:** A customer object represents the customers of SeedBank **CustomerDB:** A customerDB class contains methods to save the details of the customer

**Login:** This class contains methods related to authe ntication

**SeedBank:** The starter class of the entire application

**Estimated time: 15 Mins**

**Step 1:** Create a Customer bean class representi ng the customer of SeedBank.

**/\*\***

**\* This file contains the Customer Bean class**

**\*/**

**package com.seed.customer;**

**/\*\***

**\* <h3> Description :</h3> The Customer bean class represents a**

**\* customer of SeedBank**

**\*** [**@author**](mailto:@author) **E&R Dept, Asreet-Tech.com Ltd.**

**\*/**





**public class Customer{**

**/\***

**\* The customer Id of the customer**

**\*/**

**private int customerId;**

**/\***

**\* The name of the customer**

**\*/**

**private String customerName;**

**/\***

**\* The status of the cuatomer if has already availed the loan**

**\*/**

**private boolean loanAvailed;**

**/\*\***

**\* <h3>Description :</h3>This overloaded constructor is used to**

**\* initialize the Customer object**

ER/CORP/CRS/ED112/007 CONFIDENTIAL Version No. 1.0 7 of 32

Asreet-Tech.com Limited Lab Guide for <Co urse Nam e>

**\*** [**@param**](mailto:@param) **customerId The customer Id of the customer**

**\*** [**@param**](mailto:@param) **customerName The name of the customer**

**\*** [**@param**](mailto:@param) **loanAvailed The status of the customer if he has**

**\* availed the loan**

**\*/**

**public Customer(int customerId,String customerName,boolean loanAvailed){**

**this.customerId=customerId;**

**this.customerName=customerName;**

**this.loanAvailed=loanAvailed;**

**}**

**/\*\***

**\* <h3>Description :</h3>This overloaded constructor is used to**

**\* initialize the Customer object**

**\*** [**@param**](mailto:@param) **customerId The customer Id of the customer**

**\*** [**@param**](mailto:@param) **customerName The name of the customer**

**\*/**

**public Customer(int customerId,String customerName){**

**this.customerId=customerId;**

**this.customerName=customerName;**

**}**

**/\*\***

**\* <h3> Description :</h3> The getter method for the customer Id**

**\*** [**@return**](mailto:@return) **the customerId**





**\*/**

**public int getCustomerId() {**

**return customerId;**

**}**

**/\*\***

**\* <h3> Description :</h3> The setter method for the customer Id**

**\*** [**@param**](mailto:@param) **customerId the customerId to set**

**\*/**

**public void setCustomerId(int customerId) {**

**this.customerId = customerId;**

**}**

**/\*\***

**\* <h3> Description :</h3> The getter method for the customer**

**\* name**

**\*** [**@return**](mailto:@return) **the customerName**

**\*/**

**public String getCustomerName() {**

**return customerName;**

**}**

**/\*\***

**\* <h3> Description :</h3> The setter method for the customer**

**\* Name**

ER/CORP/CRS/ED112/007 CONFIDENTIAL Version No. 1.0 8 of 32

Asreet-Tech.com Limited Lab Guide for <Co urse Nam e>

**\*** [**@param**](mailto:@param) **customerName the customerName to set**

**\*/**

**public void setCustomerName(String customerName) {**

**this.customerName = customerName;**

**}**

**/\*\***

**\* <h3> Description :</h3> The getter method for the customer**

**\* loan status**

**\*** [**@return**](mailto:@return) **the loanAvailed**

**\*/**

**public boolean isLoanAvailed() {**

**return loanAvailed;**

**}**

**/\*\***

**\* <h3> Description :</h3> The setter method for the customer**

**\* loan status**

**\*** [**@param**](mailto:@param) **loanAvailed the loanAvailed to set**

**\*/**

**public void setLoanAvailed(boolean loanAvailed) {**

**this.loanAvailed = loanAvailed;**

**}**

**}**

**Step 2:** Create the CustomerDB class that’s helps in saving the customer data in the





List

**/\*\***

**\* This file contains the customerDB class**

**\*/**

**package com.seed.DB;**

**import java.util.ArrayList; import java.util.Iterator; import java.util.List; import java.util.Set;**

**import java.util.TreeSet;**

**import com.seed.customer.Customer;**

**/\*\***

**\* <h3>Description : </h3>This class contains methods**

**\* that deals with customer data**

**\*** [**@author**](mailto:@author) **E&R Dept, Asreet-Tech.com Ltd.**

**\***

**\*/**

ER/CORP/CRS/ED112/007 CONFIDENTIAL Version No. 1.0 9 of 32

Asreet-Tech.com Limited Lab Guide for <Co urse Nam e>

**public class CustomerDB {**

**/\***

**\* This is the list of all the customers in the**

**\*/**

**private List<Customer> customerList=new ArrayList<Customer>();**

**/\*\***

**\* <h3>Description :</h3> This method saves input**

**\* customer object in the list**

**\*** [**@param**](mailto:@param) **customer The customer bean**

**\*** [**@return**](mailto:@return) **true if the customer is successfully added else**

**\* returns false**

**\*/**

**public boolean saveCustomer(Customer customer){**

**boolean status=false;**

**//adding the customer to the list i.e saving the customer**

**//object**

**status=customerList.add(customer);**

**}**

**/\*\***





**//returning the list of customers return status;**

**\* <h3>Description :</h3> This method is used to get the details**

**\* of all the customers**

**\*** [**@return**](mailto:@return) **the list of all the customers**

**\*/**

**public List<Customer> getAllCustomers(){**

**return customerList;**

**}**

**/\*\***

**\* <h3>Description :</h3> This method returns the set of customer**

**\* ids who have availed the loan**

**\*** [**@return**](mailto:@return) **the set of customers who have availed the loan. If**

**\* there are no customers who have availed the loan then it**

**\* return an empty set.**

**\*/**

**public Set<Integer> getLoanAvailedCustomers(){**

**//creating the set of customer ids**

**Set<Integer> loanAvailedCustomers=new TreeSet<Integer>();**

**//getting the iterator for the list of customers**

**Iterator<Customer> customerIterator=**

**customerList.iterator();**

ER/CORP/CRS/ED112/007 CONFIDENTIAL Version No. 1.0 10 of 32

Asreet-Tech.com Limited Lab Guide for <Co urse Nam e>

**//iterating the list of customers while(customerIterator.hasNext()){**

**//getting the customer object**

**Customer customer=customerIterator.next();**

**//checking if the loan is availed if(customer.isLoanAvailed()){**

**//adding the customer ids to the set**

**loanAvailedCustomers.add(customer.getCustomerId());**

**}**

**}**

**//returning the set of customers who have availed the loan return loanAvailedCustomers;**

**}**

**}**

**Step 3:** Creati ng the class Login which is used to save login details required for authentication

**/\*\***

**\* This file contains the methods related to authentication**

**\*/**

**package com.seed.login;**





**import java.util.LinkedHashMap;**

**import java.util.Map;**

**/\*\***

**\* <h3>Description :</h3> This class is used for authentication**

**\*** [**@author**](mailto:@author) **E&R Dept, Asreet-Tech.com Ltd.**

**\***

**\*/**

**public class Login {**

**/\***

**\* A map containing the customer id and its corresponding**

**\* password**

**\*/**

**private Map<Integer, String> loginMap=new**

**LinkedHashMap<Integer, String>();**

**/\*\***

**\* <h3>Description :</h3> This method adds customer id and**

**\* password pair into the map for the customers availing the**

**\* online service.**

**\*** [**@param**](mailto:@param) **customerId the customer id**

**\*** [**@param**](mailto:@param) **password the password corresponding to the password**

ER/CORP/CRS/ED112/007 CONFIDENTIAL Version No. 1.0 11 of 32

Asreet-Tech.com Limited Lab Guide for <Co urse Nam e>

**\*/**

**public void addLogin(int customerId,String password){**

**//adding the input to the map**

**loginMap.put(customerId, password);**

**}**

**}**

**Step 4:** Create a SeedBank class which is the starter class for the entire application.

**/\*\***

**\* The Starter class of the entire application**

**\*/**

**package com.seed.main;**

**import java.util.Iterator;**

**import java.util.List;**

**import java.util.Set;**

**import com.seed.DB.CustomerDB;**

**import com.seed.customer.Customer;**

**import com.seed.login.Login;**

**/\*\***

**\* <h3>Description :</h3> This is the Starter class of the entire**

**\* application**

**\*** [**@author**](mailto:@author) **E&R Dept, Asreet-Tech.com Ltd.**

**\***

**\*/**





**public class SeedBank {**

**/\***

**\* creating an object of customer db to be accessed throughout**

**\* the application**

**\*/**

**private static final CustomerDB customerDB=new CustomerDB();**

**/\***

**\* Creating an object of the login**

**\*/**

**private static final Login login=new Login();**

**/\*\***

**\* <h3>Description :</h3> The starter method of the entire**

**\* application**

**\*** [**@param**](mailto:@param) **args the command line arguments for the application**

**\*/**

**public static void main(String[] args) {**

**//creating new customer and adding to the list**

ER/CORP/CRS/ED112/007 CONFIDENTIAL Version No. 1.0 12 of 32

Asreet-Tech.com Limited Lab Guide for <Co urse Nam e>

**Customer cust1=new Customer(1001,"Tan",false);**

**customerDB.saveCustomer(cust1);**

**//customer availing the online service login.addLogin(1001, "seed123");**

**Customer cust2=new Customer(1002,"John",true);**

**customerDB.saveCustomer(cust2);**

**//customer availing the online service**

**login.addLogin(1002, "seed123");**

**Customer cust3=new Customer(1003,"Sam",true);**

**customerDB.saveCustomer(cust3);**

**//customer availing the online service login.addLogin(1003,** [**"Mys@1**](mailto:Mys@123)**23");**

**Customer cust4=new Customer(1004,"Raj",false);**

**customerDB.saveCustomer(cust4);**

**//customer availing the online service login.addLogin(1004, "Asreet-Tech.com123");**

**/\***

**\* displaying the details of the customers**

**\*/**





**List<Customer> customerList=customerDB.getAllCustomers();**

**//checking if the list is empty if(customerList.isEmpty()){**

**//displaying the error message if the list is empty**

**System.out.println("No Customers in the Bank");**

**}else{**

**//calling method for displaying list of customers printCustomerList(customerList);**

**}**

**/\***

**\* displaying the list of all the customers who**

**\* have availed the loans**

**\*/**

**Set<Integer> loanAvailedCustomers =**

**customerDB.getLoanAvailedCustomers();**

**System.out.println("\nCustomers who have Availed the loan**

**");**

**System.out.println("------------------------------------");**

**for(int custId:loanAvailedCustomers){ System.out.println(custId);**

**}**

ER/CORP/CRS/ED112/007 CONFIDENTIAL Version No. 1.0 13 of 32

Asreet-Tech.com Limited Lab Guide for <Co urse Nam e>

**}**

**/\*\***

**\* <h3>Description :</h3> This method is used to print the details**

**\* of all the customers**

**\*** [**@param**](mailto:@param) **customerList the list of customers**

**\*/**

**public static void printCustomerList(List<Customer>**

**customerList){**

**Iterator<Customer> custIterator = customerList.iterator();**

**//displaying the header of the report System.out.println(" Customer Details"); System.out.println("------------------------------"+**

**"-------------"); System.out.println("cust Id\tcustomer Name\tLoan"+**

**"Availed");**

**while(custIterator.hasNext()){**



**//getting the customer from the list Customer customer=custIterator.next(); System.out.print(customer.getCustomerId()+"\t"); System.out.print(customer.getCustomerName()+**

**"\t\t");**

**//string for displaying if the loan is availed String displayString="NO"; if(customer.isLoanAvailed()){**

**displayString="YES";**

**} System.out.println(displayString);**

**}**

**}**

**}**

**Summary of this assignment:**





 You have learnt how to use methods with collections as parameters and methods returning collections

**Note:**

**Set -** Does not allow duplicate elements

ER/CORP/CRS/ED112/007 CONFIDENTIAL Version No. 1.0 14 of 32

Asreet-Tech.com Limited Lab Guide for <Co urse Nam e>

**Assignment 5: Debug the Assignment**

**Objective:** To learn how to debug a Java program containing Collections and Generics

**Problem Description:** The program populates a set with a set of Customer names and displays it.

**Estimated time: 15 Mins**

**/\***

**\* This file contains the program to be debugged**

**\*/**

**package com.seedBank.debug;**

**import java.util.Set; import java.util.HashSet; import java.util.Iterator;**

**/\*\***

**\* <h3>Description :</h3> This class contains methods to print the**

**\* data present in any set**

**\*** [**@author**](mailto:@author) **E&R Dept.,Asreet-Tech.com Ltd.**

**\*/**

**public class Debug{**

**/\*\***

**\* <h3>Description :</h3> This methods is used to print the**

**\* data in a set onto the console**

**\*** [**@param**](mailto:@param) **set The set containing data**





**\*/**

**public void showRecords(Set<Integer> set){**

**if(!set.isEmpty()){**

**Iterator<T> iterator=set.iterator();**

**while(iterator.hasNext()){ System.*out*.println(iterator.next());**

**}**

**}else {**

**System.*out*.println("Set is an empty set");**

**}**

**}**

**/\*\***

**\* <h3>Description :</h3> The starter method for the application**

**\*** [**@param**](mailto:@param) **Sting [] args the command line arguments**

**\*/**

**public static void main(String args[]){ Set<Object> set=new HashSet<String>();**

**set.add("Lourie");**

ER/CORP/CRS/ED112/007 CONFIDENTIAL Version No. 1.0 15 of 32

Asreet-Tech.com Limited Lab Guide for <Co urse Nam e>

**set.add("Amy"); set.add("Richard"); set.add("Mark");**

**Debug debug=new Debug(); System.*out*.println("Printing the records....."); debug.showRecords(set);**

**}**

**}**

**Summary of this assignment:**

 You have learnt how to debug a Java program containing Collections and

Generics

**Assignment 6: Multithreaded Programming in Java**

**Objective:** To learn how to create threads.

**Problem Description:** Automate the transactions (both Deposit and withdrawal) in SeedBank. You have to use multithre ading approach to handle multiple requests at one time.

Classe s required:



**Customer:** A customer object represents the customers of SeedBank

**Account:** An account object represents an account i n SeedBank **InsufficientBalanceException:** An exception object thrown to indicate if the balance in the account is insufficient for the transaction **UnauthorizedWithdrawalException:** The exception object thrown when the unauthorized person tries to withdraw from an account

**DepositTransaction:** This represents the depositing transaction in SeedBank

**Security:** This represents the authentication done for a transaction in SeedBank **Withdraw Transaction:** This represents the withdrawal transaction done in SeedBank

**SeedBank:** The starter class for the entire application

**Estimated time: 40 Mins**

**Step 1:** Use the Customer class created in the previous assignme nt



**Step 2:** Create an Account class which represents the account of the customer of the

SeedBank

**/\*\***

**\* This file contains an Account bean class**

**\*/**

**package com.seed.account;**

**import com.seed.customer.Customer;**

ER/CORP/CRS/ED112/007 CONFIDENTIAL Version No. 1.0 16 of 32

Asreet-Tech.com Limited Lab Guide for <Co urse Nam e>

**/\*\***

**\*** [**@author**](mailto:@author) **E&R Dept, Asreet-Tech.com Limited**

**\* <h3>Description :</h3> This Account bean class represents an**

**\* account in SeedBank**

**\*/**

**public class Account {**

**/\***

**\* The account No of the account**

**\*/**

**private int accountNo;**

**/\***

**\* The customer Id of the customer to whom the account belongs**

**to.**

**\*/**

**private Customer customer;**

**/\***

**\* The bank balance of the account**

**\*/**

**protected double balance;**

**/\*\***

**\* <h3>Description :</h3> The parameterized constructor if the**

**\* Account class**

**\*** [**@param**](mailto:@param) **accountNo the account Number of the account**





**\*** [**@param**](mailto:@param) **custId the customer Id to whom the account belongs**

**\*** [**@param**](mailto:@param) **balance the account balance**

**\*/**

**public Account(int accountNo,Customer customer,double balance){**

**this.accountNo=accountNo;**

**this.balance=balance;**

**this.customer=customer;**

**}**

**/\*\***

**\* <h3>Description :</h3> The no-arg constructor for Account**

**\*/**

**public Account(){}**

**/\*\***

**\* <h3>Description :</h3> The getter method for the Account**

**\* number**

**\*** [**@return**](mailto:@return) **the accountNo**

**\*/**

**public int getAccountNo() {**

**return accountNo;**

**}**

**/\*\***

**\* <h3>Description :</h3> The setter method for the Account**

ER/CORP/CRS/ED112/007 CONFIDENTIAL Version No. 1.0 17 of 32

Asreet-Tech.com Limited Lab Guide for <Co urse Nam e>

**\* number**

**\*** [**@param**](mailto:@param) **accountNo the accountNo to set**

**\*/**

**public void setAccountNo(int accountNo) {**

**this.accountNo = accountNo;**

**}**

**/\*\***

**\* <h3>Description :</h3> The getter method for the Account**

**\* balance**

**\*** [**@return**](mailto:@return) **the balance**

**\*/**

**public double getBalance() {**

**return balance;**

**}**

**/\*\***

**\* <h3>Description :</h3> The setter method for the Account**

**\* balance**

**\*** [**@param**](mailto:@param) **balance the balance to set**

**\*/**

**public void setBalance(double balance) {**





**this.balance = balance;**

**}**

**/\*\***

**\* <h3>Description :</h3> The getter method for the Account's**

**\* customer**

**\*** [**@return**](mailto:@return) **the customer**

**\*/**

**public Customer getCustomer() {**

**return customer;**

**}**

**/\*\***

**\* <h3>Description :</h3> The setter method for the Account's**

**\* customer**

**\*** [**@param**](mailto:@param) **customer the customer to set**

**\*/**

**public void setCustomer(Customer customer) {**

**this.customer = customer;**

**}**

**}**

**Step 3:** Create an exception class, InsufficientBalanceException that is thrown if there is insufficient balance in the account.

**/\*\***

**\* This file contains the Insufficient balance exception class**

**\*/**

ER/CORP/CRS/ED112/007 CONFIDENTIAL Version No. 1.0 18 of 32

Asreet-Tech.com Limited Lab Guide for <Co urse Nam e>

**package com.seed.exception;**

**/\*\***

**\* <h3>Description :</h3> This exception class is thrown if the**

**\* account has insufficient balance in the account for the transaction**

**\*** [**@author**](mailto:@author) **E&R Dept, Asreet-Tech.com Ltd**

**\*/**

**public class InsufficientBalanceException extends Exception {**

**/\*\***

**\* <h3>Description :</h3>The no-args constructor for the class**

**\*/**

**public InsufficientBalanceException(){**

**//calling the parent constructor super("Insuffient Balance in the account");**

**}**

**}**

**Step 4:** Create UnAuthorizedW ithdrawTransactionException class is an exception that is thrown if the customer is not authorized to do the transaction

**/\*\***





**\* This file contains the UnAuthorizedWithdrawTransactionException**

**\*/**

**package com.seed.exception;**

**/\*\***

**\* <h3>Description :</h3> This exception is thrown when**

**\* the unauthorized transaction is done.**

**\*** [**@author**](mailto:@author) **E&R Dept, Asreet-Tech.com Ltd.**

**\*/**

**public class UnAuthorizedWithdrawTransactionException extends**

**Exception {**

**/\*\***

**\* <h3>Description:<h3> This the no-arg constructor for the**

**\* UnAuthorizedWithdrawTransactionException class.**

**\*/**

**public UnAuthorizedWithdrawTransactionException() {**

**//calling the super class constructor**

**super("The user is unauthorized for the withdrawal");**

**}**

**}**

**Step 5:** Create DepositTransaction class that represents the depositing transaction in the bank.

ER/CORP/CRS/ED112/007 CONFIDENTIAL Version No. 1.0 19 of 32

Asreet-Tech.com Limited Lab Guide for <Co urse Nam e>

**/\*\***

**\* This file contains the Deposit Transaction class**

**\*/**

**package com.seed.transaction;**

**import com.seed.account.Account;**

**/\*\***

**\* <h3>Description :</h3>This class contains methods for depositing**

**\* the amount into an account**

**\*** [**@author**](mailto:@author) **E&R Dept, Asreet-Tech.com Ltd.**

**\*/**

**public class DepositTransaction extends Thread {**

**/\***

**\* The transaction id of the transaction**

**\*/**

**private int transactionId;**

**/\***

**\* The account of which the depositing is done**

**\*/**

**private Account account;**

**/\***

**\* The amount for depositing**

**\*/**





**private double amount;**

**/\***

**\* The customer id of the customer depositing the amount**

**\*/**

**private int customerId;**

**/\*\***

**\* <h3>Description :</h3>The parameterized constructor for**

**\* initializing the object**

**\*** [**@param**](mailto:@param) **transactionId the transaction id for the transaction**

**\*** [**@param**](mailto:@param) **account the account to which the amount has to be**

**\* deposited**

**\*** [**@param**](mailto:@param) **customerId the customer id of the customer depositing**

**\* the amount**

**\*** [**@param**](mailto:@param) **amount The amount of deposit**

**\*/**

**public DepositTransaction(int transactionId,Account account,int customerId,double amount){**

**this.transactionId=transactionId;**

**this.account=account;**

**this.customerId=customerId;**

**this.amount=amount;**

**}**

ER/CORP/CRS/ED112/007 CONFIDENTIAL Version No. 1.0 20 of 32

Asreet-Tech.com Limited Lab Guide for <Co urse Nam e>

**/\*\***

**\* <h3>Description :</h3>This method deposits the amount into the**

**\* given Account**

**\*** [**@param**](mailto:@param) **account the account to which the amount has to be**

**\* deposited**

**\*** [**@param**](mailto:@param) **amount the amount to be deposited**

**\*** [**@return**](mailto:@return) **the remaining balance in the account**

**\*/**

**public void deposit(Account account,double amount){**

**//depositing the amount account.setBalance((account.getBalance()+amount));**

**}**

**/\*\***

**\* <h3>Description :</h3> This method contains job for the thread**

**\*/**

**public void run(){**

**//depositing the amount deposit(account, amount);**

**//displaying the success message**

**System.out.println(transactionId+" transaction "+ "completed!!!"+customerId +" thank you for "+**

**"depositing to the Account "+account.getAccountNo());**

**}**





**}**

**Step 6:** Create a Security class that does the authorization operation

**/\*\***

**\* This file contains the Security class**

**\*/**

**package com.seed.security;**

**import com.seed.account.Account;**

**import com.seed.customer.Customer;**

**import com.seed.exception.UnAuthorizedWithdrawTransactionException;**

**/\*\***

**\*** [**@author**](mailto:@author) **E&R Dept., Asreet-Tech.com Ltd**

**\* <h3>Description :</h3> This class contains methods that authorizes**

**\* a customer for a transaction**

**\*/**

**public class Security {**

**/\*\***

**\* <h3>Description :</h3>This method authorizes a customer to**

**\* perform a particular transaction**

ER/CORP/CRS/ED112/007 CONFIDENTIAL Version No. 1.0 21 of 32

Asreet-Tech.com Limited Lab Guide for <Co urse Nam e>

**\*** [**@param**](mailto:@param) **account the account in the Seed Bank**

**\*** [**@param**](mailto:@param) **customer the customer of the Seed bank**

**\*** [**@throws**](mailto:@throws) **UnAuthorizedWithdrawTransactionException This**

**\* exception is thrown if the account does not**

**\* belong to the customer**

**\*/**

**public void authorization(Account account,Customer customer)**

**throws UnAuthorizedWithdrawTransactionException{**

**/\***

**\* Testing if the customerId of the customer and the**

**\* customerId of the customer to whom the account belongs**

**\* are the same**

**\*/**

**if(account.getCustomer().getCustomerId()!=**

**customer.getCustomerId()){**

**/\* Throwing an exception if the customer Id's doesn't**

**\* match**

**\*/**

**throw new UnAuthorizedWithdrawTransactionException();**

**}**

**}**

**}**





**Step 7:** Creates a W ithdrawTransaction class that represents the withdraw operation in the SeedBank

**/\*\***

**\* This file contains the Withdraw transaction class**

**\*/**

**package com.seed.transaction;**

**import com.seed.account.Account;**

**import com.seed.customer.Customer;**

**import com.seed.exception.InsufficientBalanceException;**

**import com.seed.exception.UnAuthorizedWithdrawTransactionException;**

**import com.seed.security.Security;**

**/\*\***

**\* <h3>Description :</h3> This class is used to perform the withdrawal**

**\* trasaction in the Seed Bank**

**\*** [**@author**](mailto:@author) **E&R Dept, Asreet-Tech.com Ltd**

**\*/**

**public class WithdrawTransaction extends Security implements Runnable{**

**/\***

**\* The transaction id of the transaction**

ER/CORP/CRS/ED112/007 CONFIDENTIAL Version No. 1.0 22 of 32

Asreet-Tech.com Limited Lab Guide for <Co urse Nam e>

**\*/**

**private int transactionId;**

**/\***

**\* The account from which the amount must be drawn**

**\*/**

**private Account account;**

**/\***

**\* The customer who is doing the transaction**

**\*/**

**private Customer customer;**

**/\***

**\* The amount of withdrawal**

**\*/**

**private double amount;**

**/\*\***

**\* <h3>Description :</h3> The parameterized constructor is used**

**\* to initialize the object**

**\*** [**@param**](mailto:@param) **transactionId the transaction id of the transaction**

**\*** [**@param**](mailto:@param) **account the account on which the transaction is done**

**\*** [**@param**](mailto:@param) **customer the customer who is doing the transaction**

**\*** [**@param**](mailto:@param) **amount the amount of withdrawal**

**\*/**





**public WithdrawTransaction(int transactionId,Account account,Customer customer,double amount){**

**this.transactionId=transactionId;**

**this.account=account;**

**this.customer=customer;**

**this.amount=amount;**

**}**

**/\*\***

**\* <h3>Description :</h3> This method does the withdrawal from**

**\* the account**

**\*** [**@param**](mailto:@param) **account the account from which the withdrawal has to be**

**\* done**

**\*** [**@param**](mailto:@param) **amount the amount that has to be withdrawn**

**\*** [**@return**](mailto:@return) **the remaining balance in the account**

**\*** [**@throws**](mailto:@throws) **InsufficientBalanceException This exception is thrown**

**\* if there is insufficient balance in the account**

**\*/**

**public double withdraw(Account account,double amount) throws**

**InsufficientBalanceException{**

**//checking if the account has sufficient balance if(account.getBalance()>=amount){**

**//withdrawing the amount from the account**

**account.setBalance(account.getBalance()-amount);**

ER/CORP/CRS/ED112/007 CONFIDENTIAL Version No. 1.0 23 of 32

Asreet-Tech.com Limited Lab Guide for <Co urse Nam e>

**}else{**

**//throwing an exception if the balance is not**

**//sufficient**

**throw new InsufficientBalanceException();**

**}**

**//returning the balance return account.getBalance();**

**}**

**/\*\***

**\* <h3>Description :</h3> This method does the job in the thread**

**\*/**

**public void run() {**

**try{**

**//checking if the customer is authorized to withdraw**

**//from the given account authorization(account, customer);**

**//withdrawing the amount after authorization**

**double balance=withdraw(account, amount);**

**//displaying the success message**





**System.out.println(transactionId+" transaction"+ "completed!!! and the balance amount is "**

**+balance);**

**}catch(UnAuthorizedWithdrawTransactionException unAuthorizedWithdrawTransactionException){**

**//displaying an appropriate message if the customer is**

**//not authorized to do the transaction**

**System.out.println(transactionId+" transaction "+ "failed!!! and "+**

**unAuthorizedWithdrawTransactionException.getMessage());**

**}catch(InsufficientBalanceException insufficientBalanceException){**

**//display the error message for insufficient balance**

**System.out.println(transactionId+" transaction "+**

**+"failed!!! and your account has insufficient"+**

**+"balance");**

**}**

**}**

**}**

**Step 8:** Create the SeedBank class which has the starter method

**/\*\***

**\* This file contains the Seed bank class**

**\*/**

ER/CORP/CRS/ED112/007 CONFIDENTIAL Version No. 1.0 24 of 32

Asreet-Tech.com Limited Lab Guide for <Co urse Nam e>

**package com.seed;**

**import com.seed.account.Account;**

**import com.seed.customer.Customer;**

**import com.seed.transaction.DepositTransaction;**

**import com.seed.transaction.WithdrawTransaction;**

**/\*\***

**\*** [**@author**](mailto:@author) **E&R Dept., Asreet-Tech.com Ltd**

**\*<h3>Description :</h3> This is the starter class for the transaction**

**\*/**

**public class SeedBank {**

**/\*\***

**\* <h3>Description :</h3>This is the starter method for the**

**\* application**

**\*** [**@param**](mailto:@param) **args The command line arguments for the main method**

**\*/**

**public static void main(String[] args) {**

**//creating the object of the first customer**

**Customer firstCustomer=new Customer(1001,"Raj");**

**//creating the object of the first account**





**Account firstAccount=new Account(2001,firstCustomer,20000);**

**//creating the second account object**

**Customer secondCustomer=new Customer(1002,"Narayan");**

**//creating the second account object**

**Account secondAccount=new**

**Account(2001,secondCustomer,20000);**

**//creating the depositing transaction thread**

**DepositTransaction firstDeposit=new**

**DepositTransaction(9001, secondAccount, 5001, 4000);**

**//starting the first deposit thread firstDeposit.start();**

**//creating the depositing transaction thread**

**DepositTransaction secondDeposit=new**

**DepositTransaction(9002, firstAccount, 5002, 7000);**

**//starting the second thread secondDeposit.start();**

**//creating the withdrawal transaction thread WithdrawTransaction secondWithdrawTransaction=new WithdrawTransaction(9004,firstAccount, secondCustomer,500);**

**Thread secondWithdraw=new**

ER/CORP/CRS/ED112/007 CONFIDENTIAL Version No. 1.0 25 of 32

Asreet-Tech.com Limited Lab Guide for <Co urse Nam e>

**//starting the thread**

**Thread(secondWithdrawTransaction);**

**secondWithdraw.start();// this transaction must fail**

**//creating the withdrawal transaction thread WithdrawTransaction firstWithdrawTransaction=new WithdrawTransaction(9003,firstAccount, firstCustomer, 500); Thread firstWithdrawal=new**

**Thread(firstWithdrawTransaction);**

**//starting the first thread firstWithdrawal.start();**

**}**

**}**

**Summary of this assignment:**





 You have learnt how to create threads by extending Thread class

 You have learnt how to create threads by implementing Runnable

**Note:** Implement java.lang.Runnable if the class exte nds any other class

**Assignment 7: To learn synchronization of Java methods**

**Objective:** To learn how to synchronize methods

**Problem Description:** Make the above program thread safe by synchronization.

**Estimated time: 10 Mins**

**Summary of this assignment:**



 You have learnt how to synchronize the Java methods

**Note:** Use **synchronized** keyword to synchronize



ER/CORP/CRS/ED112/007 CONFIDENTIAL Version No. 1.0 26 of 32

Asreet-Tech.com Limited Lab Guide for <Co urse Nam e>

**Assignment 8: Learning how to debug a multithreaded Java program**

**Objective:** To learn how to debug a multithreaded program

**Problem Description:** SeedBank wants to automate the loan disbursing operation I the

Bank. The loan transaction must be multithreaded and thre adsafe.

**Estimated time: 20 Mins**

**Step 1:** Create an Exception class, InsufficientFundException that is thrown if bank has insufficient funds for the transaction

**/\*\***

**\* This file contains the InsufficientFundException class**

**\*/**

**package com.seed.exception;**

**/\*\***

**\* <h3>Description :</h3> This exception is thrown if the Bank has**

**\* insufficient fund to process the transaction.**

**\*** [**@author**](mailto:@author) **E&R Dept.,Asreet-Tech.com Ltd**





**\*/**

**public class InsufficientFundException extends Exception {**

**/\*\***

**\* <h3>Description :</h3>This is a parameterized constructor**

**\*** [**@param**](mailto:@param) **msg The message that has to set for the exception**

**\*/**

**public InsufficientFundException(String msg){**

**//passing the message to the parent constructor**

**super(msg);**

**}**

**}**

**Step 2:** Create the BankFund class which represents the fund allocated for the loan disbursi ng operations in the SeedBank

**/\*\***

**\* This file contains the BankFund class**

**\*/**

**package com.seed.fund;**

**import com.seed.exception.InsufficientFundException;**

**/\*\***

ER/CORP/CRS/ED112/007 CONFIDENTIAL Version No. 1.0 27 of 32

Asreet-Tech.com Limited Lab Guide for <Co urse Nam e>

**\* <h3>Description :</h3>This class represents the fund reserved by**

**\* the SeedBank for disbursing loans.**

**\*** [**@author**](mailto:@author) **E&R Dept.,Asreet-Tech.com Ltd**

**\*/**

**public class BankFund {**

**/\***

**\* This represents the fund allocated by the SeedBank for Loans**

**\*/**

**private double fund;**

**/\*\***

**\* <h3>Description :</h3>This is the parameterized constructor**

**\*** [**@param**](mailto:@param) **fund The initial fund of SeedBank**

**\*/**

**public BankFund(double fund){**

**//setting the fund**

**this.fund=fund;**

**}**

**/\*\***

**\* <h3>Description :</h3> This method is used to enquire if the**

**\* requested amount of fund is available to disburse the loan.**

**\*** [**@param**](mailto:@param) **loanAmount this is the amount required to disburse a**

**\* loan**

**\*** [**@return**](mailto:@return) **returns 0 if sufficient fund is available for**

**\* disbursing the loan**





**\*** [**@throws**](mailto:@throws) **InsufficientFundException This exception is thrown if**

**\* the bank does not have sufficient fund in the Bank for the**

**\* transaction**

**\*/**

**public int checkFund(double amount) throws**

**InsufficientFundException{**

**//checking if the fund is available if(fund<amount){**

**//throwing InsufficientFundException**

**throw new InsufficientFundException("Insufficient "**

**+"fund");**

**}**

**/\*\***

**}**

**//returning the success code return 0;**

**\* <h3>Description :</h3>This method debits the input amount from**

**\* the fund for the transaction**

**\*** [**@param**](mailto:@param) **amount the amount required for the transaction**

**\*** [**@return**](mailto:@return) **Returns 0 if the transaction is successful**

**\*/**

**public double debitFund(double amount){**

ER/CORP/CRS/ED112/007 CONFIDENTIAL Version No. 1.0 28 of 32

Asreet-Tech.com Limited Lab Guide for <Co urse Nam e>

**//debiting from the fund fund-=amount;**

**//returning the success status code return fund;**

**}**

**}**

**Step 3:** Create the LoanTransaction class that does the loan transaction in SeedBank





**/\*\***

**\* This file contains the LoanTransaction class**

**\*/**

**package com.seed.transaction;**

**import com.seed.exception.InsufficientFundException;**

**import com.seed.fund.BankFund;**

**/\*\***

**\* <h3>Description :</h3>This class represents the loan transaction**

**\* performed in the SeedBank**

**\*** [**@author**](mailto:@author) **E&R Dept.,Asreet-Tech.com Ltd**

**\*/**

**public class LoanTransaction extends Runnable {**

**/\***

**\* This represents the SeedBank fund on which the transaction**

**\* is being performed**

**\*/**

**private BankFund bankFund;**

**/\***

**\* This represents the customer id of the customer who is**

**\* availing loan**

**\*/**

**private int customerId;**

**/\***

**\* This represents the amount needed for the loan**

**\*/**

**private double loanAmount;**

**/\*\***

**\* <h3>Description :</h3> This is the parameterized constructor**

**\* of the LoanTransaction class**

**\*** [**@param**](mailto:@param) **bankFund the bankFund from where the transaction will**

**\* get the fund**

**\*** [**@param**](mailto:@param) **customerId the customer Id of the customer who is**

**\* requesting for the loan transaction**

**\*** [**@param**](mailto:@param) **loanAmount this represents the loan amount**

ER/CORP/CRS/ED112/007 CONFIDENTIAL Version No. 1.0 29 of 32

Asreet-Tech.com Limited Lab Guide for <Co urse Nam e>

**\*/**

**public LoanTransaction(BankFund bankFund,int customerId, double loanAmount){**

**//setting the bankFund this.bankFund=bankFund;**

**//setting the customer ID**

**this.customerId=customerId;**

**//setting the amount this.loanAmount=loanAmount;**

**}**

**/\* (non-Javadoc)**

**\*** [**@see**](mailto:@see) **java.lang.Runnable#run()**

**\*/**

**public void start() {**

**try {**

**//checking if the bank has sufficient fund bankFund.checkFund(loanAmount);**

**//document verification delay**





**Thread.sleep((long)(2000\*Math.random()));**

**//disbursing the loan**

**double fundAvailable=bankFund.debitFund(loanAmount);**

**//displaying the success message**

**System.out.println(customerId+"The loan is disbursed."**

**+"Please collect the cash from the Teller"); System.out.println("Fund left in the bank "+**

**fundAvailable);**

**} catch (InsufficientFundException insufficientFundException) {**

**//displaying the error message**

**System.out.println("Sorry!!!"+customerId+" Please "+**

**"try some days later as we are short of funds");**

**}**

**}**

**}**

**Step 5:** Create the SeedBank class that is the starter class for the application

**/\*\***

**\* This file contains the Starter class**

**\*/**

**package com.seed.general;**

ER/CORP/CRS/ED112/007 CONFIDENTIAL Version No. 1.0 30 of 32

Asreet-Tech.com Limited Lab Guide for <Co urse Nam e>

**import com.seed.fund.BankFund;**

**import com.seed.transaction.LoanTransaction;**

**/\*\***

**\* <h3>Description :</h3>The starter class of the entire application**

**\*** [**@author**](mailto:@author) **E&R Dept.,Asreet-Tech.com Ltd.**

**\*/**

**public class SeedBank {**

**/\*\***

**\* <h3>Description :</h3> This method is the starter method for**

**\* the entire application**

**\*** [**@param**](mailto:@param) **args The commands line arguments**

**\*/**

**public static void main(String[] args) {**

**//Initializing the SeedBank fund**

**BankFund seedBankFund=new BankFund(9000000);**



**//creating an array of transactions**

**LoanTransaction[] loanTransactions=new**

**LoanTransaction[100];**

**for(int index=0;index<loanTransactions.length;index++){**

**loanTransactions[index]=new**

**LoanTransaction(seedBankFund, (1000+index), 250000);**

**}**

**//requesting for the loan at different branches**

**for(int index=0;index<loanTransactions.length;index++){**

**loanTransactions[index].run();**

**}**

**}**

**}**

**Step 6:** The SeedBank class and the LoanTransaction class co ntai ns compilation errors. Fix those errors.



**Step 6:** Make the loan transactions multithre aded and threadsafe operation. **The constraint is that you cannot modify the BankFund class.**

**Summary of this assignment:**

 You have learnt how to debug a multithreaded Java program

ER/CORP/CRS/ED112/007 CONFIDENTIAL Version No. 1.0 31 of 32

Asreet-Tech.com Limited Lab Guide for <Co urse Nam e>

**Note:** You have to synchronize the threads with respect to the



BankFund object

**Assignment 9: Exercises for Self Review**

Solve the self review questions provided





ER/CORP/CRS/ED112/007 CONFIDENTIAL Version No. 1.0 32 of 32